UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

MARCIO ALEXANDRE PEREIRA DA SILVA

Uma Arquitetura de Software Adaptativa baseada em Arquétipo OpenEHR para Sistemas de Informação em Saúde

Recife
2022

MARCIO ALEXANDRE PEREIRA DA SILVA

Uma Arquitetura de Software Adaptativa baseada em Arquétipo OpenEHR para Sistemas de Informação em Saúde

Tese apresentada ao Programa de Pós-Graduação em Ciências da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Doutor em Ciências da Computação. Área de Concentração: Banco de Dados.

Orientadora: Profª. Drª. Valéria Cesário Times.

Coorientador: Prof. Dr. Paulo Caetano da Silva

Recife

2022

Catalogação na fonte
Bibliotecária Monick Raquel Silvestre da S. Portes, CRB4-1217

S586a Silva, Marcio Alexandre Pereira da
Uma arquitetura de software adaptativa baseada em arquétipo OpenEHR para sistemas de informação em saúde / Marcio Alexandre Pereira da Silva. – 2022.
181 f.:il., fig, tab.

Orientadora: Valéria Cesário Times.
Tese (Doutorado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2022

Inclui referências e anexos.

1. Banco de dados. 2. Sistemas de informação. I. Times, Valéria Cesário (orientadora). II. Título.

025.04 CDD (23. ed.) UFPE - CCEN 2022-188

Marcio Alexandre Pereira da Silva

“Uma Arquitetura de Software Adaptativa baseada em Arquétipo OpenEHR para Sistemas de Informação em Saúde”

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Doutor em Ciência da Computação. Área de Concentração: Banco de Dados.

Aprovado em: 28/07/2022.

_____

Orientadora: Profa. Dra. Valéria Cesario Times

BANCA EXAMINADORA

_____

Prof. Dr. Vinícius Cardoso Garcia
Centro de Informática/UFPE

_____

Prof. Dr. Luciano de Andrade Barbosa
Centro de Informática/UFPE

_____

Prof. Dr. Ricardo João Cruz Correia
Departamento de Medicina / Universidade do Porto

_____

Profa. Dra. Claudia Maria Cabral Moro Barra
Programa de Pós-Graduação em Tecnologia em Saúde / PUC/PR

_____

Prof. Dr. Gustavo Henrique Matos Bezerra Motta
Departamento de Informática / UFPB

Dedico o resultado deste esforço aos meus pais, Jasiel (in memoriam) e Lucenilda, e a minha esposa, Fernanda Oliveira.

AGRADECIMENTOS

Agradeço primeiramente ao autor da vida, Deus, por me proporcionar todas as condições necessárias para a realização deste tão sonhado e árduo doutorado.

Aos meus pais, Jasiel (*in memoriam*) e Lucenilda, agradeço eternamente pelo amor, suporte incondicional e incentivos que me guiaram e me mantiveram firme durante esta longa jornada. À Fernanda Oliveira, minha esposa, obrigado pelo amor, suporte e incentivos constantes na conclusão deste sonho. Aos meus irmãos, Jasiel Jr e Clara, agradeço a amizade que sempre tivemos. Aos meus sobrinhos Marcia Regina, Theo, Gael, Alice e Henry pela alegria e energia que trazem a qualquer ambiente. Aos meus tios Samuel Pereira e Assumpção *(in memoriam*), agradeço o carinho e os conselhos, também por toda ajuda à nossa família. Devido essa ajuda, compramos nosso primeiro computador e, então, pude iniciar meus primeiros passos (ainda adolescente) na área que hoje faço meu doutorado. A minha tia Raquel, sou grato pelo gentil e carinhoso acolhimento em sua casa.

Aos demais familiares e amigos, obrigado pelo apoio, incentivo e pela convivência harmoniosa ao longo destes tantos anos. É verdade que a vida separou fisicamente alguns de nós, mas lembro de todos com muito carinho. Não serei capaz de citá-los individualmente, mas fica registrado o meu reconhecimento e agradecimento por fazerem parte de minha vida.

Aos amigos Paulo Caetano e André Araújo, parceiros de inúmeros artigos e projetos, muito obrigado pela amizade, pelas orientações, pelos conselhos e por toda ajuda ao longo dessa caminhada.

À professora Valéria, minha eterna gratidão por ter sido uma orientadora tão generosa, a quem tenho grande admiração. Expresso aqui meus sinceros agradecimentos pela oportunidade, por acreditar em mim, por me fazer um ser humano melhor e por me acolher com tanto carinho e paciência. Levarei comigo todos os ensinamentos recebidos, e lembrarei sempre com muito carinho desse período da minha vida de muito trabalho e aprendizado.

A Coordenação de Amparo à Pesquisa do Ensino Superior – CAPES, agradeço pelo incentivo financeiro proporcionado.

RESUMO

A adaptabilidade de software confere, aos sistemas de informação, a capacidade de ajustar o seu comportamento ou estrutura, mediante mudanças tanto em seu ambiente de execução, como em seus requisitos de software. É uma abordagem eficaz para lidar com ambientes dinâmicos e mutáveis, como o ambiente no qual estão inseridos os sistemas de informação em saúde (SIS) que utilizam arquétipo, um padrão openEHR. Um arquétipo pode ser definido como uma expressão computacional representada por restrições de domínio, que modelam os atributos de dados e dão significado semântico aos registros eletrônicos de saúde. Nesse contexto de ambientes mutáveis, os ajustes manuais (na implementação, nos testes e na implantação), efetuados nos SIS com o intuito de deixá-los em conformidade com novas demandas, significam um prejuízo bilionário. Por outro lado, adicionar recurso de adaptabilidade de software a esses SIS também complica significativamente a fase de implementação e traz grandes desafios às práticas de Engenharia de Software. Desta forma, esta tese apresenta uma solução que fornece um mecanismo de adaptabilidade aos SIS que utilizam arquétipos. Essa solução é uma arquitetura de software que adapta seus componentes, em tempo de execução, conforme os arquétipos utilizados no SIS. Essa arquitetura é composta por modelo de dados, interface do usuário, adaptador em tempo de execução e provedor CRUD. O modelo de dados contém o artefato computacional utilizado para representar o RES, isto é, arquétipos. A interface do usuário interage com o utilizador do software e possui dois componentes: formulário com arquétipo e controlador. O adaptador em tempo de execução tem dois componentes: identificador e localizador. Esse adaptador permite que componentes da arquitetura sejam adaptados baseados nos arquétipos utilizados na interface do usuário. O provedor CRUD são componentes da arquitetura proposta, os quais são serviço que executa operações de escrita, leitura, atualização e exclusão de dados representados por arquétipos. Além disso, a ferramenta AdaptiveHIS foi desenvolvida para avaliação da adaptabilidade da solução proposta nesta tese. Essa ferramenta permite construir uma aplicação de SIS com base na arquitetura proposta. Nessa avaliação, comparou-se a arquitetura proposta com outras arquiteturas construídas por cinco ferramentas que se encontram no estado da arte: EhrScape, EhrBase, MARCIA, Template4EHR e

Microservice4EHR. Também, mensurou-se o quão adaptável essas arquiteturas são em um ambiente mutável. Como resultado, a arquitetura apresentada nesta tese aumenta em até 62% a adaptabilidade dos SIS. Por fim, esta proposta permite que SIS sejam adaptáveis em ambientes dinâmicos e mutáveis, tornando menos dispendiosa a manutenção desses softwares, uma vez que não há necessidade de alocação de recursos manuais para adequar o software às novas demandas.

Palavras-chave: sistemas de informação em saúde (SIS); adaptabilidade; arquitetura de software; arquétipo openEHR.

ABSTRACT

Software adaptability gives information systems the ability to adjust their behaviour or structure through changes in their execution environment and in their software requirements. This adaptability is an effective approach to deal with dynamic and changing environments, such as the environment in which health information systems (HIS), that use archetype, are inserted. An archetype is an openEHR standard and can be defined as a computational expression represented by domain constraints, which model data attributes and give semantic meaning to electronic health records (EHR). In this context of changing environments, manual adjustments (in the implementation, in the tests and in the deployment) applied in the HIS in order to make them comply with new demands, mean a billionaire loss. Conversely, adding software adaptability capability to these HIS also significantly complicates the implementation phase and brings major challenges to Software Engineering practices. Thus, this thesis presents a solution that provides an adaptability mechanism to HIS that use archetypes. This solution is a software architecture that adapts its components, at runtime, according to the archetypes used in the HIS. This architecture is composed by data model, user interface, runtime adapter, and CRUD provider. Data model contains the computational artifact used to represent the EHR, which means, the archetypes. User Interface interacts with the user and has two components: archetype-based form and controller. Runtime adapter has two components: identifier and locator. This adapter allows architecture components to be adapted based on the archetypes used in the user interface. CRUD provider is a component of the proposed architecture, which is a set of services that perform operations of writing, reading, updating and deleting data represented by archetypes. In addition, the AdaptiveHIS tool was developed for evaluation of the adaptability of the solution proposed in this thesis. This tool allows building a HIS application based on the proposed architecture. In this evaluation, the proposed architecture was compared with other architectures built by five state-of-the-art tools: EhrScape, EhrBase, MARCIA, Template4EHR and Microservice4EHR. Also, it was measured how adaptable these architectures are in a changing environment. As a result, the architecture presented in this thesis increases the

adaptability of HIS by up to 62%. Finally, this proposal allows HIS to be adaptable in dynamic and changing environments, making the maintenance of these software less expensive, since there is no need to allocate manual resources to adapt the software to new demands.

Keywords: healthcare information system (HIS); adaptability; software architecture; openEHR archetype.

LISTA DE FIGURAS

Figura 1 -	Desperdícios na área de TI avaliados em dólares, em todo o mundo.	23
Figura 2 -	Um formulário web construído utilizando três arquétipos.	25
Figura 3 -	Etapas da metodologia.	27
Figura 4 -	Mapa mental do arquétipo *Person*.	31
Figura 5 -	Arquétipo `person.v0`, seus atributos e algumas restrições de domínio.	32
Figura 6 -	Exemplo de comunicação com um serviço web.	33
Figura 7 -	Visão geral da arquitetura proposta em FERREIRA et al. (2012).	45
Figura 8 -	Ferramenta LIU EEE.	46
Figura 9 -	Visão geral da arquitetura proposta em REIS et al. (2018).	48
Figura 10 -	Estudos sobre o uso de arquétipo e a promoção da independência conceitual.	49
Figura 11 -	Ferramenta EhrScape.	51
Figura 12 -	Visão geral da arquitetura proposta na Template4EHR.	53
Figura 13 -	Ferramenta CloudEHRServer.	54
Figura 14 -	Sistema Adaptativo baseado em aprendizagem por reforço.	56
Figura 15 -	Ferramenta MAA-FMS.	57
Figura 16 -	Caso de uso do sistema IDPT adaptativo.	58
Figura 17 -	Arquétipo *Patient Admission*.	65
Figura 18 -	Mapeamento de campos de um formulário web com os respectivos atributos de arquétipos.	69
Figura 19 -	Formulário web e API construídos com os arquétipos `person.v0` e `admission.v0`.	71
Figura 20 -	Interface do usuário (GUI) e API construídos com	73

diferentes arquétipos.
Figura 21 - Nível de contexto da arquitetura proposta: pessoas, sofware, modelo de dados e suas relações 75
Figura 22 - Nível de *containers* da arquitetura proposta e fluxo de dados entre os *containers* e componentes 76
Figura 23 - Nível de componentes de *Formulário com Arquétipo*. 78
Figura 24 - Nível de componentes *Controlador* e suas interações com *Adaptador e Provedor*. 79
Figura 25 - Nível de componentes *Adaptador em Tempo de Execução* e seu fluxo de dados. 81
Figura 26 - Nível de componentes e fluxo de dados de *provedor CRUD*. 81
Figura 27 - Visão de modelo de dados da arquitetura proposta nesta tese. 83
Figura 28 - Diagrama de casos de uso do cenário funcional mostrado na Seção 4.1. 86
Figura 29 - Diagrama de sequência do comportamento do cenário funcional. 87
Figura 30 - Diagrama de sequência do comportamento da arquitetura proposta. 87
Figura 31 - Arquétipo *Blood Pressure*. 95
Figura 32 - Uma GUI construída com os arquétipos `person.v0` e `admission.v0`. 96
Figura 33 - Diagrama das arquiteturas de software geradas pelas ferramentas EhrScape, EhrBase, e MARCIA, para o cenário da UBS. 97
Figura 34 - Diagrama das arquiteturas geradas pelas ferramentas Template4EHR e Microservice4EHR, no cenário da UBS. 98
Figura 35 - Diagrama da arquitetura gerada pela AdaptiveHIS, no cenário da UBS. 99
Figura 36 - Uma GUI construída com os arquétipos `height.v1`, 100

`body_weight.v1`, `lab_test-blood_glucose.v1` e `pulse.v1`.
Figura 37 - Diagrama das arquiteturas geradas por EhrScape, EhrBase e MARCIA, no cenário dos hemocentros. 101
Figura 38 - Diagrama das arquiteturas geradas pelas ferramentas Template4EHR e Microservice4EHR, no cenário dos hemocentros. 103
Figura 39 - Diagrama da arquitetura gerada pela AdaptiveHIS, no cenário dos hemocentros. 104
Figura 40 - Nova versão do SIS da UBS. 105
Figura 41 - Nova versão do SIS dos hemocentros. 107
Figura 42 - Erros, falhas ou inconsistências, no cenário da UBS. 109
Figura 43 - Erros, falhas ou inconsistências, no cenário dos hemocentros. 109
Figura 44 - Novos componentes após mudanças dos SIS, no cenário da UBS. 112
Figura 45 - Novos componentes após mudanças dos SIS, no cenário do hemocentro. 112
Figura 46 - Três *campos de dados* do formulário Medidas do Corpo, e seus respectivos códigos fontes em HTML. 119
Figura 47 - Fragmentos de códigos fontes do formulário "*Medidas do Corpo*" e seu respectivo *Controlador*. 121
Figura 48 - Documentação de API construída para executar o *Adaptador em Tempo de Execução*. 122

LISTA DE TABELAS

Tabela 1 -	Análise comparativa dos trabalhos correlatos.	60
Tabela 2 -	Análise de adaptabilidade dos cenários da UBS e Hemocentros.	113
Tabela 3 -	Diretrizes para replicação da avaliação desta tese.	115

LISTA DE ABREVIATURAS E SIGLAS

ADL	*Archetype Definition Language*
API	*Application Programming Interface*
AQL	*Archetype Query Language*
ASAC	*Archetype-based Software Architecture Coupling*
C4	*Context, container, component, code*
CKM	*Clinical Knowledge Manager*
CRUD	*Create-Read-Update-Delete*
COVID-19	*Coronavirus Disease 2019*
DHSA	*Decoupled Health Software Architecture*
EHR	*Electronic Healthcare Record*
FHIR	*Fast Healthcare Interoperability Resources*
GUI	*Graphical User Interface*
HATEAOS	*Hypermedia as the Engine of Application State*
HIS	*Healthcare Information System*
HL7	*Health Level Seven*
HTML	*Hypertext Markup Language*
HTTP	*Hypertext Transfer Protocol*
HSML	*Health Sofware Modeling Language*
IaaS	*Infrastructure as a Service*
IC	Independência Conceitual
IDPT	*Internet-Delivered Psychological Treatments*
IEC	*International Electrotechnical Commission*
IEEE	*Institute of Electrical and Electronics Engineers*
ISO	*International Organization for Standardization*
JSON	*Javascript Object Notation*
LIU EEE	*Linköping University Educational* EHR *Environment*
LSA	*Level system adaptability*
PaaS	*plataform as a service*
MAA-FMS	*Multi-Agent Autonomic Fetus Monitoring System*
MARCIA	*Manejo de Registro Clínico Aplicado*

MDD	*Model-Driven Development*
MIMIC	*Medical Information Mart for Intensive Care*
MSA	*MicroService Architecture*
NoSQL	*Not Only SQL*
RES	Registro eletrônico de Saúde
RF	Requisito Funcional
RNF	Requisito Não Funcional
REST	*Representational State Transfer*
SaaS	*Software as a Service*
SAHIB	*Secure Agent based Health Information systems Brokering*
SIS	Sistemas de Informação em saúde
SOA	*Service-Oriented Architecture*
SQL	*Structured Query Language*
UBS	Unidade Básica de Saúde
URI	*Uniform Resource Identifier*
URL	*Uniform Resource Locator*
XML	*eXtensible Markup Language*

SUMÁRIO

1	**INTRODUÇÃO**	**18**
1.1	**Contextualização**	**18**
1.2	**Motivação**	**21**
1.3	**Problema de Pesquisa e Hipótese**	**25**
1.4	**Objetivos**	**26**
1.5	**Metodologia**	**27**
1.6	**Sumário de Contribuições**	**28**
1.7	**Organização do Trabalho**	**29**
2	**FUNDAMENTAÇÃO TEÓRICA**	**30**
2.1	**Arquétipo**	**30**
2.1.1	***Atributos***	**31**
2.1.2	***Restrições de Domínio (Constraints)***	**32**
2.2	**Serviços WEB**	**33**
2.2.1	***Protocolo de Comunicação com Serviços***	**34**
2.2.2	***Arquitetura Cliente-Servidor***	**35**
2.2.3	***Arquiteturas de Software baseada em Serviços***	**36**
2.3	**Computação em Nuvem**	**37**
2.4	**Adaptabilidade de Software**	**38**
2.5	**Considerações Finais**	**42**
3	**TRABALHOS CORRELATOS**	**44**
3.1	**Arquiteturas de Software baseadas em Arquétipos**	**44**
3.2	**Ferramentas para Desenvolvimento de SIS que Usam Arquétipos**	**51**
3.3	**Arquiteturas de Software Adaptativas para Área de Saúde**	**52**
3.4	**Discussão Sobre os Trabalhos Investigados**	**60**
3.5	**Considerações Finais**	**61**
4	**UMA ARQUITETURA DE SOFTWARE ADAPTATIVA BASEADA EM ARQUÉTIPOS PARA SISTEMAS DE INFORMAÇÃO EM SAÚDE**	**63**
4.1	**Cenário Funcional**	**64**
4.2	**Requisitos de Software**	**67**
4.2.1	***Requisitos Funcionais e Não Funcionais***	**67**
4.3	**Arquitetura de Software Proposta**	**70**

4.3.1 Mecanismo de Adaptação 70
4.3.2 Níveis da Arquitetura 73
4.3.3 Visão de Modelo de Dados 82
4.3.4 Comportamentos da Arquitetura 84
4.4 Considerações Finais 88
5 AVALIAÇÃO E RESULTADOS 89
5.1 Planejamento 92
5.2 Primeira Etapa: Construção dos SIS 94
5.2.1 Cenário da UBS 95
5.2.2 Cenário dos Hemocentros 100
5.3 Segunda e Terceira Etapa: Mudança de Arquétipos e Análise 103
5.4 Resultados 106
5.5 Limição da Métrica LSA 114
5.6 Replicabilidade da Avaliação 114
5.7 Respostas às Questões de Pesquisa 118
5.8 Ameaças à Validade da Avaliação 122
5.9 Considerações Finais 123
6 CONCLUSÃO 125
6.1 Considerações Finais 125
6.2 Contribuições 126
6.3 Limitações 127
6.4 Trabalhos Futuros 128
REFERÊNCIAS 131
ANEXO A – CÓDIGO DO PROVEDOR CRUD 155
`BLOOD_PRESSURE.V2`
ANEXO B – CÓDIGO DA FERRAMENTA ADAPTIVEHIS 160

1 INTRODUÇÃO

Este capítulo tem a finalidade de situar o tema abordado neste trabalho. A Seção 1.1 contextualiza e discute a importância das arquiteturas de software de sistemas de informação em saúde (SIS) que usam arquétipos para representação de seus dados. A Seção 1.2 motiva a utilização de *adaptabilidade de software* nesse contexto.

A Seção 1.3 descreve o problema de pesquisa e a hipótese. As seções 1.4 e 1.5 apresentam os objetivos almejados e a metodologia usada no desenvolvimento desta tese, respectivamente. A Seção 1.6 resume as contribuições deste trabalho. Por fim, a Seção 1.7 descreve como esta tese está organizada.

1.1 Contextualização

Engenharia de Software é a aplicação sistemática de metodologias e abordagens de engenharia para o desenvolvimento de software de qualquer domínio (por exemplo: de saúde). Por sua vez, o desenvolvimento de software é um processo pelo qual artefatos (ou componentes) de software são criados, estruturados e relacionados, mediante uso de recursos computacionais como: linguagens de computador, padrões, modelos, bibliotecas e *frameworks*. A estruturação e a relação desses componentes de software definem o que a Engenharia de Software chama de arquitetura de software (ou somente “arquitetura”) (PRESSMAN & MAXIM, 2019).

Arquitetura de software consiste em definir os artefatos de software que compõem um software. Nessa definição, especifica-se as propriedades desses artefatos e os relacionamentos com outros componentes (BASS et al., 2022). Existe um consenso de que a arquitetura de software desempenha um papel central no desenvolvimento de software e um papel importante no alcance dos objetivos de negócios das organizações, cujas atividades administrativas são executadas com o auxílio de softwares (ou sistemas de informação) (CHA et al., 2016).

Sistemas de informação são construídos para satisfazer os objetivos de negócios das organizações. Desta forma, a arquitetura de software é uma ponte entre esses objetivos de negócios (geralmente abstratos) e o sistema de informação final resultante (concreto). Embora o caminho que parte de objetivos abstratos para sistemas concretos possa ser complexo, um aspecto a ser considerado é que as arquiteturas de software podem ser projetadas, analisadas, documentadas e implementadas usando técnicas conhecidas que dão suporte ao alcance desses objetivos de negócios (BASS et al., 2022).

Na área de Informática em Saúde, essa abordagem sobre arquitetura de software não é diferente. Sabe-se que SIS possuem um conjunto de componentes que atuam de forma integrada, através de mecanismos de coleta, processamento, análise e transmissão de dados para implementar processos de decisões na área da saúde (WHO, 2022). De uma maneira geral, os SIS são tecnicamente auxiliados pela Engenharia de Software (DETRO et al., 2020). Por exemplo, a concepção (ou construção) de um SIS somente é factível porque há um processo de desenvolvimento de software (STOCKER, 2018).

Isso significa que, como descrito anteriormente, a arquitetura de software também tem um papel central no processo de desenvolvimento de um SIS, no qual se aplicam técnicas e modelos de arquitetura especificados no domínio de Engenharia de Software (SILVA et al., 2019; MOTTA et al., 2020).

Ao longo dos últimos anos, a indústria de software, as instituições governamentais e a academia têm debatido o uso de padrões da área da saúde no desenvolvimento de software e na especificação de arquitetura de software (SILVA et al., 2020a). O propósito disso é melhorar a qualidade dos serviços prestados, aumentar a resiliência dos SIS às constantes mudanças, e popularizar o acesso às informações do registro eletrônico de saúde (RES) (ARAÚJO et al., 2020). Dado este cenário, profissionais e cientistas têm debatido e aplicado normas, padrões e tecnologias emergentes na área da saúde sobre as técnicas de desenvolvimento de software, especificações de arquiteturas de software e, também, sobre o ciclo de vida de um SIS (SILVA et al., 2022).

Então, verificam-se estudos cujas arquiteturas de software têm sido especificadas a partir da utilização de padrões das áreas da saúde e da Engenharia

de Software, em conjunto. Essa abordagem resulta na construção de SIS em conformidade com boas práticas de ambos os domínios (ARAÚJO et al., 2016; MUSLIM et al., 2017; GOMES et al., 2018; ARAÚJO et al., 2020; OLIVEIRA et al., 2021; SILVA et al., 2022).

Dentre esses padrões da Saúde, o arquétipo openEHR (ou somente "arquétipo") (openEHR, 2022) se destaca por sua adoção e reconhecimento em nível mundial (ISO, 2019; SCHLIEMANN et al., 2019; GOMES et al., 2021; LADAVA & NORRIS, 2022; MUSZYNSKI et al., 2022; POHJONEN, 2022; JONES et al., 2022). Arquétipo é um padrão especificado pela fundação openEHR para representação de registros eletrônicos de saúde (openEHR, 2022). Arquétipo é utilizado para remodelar os conceitos clínicos de sistemas legados, implementar o RES em sistemas de banco de dados e definir os requisitos de dados e as terminologias (ARAÚJO et al., 2020). Arquétipo também é usado como padrão de dados na comunicação entre um conjunto de SIS, ou entre um SIS e um repositório de dados centralizado (ISO, 2019).

No que se refere a utilização de arquétipos e a representação de processos clínicos, alguns autores propõem a utilização da ferramenta *Business Process Model and Notation* (BPMN), uma ferramenta utilizada na Engenharia de Software para modelar processos computacionais (IGLESIAS et al., 2022). Também, a openEHR propõe a *Task Planning Visual Modelling Language* (TP-VML), uma solução baseada em BPMN, *Activity-Based Design* (ABD), *Case Management Model and Notation* (CMMN) e arquétipos, com a qual é possível criar diagramas descrevendo fluxos de trabalho da Saúde (OpenEHR, 2022).

No processo de desenvolvimento de software, diversos estudos encontrados no estado da arte propõem a construção de artefatos de software com base nas especificações dos arquétipos (SILVA et al., 2020b; FRADE et al., 2022). Essas propostas significam boas práticas na manutenção dos SIS já que se encontram em conformidade com os padrões internacionais de saúde (ARAÚJO et al., 2020; COUTO et al., 2022; KRYSZYN et al., 2022).

Além disso, diversos estudos propõem especificações de arquiteturas de software (ou parte delas) em consonância com as especificações de arquétipos (SACHDEVA et al., 2011; SANDVAL et al., 2013; BETTER, 2017; GOMES et al., 2018; ARAÚJO et al., 2016; ARAÚJO et al., 2018; REIS et al., 2018; SILVA et al., 2019; SILVA et al., 2020a; SILVA et

al., 2020b; SILVA et al., 2022; ARAÚJO et al., 2020; CABOLABS, 2022; PALIWAL et al., 2022; FRADE et al., 2022). O propósito dessa abordagem é usufruir do benefício de adotar um padrão internacional da saúde em conjunto com abordagens modernas de Engenharia de Software. Dado esse contexto, a seguir a motivação desse trabalho é descrita.

1.2 Motivação

A área da Saúde é composta em sua maioria por sistemas de informação legados e isso pode levar a ineficiências. Embora governos e empresas médicas sejam os principais investidores há muito tempo, grandes empresas de tecnologia, incluindo Google e Microsoft, começaram a investir mais na modernização dessa área. Além disso, com a pandemia do COVID-19 em todo o mundo, as discussões sobre novas tecnologias para suporte da área da Saúde estão surgindo cada vez mais (HAN & LEE, 2021).

A área da saúde possui um ambiente cujos protocolos (também chamados de “protocolos de saúde”) estão em constante modificação. A pandemia do COVID-19, por exemplo, trouxe vários desafios e evidenciou algumas limitações. Para reduzir a mortalidade devido ao COVID-19, protocolos de saúde foram modificados em todo o mundo para atender às atividades de atendimento clínico ao paciente.

Essas mudanças de protocolos visaram aumentar a eficácia do tratamento, como a verificação dos níveis de saturação de oxigênio, a realização imediata de exames laboratoriais para detecção de COVID-19 e outras infecções, a introdução de terapia de suporte ventilatório e notificações de pacientes infectados para melhorar a tomada de decisão por agências de saúde (WHO, 2022; G. BRASILEIRO, 2020a; G. BRASILEIRO, 2020b).

Com essa pandemia, uma limitação dos estudos encontrados no estado da arte é a conformidade dos SIS (em uso nas unidades de saúde) com esses novos protocolos. No contexto de SIS, mudanças nos protocolos de saúde desencadeiam outras alterações as quais impactam o ciclo de vida de um SIS, por exemplo: alterações nos requisitos de software (SILVA et al., 2022).

Requisitos de software são as ações que o software deve executar, possuindo características e condições próprias, de forma a automatizar uma tarefa inerente ao domínio que o software pertence (PRESSMAN & MAXIM, 2019). Assim, alterações de requisitos de software de um SIS significam que modificações são necessárias na estrutura e nos artefatos de software que compõem esse SIS (GRUA et al., 2020).

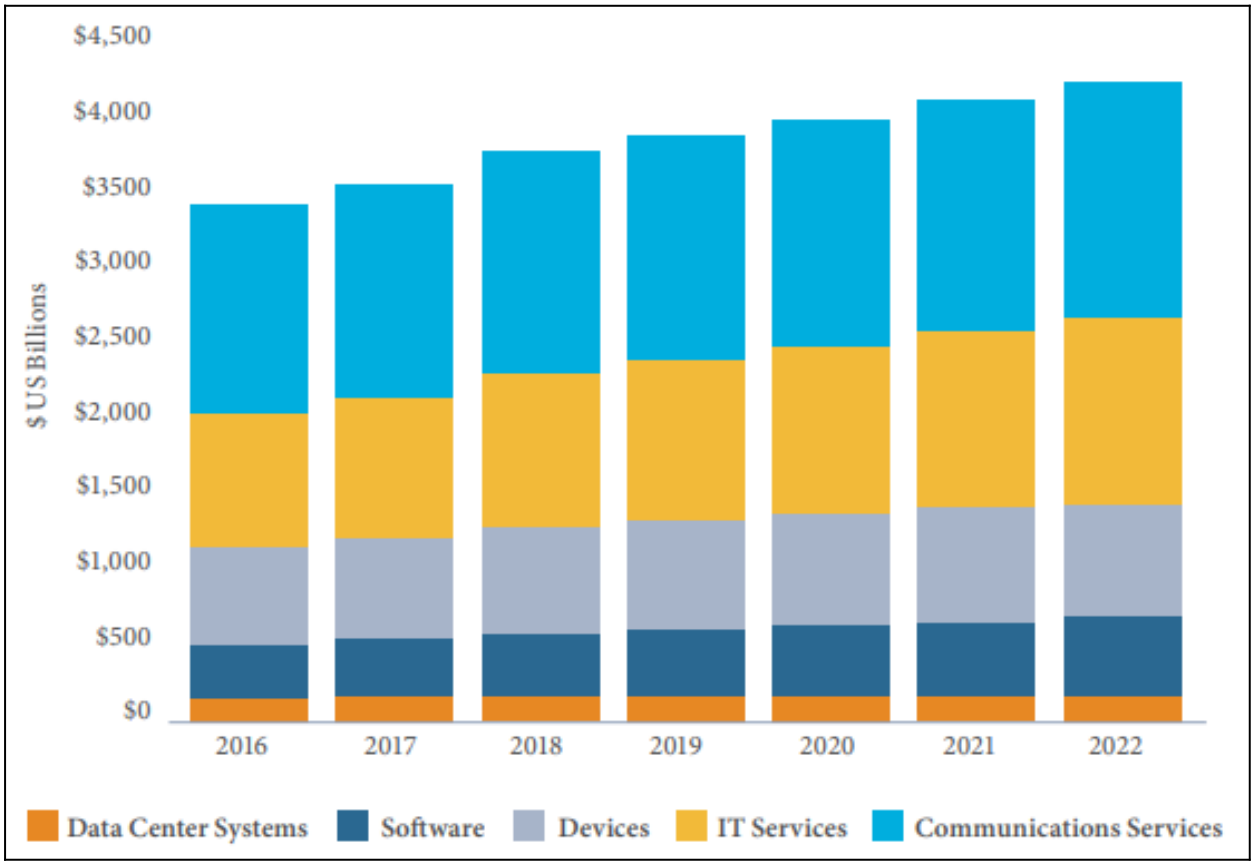
Desta forma, a pandemia do COVID-19 não desafiou apenas a área da saúde, mas também a de Engenharia de Software. Isso ocorreu porque demandou mudanças nos SIS para dar suporte às atividades das unidades de saúde, em escala global (REEVES et al., 2020). Essas mudanças nos SIS têm como propósito deixar as atividades administrativas das unidades de saúde de acordo com os novos protocolos de saúde (OLIVEIRA et al., 2021; LI et al., 2022).

Na área de Engenharia de Software, sabe-se que mudanças nos requisitos de software são eventos comuns no ciclo de vida de um software. Porém, isso é uma das tarefas mais desafiadoras no ciclo de vida de desenvolvimento de software (ANWER et al., 2019) e impacta na conformidade da arquitetura projetada para o respectivo software (GARLAN et al., 2004). Historicamente, sabe-se que acréscimos ou mudanças nos requisitos de software são inevitáveis, demandam retrabalho e representam um custo muito alto no desenvolvimento de software (AFFONSO & NAKAGAWA, 2015).

A Figura 1 mostra uma estimativa de desperdício financeiro na área de TI, onde está indicado que software de baixa qualidade representa um prejuízo crescente ao longo dos anos e que em 2022, esse prejuízo atingirá o valor de 500 bilhões de dólares.

Segundo o *Consortium for IT Software Quality* (CISQ), desenvolvedores de software gastam a maior parte de seu tempo em atividades de retrabalho. Esse consórcio ainda diz que o retrabalho é um dos principais fatores que promove a baixa qualidade no software (KRASNER, 2018). Para manter continuamente a conformidade de um software (ou um sistema de informação) com os requisitos de software, o retrabalho é feito em algumas fases do desenvolvimento de software, por exemplo: na escrita de código, nos testes e na implantação do software no cliente (AFFONSO & NAKAGAWA, 2015). Esse retrabalho também aloca muitos atores no desenvolvimento de software (desenvolvedores de software, testadores, engenheiros de implantação, analistas de suporte).

Figura 1 - Desperdícios na área de TI avaliados em dólares, em todo o mundo.



Fonte: KRASNER (2018).

Esse cenário se torna oneroso em termos de esforço, custo financeiro, além de ser passível de erros devido à injeção involuntária de incertezas pelos atores envolvidos (SAVARGIV et al., 2017).

Para diminuir esse processo de retrabalho, nos sistemas de informação em geral, algumas propostas têm sido apresentadas no domínio de Engenharia de Software. Estudos propõem técnicas capazes de incorporar dinamicamente novas características estruturais ou comportamentais.

Essa incorporação dinâmica permite que o software se adapte às modificações, diminuindo a necessidade de retrabalho na escrita de código, testes ou implantação (GARLAN et al., 2004; MCKINLEY et al., 2004; HORIKOSHI et al., 2012; PEREZ-PALACIN et al., 2014; BUCHANA e SEYMOUR, 2015; SAVARGIV et al., 2017; GRUA et al., 2020). Alguns autores sugerem que SIS podem ser

modificados mais rapidamente e com menos código se utilizarem arquiteturas de software baseadas nas especificações openEHR (ATALAG et al., 2014).

Em paralelo, no contexto de SIS, alguns estudos e ferramentas (encontrados no estado da arte e da prática) trazem contribuições importantes na construção de arquiteturas de software usando o conceito de arquétipos. Algumas dessas contribuições são o uso de computação em nuvem, geração dinâmica de códigos, arquitetura em conformidade com arquétipos e arquitetura baseada em microsserviços (ARAÚJO et al., 2016; GOMES et al., 2018; SILVA et al., 2019; ARAÚJO et al., 2020; CABOLABS, 2022; RIPPLE, 2022).

Contudo, esses trabalhos se mostram suscetíveis ao retrabalho, no que se refere às mudanças nos protocolos de saúde e requisitos de software. Isso implica dizer que, se houver essas mudanças, haverá uma demanda de retrabalho, seja na escrita de novos códigos, nos testes ou na implantação do software.

Essa suscetibilidade ao retrabalho é uma limitação. Diante de uma área cujos protocolos mudam constantemente, pode resultar em grande prejuízo financeiro, conforme ilustrado na Figura 1.

Nesta tese, identificou-se que os SIS que usam arquétipo quando sofre mudanças nos protocolos de saúde, requerem alterações nos arquétipos utilizados na interface gráfica e, consequentemente, demandam alterações em toda arquitetura do SIS para processamento dos dados. Isso significa que o SIS não tem adaptabilidade ao conjunto de arquétipos, conceito explicado a seguir.

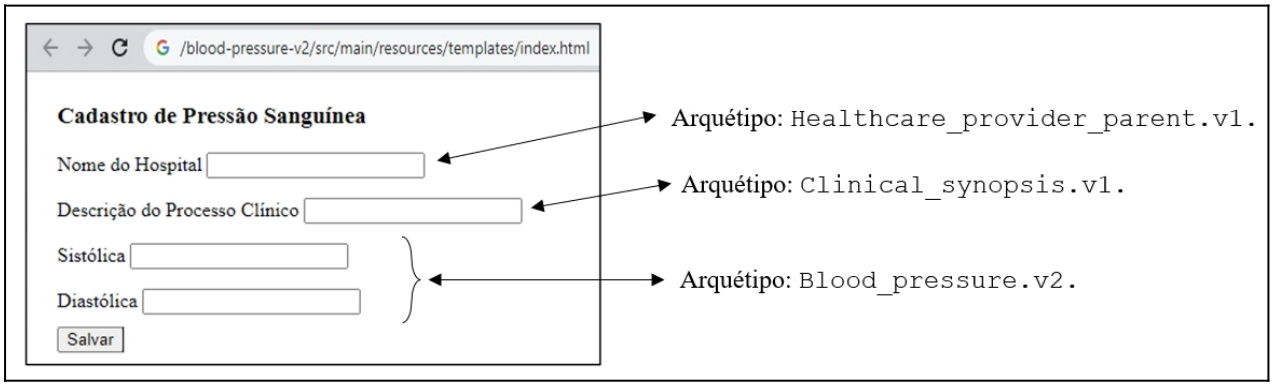
Para melhor entender o conceito de “adaptabilidade ao conjunto de arquétipos” (ou “adaptabilidade aos arquétipos”) utilizado nesta tese. Trata-se da adaptação dos componentes de software da arquitetura, com a finalidade de processar os dados representados por qualquer conjunto de arquétipos utilizados no SIS.

Por exemplo, a Figura 2 mostra um formulário web cujos campos referem-se aos arquétipos `healthcare_provider_parent.v1`, `clinical_synopsis.v1` e `blood_ pressure.v2`. Neste caso, a arquitetura proverá componentes de software que processam os dados representados por esses três arquétipos.

Contudo, em um outro momento, supõe-se que há uma necessidade de remover os campos *Nome do Hospital* e *Descrição do Processo Clínico* desse

formulário e adicionar o campo *Nome do Paciente* (o qual se refere ao arquétipo `person.v0`). Então, haverá a necessidade de adaptar a arquitetura para atender o processamento desse novo conjunto de arquétipos (isto é, `person.v0` e `blood_pressure.v2`).

Figura 2 - Um formulário web construído utilizando três arquétipos.



Fonte: O autor (2022).

Desta forma, a arquitetura proverá componentes de software que são capazes de processar os dados representados por esse novo conjunto de arquétipos. Assim, “adaptabilidade ao conjunto de arquétipos” significa a modificação feita na arquitetura de software para prover componentes de software para processar os dados referentes ao conjunto de arquétipos utilizados no SIS. Este cenário motiva a criação de uma solução que mitigue essa limitação de falta de “adaptabilidade ao conjunto de arquétipo”, trazendo um recurso de adaptabilidade aos SIS que usam arquétipos.

1.3 Problema de Pesquisa e Hipótese

O problema de pesquisa central que norteia o desenvolvimento desta tese é:

- Questão 1: Como construir uma solução com a qual os SIS possam utilizar qualquer conjunto de arquétipos e, quando já implantado e em uso, mediante a necessidade de mudanças desses arquétipos utilizados, a arquitetura consegue se adaptar para processar os dados?

Para o problema de pesquisa aqui investigado, formulou-se a seguinte hipótese:

- Hipótese 1: A utilização de uma arquitetura de software que identifique, em tempo de execução, o conjunto de arquétipos utilizados pelo SIS e, então, adapte um conjunto de componentes de software capaz de processar os dados representados por esse conjunto de arquétipo. Isso permitirá ao SIS se adaptar a qualquer conjunto de arquétipos e processar seus dados.

1.4 Objetivos

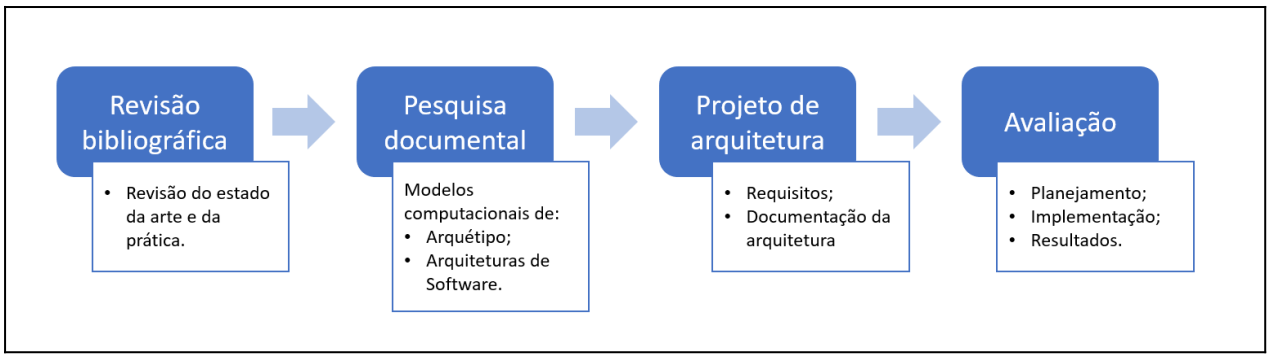
Este trabalho tem como objetivo geral definir uma arquitetura que forneça um arcabouço para o desenvolvimento de software de SIS, com um mecanismo de adaptabilidade ao conjunto de arquétipos utilizado em um SIS. Baseado neste objetivo, listam-se os seguintes objetivos específicos:

- Projetar componentes de uma arquitetura de software, os quais estejam em conformidade com as especificações definidas em arquétipos. Além disso, que esses componentes consigam identificar o conjunto de arquétipos que é utilizado em um SIS, em tempo de execução;
- Projetar uma arquitetura de software que consiga adaptar, em tempo de execução, componentes de software para processamento dos dados representados pelos arquétipos identificados;
- Desenvolver um serviço web (conceito explicado na seção 2.2 do Capítulo 2) com a finalidade de executar a adaptabilidade proposta nesta tese;
- Desenvolver um conjunto de serviços web para processar as operações de criar, ler, atualizar e excluir dados representados por arquétipo, em um banco de dados;
- Desenvolver uma ferramenta com a qual é possível gerar dinamicamente componentes de software para executar a arquitetura projetada nesta tese;
- Realizar uma avaliação para mensurar a adaptabilidade da arquitetura proposta nesta tese, comparando-a com outras arquiteturas de software que usam arquétipos e são geradas por ferramentas no estado da arte e da prática.

1.5 Metodologia

Para elaboração desta tese, utilizou-se uma metodologia composta por quatro etapas. Conforme ilustrado na Figura 3, as etapas são: revisão bibliográfica, pesquisa documental, projeto de arquitetura e avaliação.

Figura 3 - Etapas da metodologia.



Fonte: O autor (2022).

Na etapa da revisão da literatura, procurou-se entender as propostas de arquiteturas de software que usam arquétipos. Para isso, foi realizada uma revisão do estado da arte, identificando as arquiteturas propostas pelas ferramentas que geram SIS usando as especificações dos arquétipos, suas limitações e desafios existentes. Por fim, nesta etapa da metodologia identificou-se o problema abordado nesta tese, ou seja, ausência de arquiteturas adaptáveis ao conjunto de arquétipos usados em um SIS.

A segunda etapa foi utilizada para entender os modelos computacionais fornecidos pela openEHR (i.e., arquétipos) e os modelos de arquitetura de software disponíveis na literatura e, em seguida, extrair os artefatos de software necessários para compor a arquitetura de software proposta neste trabalho.

Em seguida, projetou-se a arquitetura de software adaptativa baseada em arquétipos. Desse modo, foram identificados os artefatos de software que a compõem, os requisitos necessários para o funcionamento dela, e uma documentação dessa arquitetura foi efetuada baseada em linguagens de modelagem em uso na literatura.

Por conseguinte, avaliou-se a arquitetura de software. Em vista disso, elaborou-se um planejamento para o uso da arquitetura de software em dois cenários do mundo real. Implementou-se o cenário projetado e analisou-se o comportamento dos requisitos da arquitetura de software.

1.6 Sumário de Contribuições

A principal contribuição científica deste trabalho é a definição de uma arquitetura de software adaptativa para dar suporte aos SIS que usam arquétipos. Neste projeto, optou-se pelo uso da computação em nuvem (conceito explicado na seção 2.3 do Capítulo 2) para executar os processamentos necessários da arquitetura proposta. Essa escolha da computação em nuvem foi efetuada para usufruir as vantagens de seu uso, com a qual se viabiliza a execução da adaptação proposta por meio de serviços web. Isso torna a arquitetura proposta disponível na Internet, cujo acesso pode ser feito em todo mundo.

Para viabilizar a solução proposta nesta tese, SILVA et al. (2019) propõe uma arquitetura de software descentralizada e que usa arquétipo. Os componentes dessa arquitetura são serviços web disponíveis na Internet (conceito explicado na seção 2.2 do Capítulo 2).

Ademais, outros trabalhos foram publicados, nos quais são abordadas outras características do desenvolvimento e arquitetura de software, por exemplo:

- SILVA et al. (2020a) apresenta a *Decoupled Health Software Architecture* (DHSA), uma arquitetura de software que usa arquétipos e possui atributos de desacoplamento. A métrica *Archetype-based Software Architecture Coupling* (ASAC) foi também desenvolvida nesse trabalho, com a qual é possível medir o nível de acoplamento das arquiteturas de software em saúde que utilizam arquétipos.
- Em SILVA et al. (2020b), a linguagem *Health Sofware Modeling Language* (HSML) é apresentada, com a qual é possível construir aplicações em saúde. Essa linguagem viabiliza o *model-driven development* (MDD), também chamado de desenvolvimento “*low code*”. O software construído através

dessa linguagem é baseado em uma arquitetura de software cujos componentes são reutilizáveis.

SILVA et al. (2022) é outro trabalho que propõe uma arquitetura de software que possui um mecanismo de adaptabilidade aos arquétipos utilizados em um SIS. Esse trabalho também propõe uma ferramenta que gera códigos dinamicamente, chamada AdaptiveHIS. Com essa ferramenta é possível construir artefatos de software usados para compor as arquiteturas de SIS. Esses artefatos se comunicam e executam seus processamentos com base na definição da arquitetura proposta nesta tese.

1.7 Organização do Trabalho

O restante deste trabalho está organizado da seguinte maneira:

- O Capítulo 2 mostra a fundamentação teórica contendo os conceitos necessários ao entendimento deste estudo;
- O Capítulo 3 apresenta os trabalhos correlatos a esta tese, encontrados no estado da arte e da prática. Esses trabalhos mostram como as arquiteturas de software que usam arquétipos têm sido utilizadas na área da saúde e para identificar desafios nesse campo de pesquisa;
- O Capítulo 4 descreve uma abordagem de arquitetura de software que usa arquétipo e tem recursos de adaptabilidade referente aos arquétipos utilizados pelo SIS;
- O Capítulo 5 mostra a avaliação do trabalho, no qual um conjunto de arquiteturas foram avaliadas dentro de dois cenários do mundo real, os quais são considerados oportunos pois ambos representam a necessidade de mudanças devido a pandemia do COVID-19;
- O Capítulo 6 expõe as conclusões desta tese, suas contribuições, limitações e propostas de trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, são apresentados os conceitos essenciais ao entendimento desta tese. Assim, a Seção 2.1 mostra a definição de arquétipo. O conceito de serviço web é descrito na Seção 2.2. A Seção 2.3 apresenta a computação em nuvem. Em seguida, a adaptabilidade de software e suas abordagens são discutidas na Seção 2.4. Por fim, as considerações finais deste capítulo são apresentadas na Seção 2.5.

2.1 Arquétipo

Arquétipo é um padrão internacional da área da saúde utilizado na representação computacional de RES. Esse padrão é especificado pela fundação openEHR, uma organização internacional composta por especialistas da área da saúde (openEHR, 2022). Por meio de arquétipos, computadores conseguem obter, processar e transferir dados. Com essa especificação é possível definir restrições de domínio, modelar os atributos de dados e dar significado aos RES (BEALE, 2002; ISO, 2019).

Nos últimos anos, diversos trabalhos usam arquétipos na indústria de software, em soluções governamentais e em propostas científicas (ULRIKSEN et al., 2017; MUSLIN et al.,2017; GOMES et al., 2018; ARAÚJO et al., 2018; SCHREIER & HAYN, 2018; SCHLIEMANN et al., 2019; SILVA et al., 2019; ARAÚJO et al., 2020; HAK et al., 2020; GOMES et al., 2021; OLIVEIRA et al., 2021; PALIWAL et al., 2022; LI et al., 2022a; LI et al., 2022b; MELLO et al., 2022; MUSZYNSKI et al., 2022; KRYSZYN et al., 2022; POHJONEN, 2022; FRADE et al., 2022; PEDRERA-JIMÉNEZ et al., 2022; JONES et al., 2022; COUTO et al., 2022).

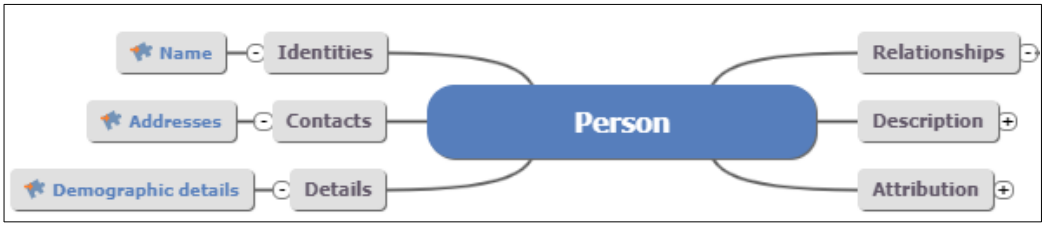
Grandes empresas de tecnologia também têm adotado arquétipos como padrão de dados de saúde em suas soluções, por exemplo: a Microsoft (LADAVA & NORRIS, 2022). Esse cenário confirma a ampla adesão do uso de arquétipos em todo mundo. Outra confirmação da adesão desse padrão internacional é a adoção de arquétipo como um padrão ISO, definido por um órgão internacional de normalização composto por representantes de várias nacionalidades (ISO, 2019).

Na área de informática em saúde, os arquétipos são utilizados em SIS como uma alternativa para remodelar os conceitos clínicos de sistemas legados, implementar o RES em sistemas de banco de dados e, também, definir os requisitos de dados e as terminologias utilizadas em sistemas de informação de saúde (ARAÚJO et al., 2020; openEHR, 2022). Além disso, arquétipos também têm sido utilizados na especificação de modelos de arquitetura de software, conforme apresentado nas seções 3.1 e 3.2 do Capítulo 3.

2.1.1 Atributos

Um arquétipo é composto por atributos, os quais são dados que modelam conceitos clínicos e informações do RES. A Figura 4 mostra o arquétipo `person.v0` e seus atributos, os quais representam o nome de uma pessoa, o documento de identificação, o endereço, o contato (via e-mail, celular, dentre outros), a organização a qual essa pessoa pertence (ou trabalha), detalhes adicionais não mencionados nos atributos anteriores e, por fim, os comentários.

Figura 4 - Mapa mental do arquétipo *Person*.



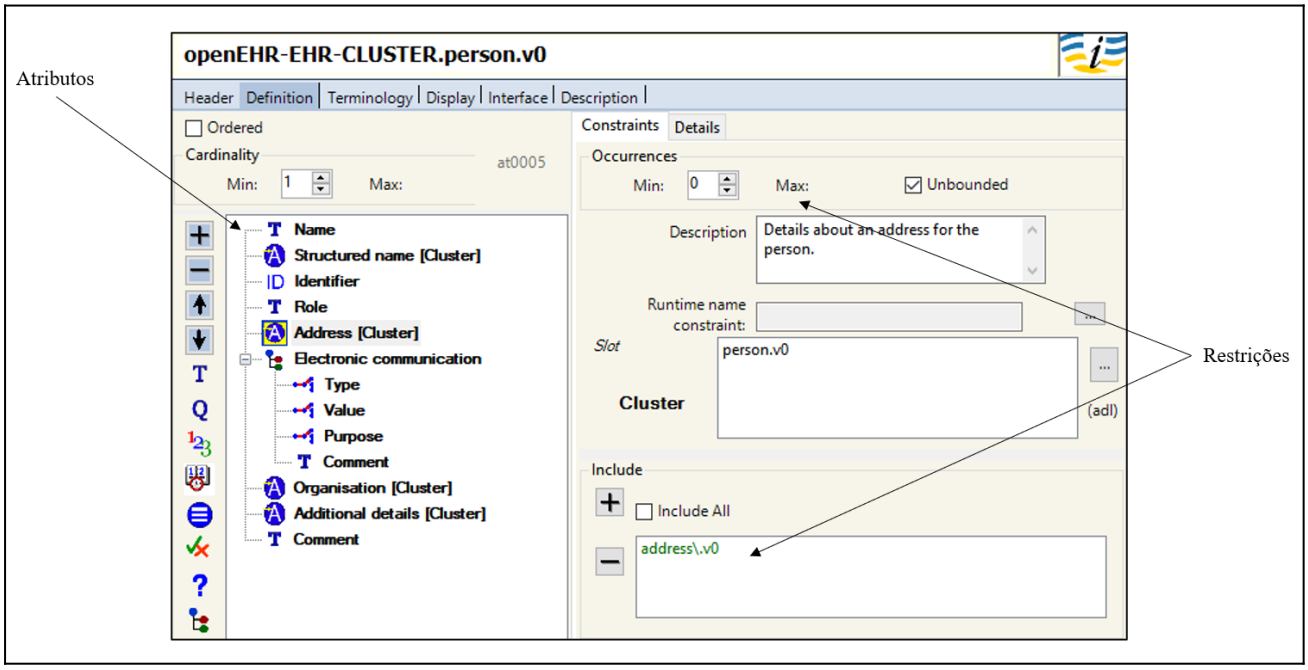
Fonte: *Clinical Knowledge Manager* (CKM) (openEHR, 2022).

Um atributo pode ser composto por outros atributos, conforme mostrado no atributo *Electronic communication*, o qual contém os atributos *Type*, *Value*, *Purpose* e *Comment*. A definição de um atributo pode ser descrita por meio de uma regra (ou restrição de domínio) (openEHR, 2022). Por exemplo, um atributo pode ser representado por um ou mais arquétipos e a sua ocorrência pode ter um escopo mínimo e máximo. A definição dessas regras é efetuada nas restrições de domínio de um arquétipo, conforme explicado a seguir.

2.1.2 Restrições de Domínio (Constraints)

As restrições de domínio (ou *constraints*) foram especificadas para definição de regras nos arquétipos e nos seus atributos. Por exemplo, a Figura 5 apresenta as definições do arquétipo `person.v0` através da ferramenta *Archetype Editor* (openEHR, 2022). Nessa figura, é possível ver todos os atributos do arquétipo e algumas restrições de domínio relacionadas ao atributo *address*. Nessa definição, no campo de ocorrência, é descrito que o atributo *address* pode ter nenhuma ou ilimitadas ocorrências. Ainda, nessas restrições é definido que esse atributo é representado pelo arquétipo `address.v0`.

Figura 5 - Arquétipo `person.v0`, seus atributos e algumas restrições de domínio.



Fonte: *Archetype Editor*.

Para expressar um arquétipo, a fundação openEHR especificou uma linguagem computacional chamada *Archetype Definition Language* (ADL), por meio da qual é possível definir os arquétipos, seus atributos e restrições de domínio. Contudo, outras linguagens computacionais (ou formato de dados) também são usados para esse propósito, por exemplo: *Extensible Markup Language* (XML) e

Javascript Object Notation (JSON) (openEHR, 2022). Exemplos de arquétipos descritos em ADL e XML estão disponíveis no CKM.

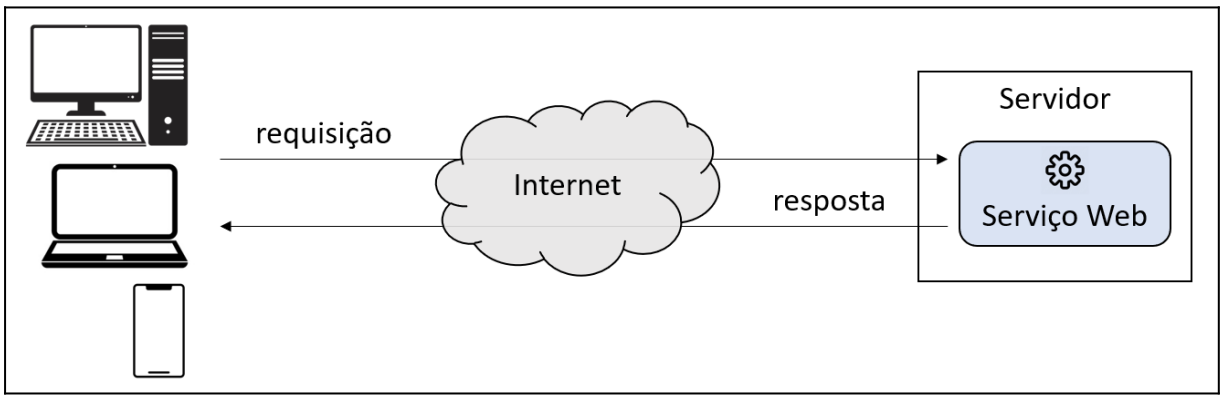
Para promover a padronização de RES entre as instituições de saúde em todo mundo, a fundação openEHR disponibiliza gratuitamente todos os seus arquétipos no CKM (openEHR, 2022). Desta forma, como um dos objetivos da arquitetura proposta nesta tese é o uso de um padrão internacional para representação de seus dados, achou-se pertinente a adoção de arquétipos.

2.2 Serviços WEB

Serviço web (ou somente “serviço”) é um recurso da Engenharia de Software, com o qual sistemas de software em diferentes máquinas interagem entre si, por meio de uma rede de computadores local ou pela Internet (CHRISTENSEN et al., 2000; LEYMANN, 2001). Um serviço executa um processamento específico em um servidor remoto, sob demanda, e não é necessário instalar a aplicação no computador do cliente, bastando apenas acessá-lo por uma rede de computadores (IBM, 2019).

A Figura 6 mostra a comunicação (ou interação) com um serviço. Um software instalado em um dispositivo (computador, *laptop* ou *smartphone*) envia uma requisição em uma rede (local ou na Internet).

Figura 6 - Exemplo de comunicação com um serviço web.



Fonte: O autor (2022).

Ao receber essa requisição, o serviço inicia e executa sua tarefa. Após finalizar sua tarefa, o serviço inicia uma nova comunicação pela rede, na qual envia uma resposta. Essa resposta pode ser de sucesso, erro ou falha (LEMAHIEU, 2001; FENSEL & BUSSLER, 2002).

2.2.1 Protocolo de Comunicação com Serviços

Todo serviço web possui um conjunto de regras de comunicação com o qual é possível enviar requisições para o serviço e receber respostas dele. Esse conjunto de regras é chamado “protocolo de comunicação”. Um exemplo de protocolo é o *hypertext transfer protocol* (HTTP) (FIELDING, 2000). Com esse protocolo é possível definir:

- Métodos de acesso: indica a ação a ser solicitada (por exemplo: *get*, *post*, *etc*);
- Campo de cabeçalho: permite a recuperação de informação sobre um recurso (por exemplo: se há permissão de acesso);
- Mensagem de corpo: permite o envio de dados (por exemplo: os dados de registro de um novo paciente);
- Código de status: número que define o resultado do processamento solicitado (por exemplo: 200, que significa que o processamento ocorreu com sucesso).
- Razão do status: frase que define o resultado do processamento solicitado (por exemplo: “bad request”, que significa que a requisição é inválida).

Um outro protocolo de comunicação utilizado na indústria de software e academia, incluindo a área de informática em saúde, é o *Representational State Transfer* (REST) (FIELDING, 2000). Esse protocolo também tem um conjunto de regras que viabiliza a comunicação entre sistemas de software e serviços distribuídos em uma rede de computadores local ou na Internet.

O protocolo REST utiliza as regras do protocolo HTTP (explicados anteriormente) e adiciona outras regras, por exemplo: o *Hypermedia as the Engine of Application State* (HATEAOS). Esse recurso permite usar *links* de hipermídia para navegar dinamicamente entre serviços obtendo dados desses serviços. Exemplos de links de hipermídia são *uniform resource locator* (URL) e *uniform resource*

identifier (URI), os quais são também utilizados para acessar páginas de Internet, em ferramentas de navegação.

Diversos estudos são encontrados nos quais os SIS e serviços comunicam-se utilizando o protocolo REST (SUNDVAL et al., 2013; YOSHIURA et al., 2016; MUSLIM et al., 2017; ELLOUZEA et al., 2017; GOMES et al., 2018; SCHREIER & HAYN, 2018; SILVA et al., 2019; ARAÚJO et al., 2020; LADAS et al., 2022; BAELE & HOVENGA, 2022).

A comunicação entre serviços oculta os detalhes de implementação utilizados nos serviços. Isso ocorre porque os serviços são construídos para serem consumidos independentemente da plataforma de hardware ou da linguagem de programação utilizada no software que faz a requisição (BEYER et al., 2005). Há vários anos, a literatura reporta o uso de serviço web na área de informática em saúde (CHATTERJEE, 2003; VAZHENIN, 2012; BUCHANA & SEYMOUR, 2015; PALIWAL et al., 2022).

Ainda, corroborando com a adoção dos serviços na área de saúde, a ISO em conjunto com a *International Electrotechnical Commission* (IEC) e o *Institute of Electrical and Electronics Engineers* (IEEE) especificaram esse recurso como um padrão internacional na comunicação entre aplicações de saúde (ISO/IEC/IEEE, 2020).

2.2.2 Arquitetura Cliente-Servidor

A arquitetura Cliente-Servidor trata-se de uma estrutura de aplicação (ou sistema de informação) distribuída, cujos componentes interagem entre si por meio de requisições e respostas em uma rede de computadores. Essa rede de computadores pode ser a Internet.

A arquitetura Cliente-Servidor distribui as funcionalidades de seus componentes entre os fornecedores de um recurso ou serviço de processamento, designados como “camada do servidor”, e os requerentes dos serviços, designados como “camada do cliente” (BASS et al., 2022). Desta forma, a camada de cliente é a parte do software que interage com o usuário da aplicação. A camada do servidor é a parte do software que processa os dados e pode ser composta por vários serviços, conforme descrito a seguir.

2.2.3 Arquiteturas de Software baseada em Serviços

O uso de serviço web, na composição dos softwares em geral, derivou dois modelos de arquiteturas de software bastante conhecidos na área de Engenharia de Software: *service-oriented architecture* (SOA) (FELHI & AKAICHI, 2013) e *microservice architecture* (MSA) (KAKIVAYA et al., 2018).

A semelhança entre ambas é que as duas usam serviços web em sua composição. A diferença é que a segunda arquitetura, especificada mais recentemente, propõe a diminuição do escopo de tarefas de um serviço. Isto é, um serviço tem como responsabilidade a execução de uma tarefa pequena e específica.

Então, comparando com a proposta SOA, a segunda arquitetura é composta por um número maior de pequenos serviços, os quais são denominados na literatura de “microsserviços”. Na área de informática em saúde, a abordagem MSA é utilizada há alguns anos para dar suporte aos SIS (ESPOSITO et al., 2017; IANCULESCU et al., 2019; SILVA et al., 2019; ANDRIKOS et al., 2019; SILVA et al., 2020a; SILVA et al., 2020b; CALDERÓN-GOMES et al., 2020; BETTONI et al., 2021).

No que se refere aos SIS que usam arquétipos como padrão para representação de seus dados, também encontram-se na literatura estudos e ferramentas que propõem o uso de serviços web nesse contexto (PATRIARCA-ALMEIDA et al., 2013; SUNDVAL et al., 2013; SANTOS et al., 2013; CÉSAR, 2013; ATALAG et al., 2014; VIEIRA-MARQUES et al., 2014; YOSHIURA et al., 2016; DEMSKI et al., 2016; MUSLIM et al., 2017; ELLOUZEA et al., 2017; GOMES et al., 2018; SCHREIER & HAYN, 2018; REIS et al., 2018; SILVA et al., 2019; MUKHIYA et al., 2019; ARAÚJO et al., 2020; SILVA et al., 2020a; SILVA et al., 2020b; RIPPLE, 2022; TARENSKEEN et al., 2020; PALIWAL et al., 2022; LADAS et al., 2022; LI et al., 2022a; BAELE & HOVENGA, 2022; LADAVA & NORRIS, 2022; EHRBASE, 2022).

A Internet é um ambiente instável e para mitigar problemas relacionados a essa instabilidade é necessário optar por uma infraestrutura com a qual a comunicação entre os serviços seja menos propensa a falhas e tenha maior probabilidade de estabilidade (JAYBHAYE & ATTAR, 2017).

2.3 Computação em Nuvem

Computação em nuvem é um ambiente computacional composto por um conjunto de servidores (físicos ou virtuais) com capacidade de processamento e gerenciamento de aplicações, plataformas e serviços disponibilizados na internet (BUYYA et al., 2010). Uma característica da computação em nuvem é permitir o acesso sob demanda e a utilização de serviços web para prover funcionalidades e recursos (ARAÚJO et al., 2020).

Desta forma, um conjunto de recursos computacionais configuráveis pode ser rapidamente adquiridos e liberados com o mínimo de esforço de configuração ou interação com o provedor de serviços (RAJESWARI & KALAISELVI, 2017). O termo nuvem é uma metáfora para a internet ou a infraestrutura de comunicação entre os componentes arquiteturais, deixando evidente uma abstração que esconde do usuário, toda a complexidade da infraestrutura e as tecnologias empregadas para oferecer os serviços (JAYBHAYE & ATTAR, 2017; KAUR et al., 2018).

Outra característica importante é que a infraestrutura necessária para o processamento,
a conectividade e o armazenamento dos dados, ficam hospedados em provedores especializados nesse tipo de serviço, por exemplo: SALESFORCE (2022), AMAZON (2022), GOOGLE (2022), MICROSOFT (2022).

Por outro lado, em um ambiente tradicional da área da saúde, para se construir ou gerir um SIS, os profissionais de tecnologia da informação (TI) têm que se preocupar com o desenvolvimento, a instalação, a configuração e a atualização de softwares, além de outros gastos como custos de licenças de softwares.

O padrão arquitetural da computação em nuvem é composto pelos seguintes serviços: software como serviço (i.e., SaaS), plataforma como serviço (i.e., PaaS) e infraestrutura como serviço (i.e., IaaS) (BASS et al., 2022). A definição de cada serviço é mostrada a seguir.

- **Software como Serviço (Software as a Service ou SaaS):** O software é oferecido como um serviço ou sob demanda. Nesse contexto, há o uso de serviços web (CHOU, 2008, VAZHENIN, 2012). O software é executado em

um servidor remoto e não é necessário instalar a aplicação no computador do cliente, bastando apenas acessá-lo pela internet.

- **Plataforma como Serviço:** Característica provida pela nuvem que possibilita aos profissionais de TI, desenvolverem aplicações de software utilizando ferramentas e linguagens de programação disponíveis no ambiente da nuvem;
- **Infraestrutura como Serviço:** Consiste no fornecimento de infraestrutura de processamento, armazenamento, redes, entre outros. Este serviço, assim como os demais, tem seus recursos compartilhados com diversos usuários simultaneamente.

Na literatura, encontram-se trabalhos que utilizam a computação em nuvem como plataforma de suporte aos SIS que usam arquétipos (GOMES et al., 2018, MUKHIYA et al., 2019; ARAÚJO et al., 2020; PALIWAL et al., 2022; LADAVA& NORRIS, 2022; EHRBASE, 2022). Desta forma, o uso da computação nuvem foi adotada na proposta desta tese.

2.4 Adaptabilidade de Software

Conforme apresentado na introdução, o retrabalho no desenvolvimento de software tem um elevado custo porque nesse processo há desperdício de tempo e dinheiro (KRASNER, 2018). Para mitigar problemas relacionados a esse retrabalho manual, alguns estudos propõem o uso da adaptabilidade de software (MCKINLEY et al., 2004; AFFONSO & NAKAGAWA, 2015).

A adaptabilidade (ou adaptação) de software é um conceito bastante trabalhado na área de Engenharia de Software e emergiu, há algumas décadas, da necessidade de um software lidar com ambientes de rápidas mudanças (FAYAD & CLINE, 1996).

A adaptabilidade de software é um processo quase inevitável, devido às comuns mudanças nos requisitos dos clientes, necessidades de desenvolvimento mais rápido de novos sistemas de software ou manutenção de sistemas de software existentes, dentre outros fatores (OREIZY et al., 1999; SUBRAMANIAN & CHUNG, 2001).

A ideia principal da adaptabilidade de software é tornar um software capaz de mudar, ou efetuar ajustes, com a finalidade de resolver problemas diferentes ao longo do tempo. Essa adaptabilidade abrange tudo, desde uma simples substituição de um componente isolado, até reconfigurações que são difundidas e fisicamente distribuídas (MCKINLEY et al., 2004; GARLAN et al., 2004).

Quando um software é construído com recursos de adaptabilidade, a literatura chama esse software de “auto adaptável”, “adaptável”, “autoadaptativo” ou “adaptativo” (ROBERTSON et al., 2001; CHENG et al., 2009; LEMOS et al., 2013, WANG et al., 2022). Desta forma, “sistemas autoadaptativos” são sistemas de software capazes de modificar sua estrutura (ou infraestrutura) ou comportamento, em tempo de execução, para atingir os objetivos de negócio para o qual foi desenvolvido (SALEHIE & TAHVILDARI, 2009).

Circunstâncias imprevisíveis são motivadores para desencadear ações autoadaptativas em um sistema de software. Exemplos de circunstâncias imprevisíveis são: mudanças no ambiente, e alterações nos requisitos de negócio e de software (SUH et al., 2012; YONBAWI & CALINESCU, 2018; CALINESCU et al., 2018; D’ANGELO, 2018; D’ANGELO et al., 2019; SADUOVA & AL-MASRI, 2021; KORRA et al., 2021; RIAZ et al., 2021; GIL et al., 2021; CABRERA & CLARKE, 2022).

Nos últimos anos, diversos estudos propõem soluções e reafirmam a necessidade de construir sistemas de softwares cada vez mais adaptáveis. Esses estudos focam no desenvolvimento de novas técnicas para lidar de forma eficiente com sistemas de software que sejam capazes de evoluir ao longo do tempo e se adaptar às rápidas mudanças de seus requisitos (PEREZ-PALACIN et al., 2014).

Alguns motivos corroboram com essa necessidade de adaptação, por exemplo: os aplicativos de software modernos estão sujeitos a ambientes computacionais cada vez mais complexos, condições operacionais incertas, como dinâmicas na disponibilidade de serviços e variações de objetivos do sistema.

Esse cenário tem demandado sistemas de software com maior capacidade de adaptação, tanto na sua estrutura como no seu comportamento (AFFONSO & NAKAGAWA, 2015; SAVARGIV et al., 2017; SHEVTSOV et al., 2018; YONBAWI & CALINESCU, 2018; LIASKOS et al., 2019; GRUA et al., 2020; ROSA et al., 2020; RUST et al., 2021; JUNIOR et al., 2021; QUIN et al., 2021; CARDELLINI et al., 2022).

Essa demanda por software que tenha recursos de adaptabilidade também é encontrada nos sistemas de software da área da saúde (BUCHANA & SEYMOUR, 2015; MALLYA & KOTHARI, 2018; MUKHIYA et al., 2020; SILVA et al., 2022).

Na literatura, a adaptabilidade de software tem sido abordada em conjunto com outros conceitos da Engenharia de Software, por exemplo: processamento em tempo de execução, serviços web, computação em nuvem e arquitetura de software, conforme apresentado a seguir.

Adaptabilidade em Tempo de Execução: trata-se de um requisito com o qual o software tem a capacidade de alterar sua estrutura ou comportamento durante o tempo em que está em execução, sem exigir interrupções ou reinicialização do sistema. Isto é, a adaptação é efetuada sem afetar a operação contínua do software.

Esse tipo de adaptabilidade é um requisito não funcional importante para os atuais sistemas de software complexos. Nesse sentido, várias abordagens usam modelos de arquitetura de software que permitem o monitoramento, o raciocínio e a adaptação em tempo de execução (OREIZY et al., 2008; TAYLOR et al., 2009, QURESHI & PERINI, 2010; VOGEL & GIESE, 2010; GADIOLI et al., 2015; AFFONSO & NAKAGAWA, 2015; WEYNS & IFTIKHAR, 2016; MUTANO & KOTONYA, 2016; AYALA et al., 2016; LIM et al., 2016; BARBOSA et al., 2017; MOGHADAM et al., 2018; MUTANO & KOTONYA, 2018; ABBAS et al., 2020; LEE et al., 2022; CARDELLINI et al., 2022).

Na área de saúde, alguns estudos propõem o uso dessa abordagem para resolução de problemas, por exemplo: na composição automatizada de serviços web de suporte aos SIS, em sistemas embarcados em dispositivos médicos, robótica médica em estação espacial, dentre outros (STELMACH & FALAS, 2013, REINBACHER et al., 2014; KEMPA et al., 2020).

Adaptabilidade e Serviços Web: A utilização de serviços web para solucionar problemas de adaptabilidade de software também são encontrados na literatura. Esses estudos afirmam que arquitetura de software baseada em serviços web facilitam o desenvolvimento de sistemas computacionais mais adaptáveis.

A vantagem do uso de serviços web é porque são implementados para criar aplicativos de software de maneira fracamente acoplada, e isso facilita a adaptação desses serviços em diferentes sistemas de software (KIRDA, 2001; BALIGAND & MONFORT, 2004; QURESHI & PERINI, 2010; HOG et al., 2012; FELHI & AKAICHI, 2013; SAVARGIV et al., 2017; HOUMANI et al., 2020; PATNAIK & BABU, 2021). Na

área da Saúde, diversos trabalhos adotam os serviços web para compor as arquiteturas de softwares adaptativos (STELMACH & FALAS, 2013; GARAI & PÉNTEK, 2015; YONBAWI & CALINESCU, 2018; GRUA et al., 2020; TARENSKEEN et al., 2020).

Adaptabilidade e Arquitetura de Software: Alguns estudos afirmam que para um software ser adaptável, a sua arquitetura de software deve ter recursos de adaptação. Essa correlação ocorre porque não tem como dissociar um software de sua arquitetura. Isto é, se a arquitetura de software tem mecanismos de adaptabilidade, o software resultante é adaptável (OREIZY et al., 1999; SUBRAMANIAN & CHUNG, 2001; TAYLOR et al., 2009; CHENG et al., 2009; ANDERSON et al., 2013; LEMOS et al., 2013; RIAZ et al., 2021).

Desenvolver arquiteturas de software adaptáveis é crucial para reduzir o custo de desenvolvimento de software. Para alcançar essa adaptabilidade, os engenheiros de software devem especificar as camadas da arquitetura de forma que elas possam ser adicionadas ou removidas. Construir uma arquitetura sem mecanismos de adaptação, pode demandar modificações consideráveis nessa arquitetura à medida que o sistema evolui ou muda devido a requisitos novos ou adicionados.

Essa modificação não planejada, pode demandar um alto custo financeiro e de tempo (FAYAD et al., 2005; VOGEL & GIESE, 2010; WU et al., 2013; CIANCIARUSO et al., 2014; AFFONSO & NAKAGAWA, 2015; CHA et al., 2016; SAVARGIV et al., 2017). Na área de saúde, alguns estudos propõem arquiteturas de software para dar suporte aos SIS adaptativos (FELHI & AKAICHI, 2013; STELMACH & FALAS, 2013; MALLYA & KOTHARI, 2018; GRUA et al., 2020; SELMI et al., 2020; JUNIOR et al., 2021; SILVA et al., 2022).

Adaptabilidade e Computação em Nuvem: Impulsionado pelo uso de serviços como componentes de arquiteturas de software, estudos sugerem o uso da computação em nuvem como ambiente propício para executar arquiteturas de software com mecanismos de adaptabilidade. Algumas vantagens do uso desse ambiente em sistemas autoadaptativos está na sua tolerância a falhas e também no uso de SaaS.

Esse uso de SaaS possibilita uma adaptação desses serviços para ser utilizados por sistemas de softwares diferentes (WU et al., 2013; ZOGHI et al., 2016; CARLINI et al., 2016; VILLARREAL-VASQUEZ, 2017; YONBAWI & CALINESCU, 2018; SABUHI et al., 2020; PEREIRA et al., 2020; KORRA et al., 2021; QUIN et al., 2021; BASSENE & GUEYE, 2022).

Na área de saúde, alguns estudos abordam algum mecanismo de adaptabilidade de software em conjunto com o uso de computação em nuvem (GARAI & PÉNTEK, 2015; ADAMKO et al., 2017; MUKHIYA et al., 2019; GRUA et al., 2020; JUNIOR et al., 2021).

Para a concepção da arquitetura proposta nesta tese, utilizou-se os conceitos de adaptabilidade de software apresentados nesta seção. A seguir, as considerações finais deste capítulo são apresentadas.

2.5 Considerações Finais

Este capítulo listou os conceitos essenciais ao entendimento deste trabalho. Na Seção 2.1 mostrou-se a definição de arquétipo, seus atributos e suas restrições de domínio. Por seguida, a Seção 2.2 abordou serviços web e como efetuar uma comunicação por meio de uma rede de computadores (locais ou na Internet) e utilizando um padrão internacional. Ainda, nessa seção, foi considerada a adoção de serviços web como padrão internacional da ISO, IEC e IEEE, além de sua ampla adoção na área de informática em saúde.

Na Seção 2.3, apresentou-se o significado de computação em nuvem, explicando os serviços disponíveis nesse paradigma (SaaS, PaaS, IaaS) e informando a utilização desses serviços em SIS. Por fim, a Seção 2.4 trouxe as definições de adaptabilidade de software, indicando como essa adaptabilidade vem sendo abordada (em diferentes trabalhos) em conjunto com outros conceitos, por exemplo: processamento em tempo de execução, serviços web, computação em nuvem e, finalmente, com modelos de arquitetura de software.

Esse conceitos são utilizados para construção da arquitetura de software especificada nesta tese. Isto é, arquétipos são usados como modelos de dados, alguns componentes de software são serviços web os quais são instanciados em uma infraestrutura de computação em núvem. Por fim, um mecanismo de adaptabilidade utilizando todos esses conceitos é especificado e implementado.

Esses conceitos também são utilizados em trabalhos encontrados no estado da arte. Porém, há uma lacuna na literatura, a qual se trata de não haver nenhum

trabalho que faça uso desse conceitos em conjunto para execução de adaptabilidade em SIS que usam arquétipos.

Essa lacuna é preenchida pela proposta desta tese, a qual se trata de uma arquitetura de software que utiliza um conjunto de arquétipos, serviços web, computação em nuvem e mecanismo de adaptabilidade em tempo de execução. A especificação dessa arquitetura proposta é descrita no Capítulo 4.

3 TRABALHOS CORRELATOS

Este capítulo apresenta trabalhos e estudos realizados pela comunidade científica, os quais estão relacionados a esta tese por especificarem arquiteturas de software da saúde, as quais usam arquétipos ou possuem recursos de adaptabilidade. Inicialmente, neste capítulo, propostas dos trabalhos correlatos são descritas. Posteriormente, uma avaliação sobre esses trabalhos é apresentada, na qual se identifica algumas lacunas, que foram consideradas e abordadas nesta pesquisa.

Este capítulo está dividido da seguinte forma. A Seção 3.1 descreve alguns estudos que especificam modelos de arquiteturas de software que utilizam arquétipos. Na sequência, a Seção 3.2 mostra ferramentas que usam arquétipos para construir SIS e suas respectivas arquiteturas de software. Na Seção 3.3, arquiteturas de software que utilizam recursos de adaptabilidade e dão suporte à área da saúde são listadas. Posteriormente, a Seção 3.4 faz um comparativo entre todos esses trabalhos. Finalmente, as considerações finais desse capítulo são descritas na Seção 3.5.

3.1 Arquiteturas de Software baseadas em Arquétipos

Diversos trabalhos são encontrados no estado da arte os quais propõem arquiteturas de software para o domínio da saúde, conforme apresentado no Capítulo 2. Nesta seção, considerou-se os trabalhos que apresentam arquiteturas de software que usam arquétipos como padrão de representação de seus dados.

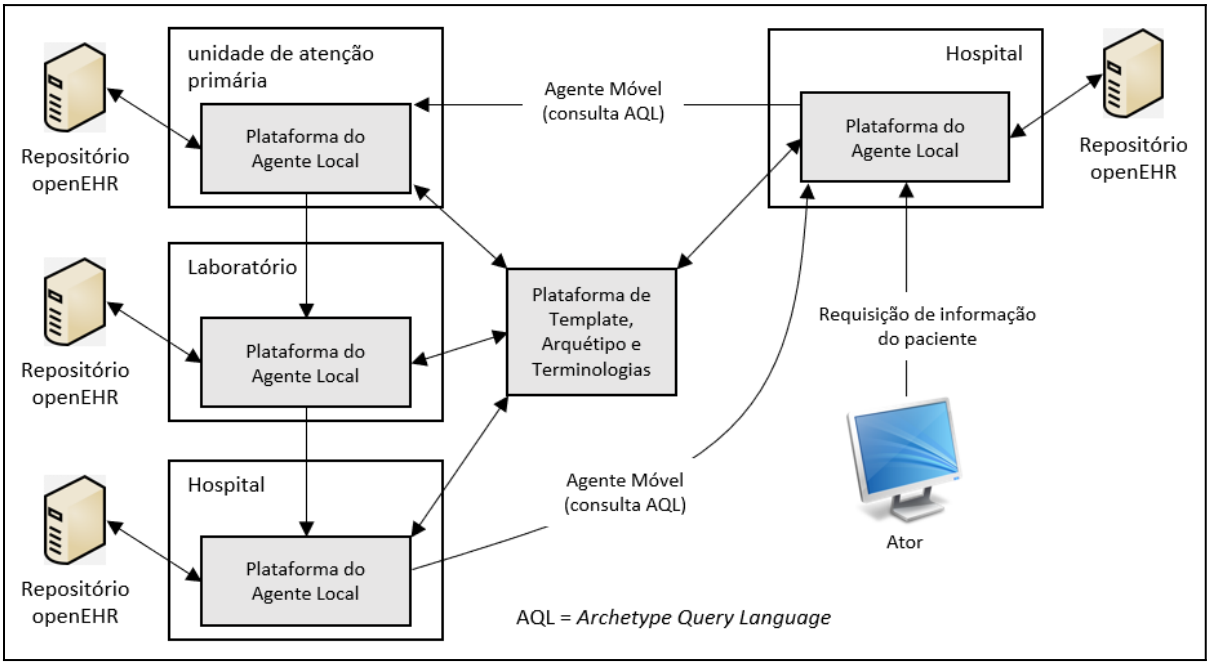
Antes de listar os trabalhos correlatos, é importante entender o conceito de API (*Application Programming Interface*) porque há o uso desse conceito nesses trabalhos. API é uma interface que viabiliza a interação de diferentes softwares ou serviços e, por meio da qual, dados podem ser enviados para processamento utilizando uma rede de computadores local ou pela Internet (BETTER, 2017; REIS et al, 2018; CABOLABS, 2022; EHRBASE, 2022).

SILVA-FERREIRA et al. (2012) definem uma arquitetura de software que permite integrar especificações openEHR e o padrão HL7 para processar consultas em repositórios de dados de pacientes em instituições externas. Essa arquitetura é executada da seguinte forma: um sistema multiagente busca e recupera dados de pacientes localizados em várias instituições de saúdes. A Figura 7 mostra a arquitetura com um sistema multiagente, a qual é dividida em agentes locais e móveis.

Esses agentes locais representam as plataformas computacionais fisicamente localizadas em diferentes unidades de saúde, por exemplo: uma unidade de atenção primária, um laboratório, um hospital. Os agentes móveis são processos que são executados sobre os agentes locais em busca de dados de pacientes. Um ator pode ser um usuário de um SIS ou um processo computacional, o qual requisita um conjunto de dados (informação) de um paciente.

Quando essa requisição é iniciada, o sistema multiagente cria e atribui um agente local ao ator. A Plataforma de *Template* atribui a essa requisição um *template* openEHR (openEHR, 2022), o qual se trata de um mecanismo de agrupamento de arquétipos que representam os dados requisitados.

Figura 7 - Visão geral da arquitetura proposta em FERREIRA et al. (2012).



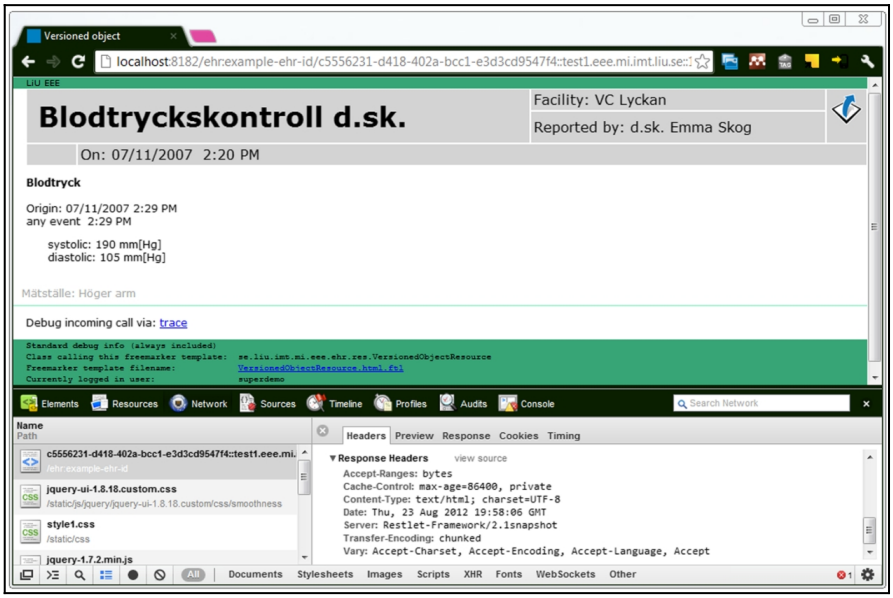
Fonte: Adaptado de FERREIRA et al. (2012).

As consultas geradas serão atribuídas, pelo agente local, a um conjunto de agentes de consulta que serão criados neste momento para as tarefas de busca. Os agentes móveis de consulta pesquisarão em múltiplos agentes locais, trafegando entre plataformas computacionais das várias instituições hospitalares. Os agentes móveis consultam dados de um agente local por meio de consultas escritas em *archetype query language* (AQL).

SUNDVAL et al. (2013) propõem uma arquitetura orientada a serviço cuja comunicação entre esses serviços é efetuada usando o protocolo REST. Nesse trabalho, a arquitetura é composta por subcomponentes funcionais baseados na especificação openEHR. Essa arquitetura permite o armazenamento, a recuperação e o versionamento de dados baseados em arquétipo. A especificação dos subcomponentes foi guiada pela especificação openEHR e pela separação de "camadas": GUI, processamento e armazenamento.

Esses subcomponentes se comunicam em um ambiente distribuído em uma rede de computadores (local ou Internet). Essa arquitetura é utilizada na ferramenta *Linköping University Educational* EHR *Environment* (LIU EEE). Conforme ilustrado na metade superior da Figura 8, essa ferramenta mostra a interface para desenvolvedores e o arquétipo "*blodtryck*" (versão e composição).

Figura 8 - Ferramenta LIU EEE.



Fonte: Extraído de SUNDVALL et al. (2013).

A metade inferior dessa figura mostra a visualização de rede com uma ferramenta de depuração presente nos navegadores (acessados clicando com o botão direito do mouse e selecionando “inspecionar”). Desta forma, é possível visualizar dados detalhados, por exemplo: controle de cache, tipo de dados trafegados, tempo de execução, e qual ferramenta foi utilizada no servidor.

VIEIRA-MARQUES et al. (2014) propõem a arquitetura *Secure Agent based Health Information systems Brokering* (SAHIB), a qual integra dados de saúde de um paciente, armazenados em sistemas processados em locais distintos. Essa arquitetura utiliza um sistema multiagente. Nessa arquitetura, diferentes instituições de saúde têm os seus respectivos agentes locais. A função de um agente local é armazenar dados clínicos de sua respectiva instituição e, também, deixar esses dados disponíveis para consultas de outros agentes de outras instituições.

Com a arquitetura SAHIB é possível a coleta os dados clínicos de um conjunto de instituições de saúde, através da comunicação desses agentes locais. Um SIS que usa a arquitetura SAHIB executa três processos principais: agendamento de eventos, pesquisa de dados, recuperação de dados, os quais são descritos a seguir. *Agendamento de eventos* é registro de novos eventos, por exemplo: o armazenamento de dados clínicos de um paciente em uma instituição de saúde.

Neste caso, esses dados ficam armazenados no agente local dessa instituição. *Pesquisa de dados* é quando ocorre uma busca de um conjunto de dados, por exemplo: a busca de dados clínicos de um paciente. Nessa busca, a arquitetura SAHIB consulta esses dados em todos os agentes locais das diferentes instituições de saúde. *Recuperação de dados* é o fornecimento dos dados localizados ao usuário que fez essa busca.

REIS et al. (2018) propõem uma arquitetura de software com a qual é possível integrar dados de saúde baseados em arquétipos e armazenados em fontes heterogêneas (ou seja, em diferentes formatos). Conforme mostrado na Figura 9, essa arquitetura é composta de oito componentes: *Serviço de Agrupamento*, *Camada de Virtualização de Dados, Agente Federado*, *Serviço Mapeador, Serviço de Consultqa*, *Serviço de Autenticação*, *Analisador de Consulta* e

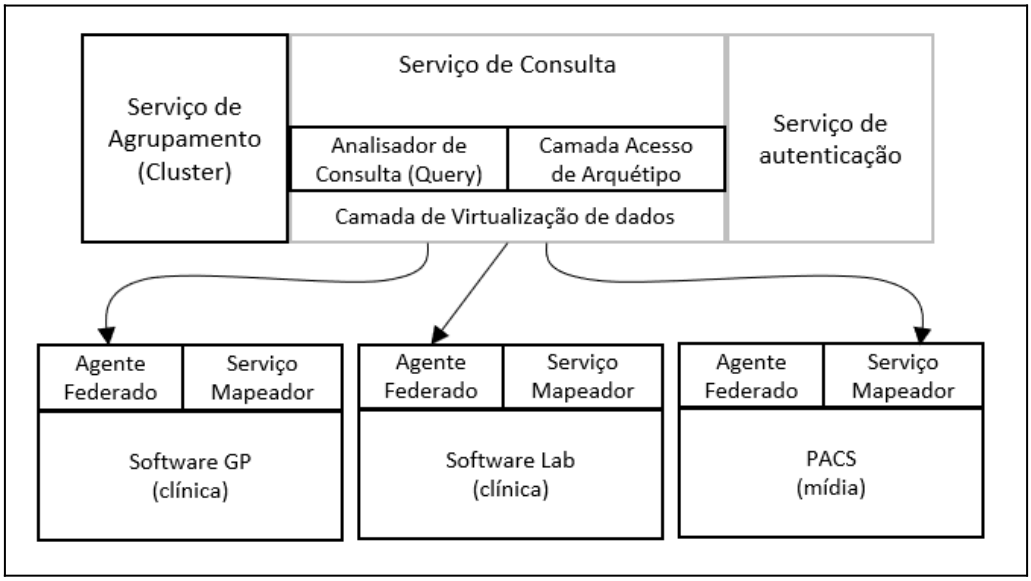
Camada Acesso de Arquétipo. Os quatro primeiros componentes executam o processo de integração de dados e não interagem diretamente com o usuário. Esses componentes são explicados a seguir.

Serviço de Agrupamento efetua processos de conexão, descoberta e transmissão de consultas. A camada *Virtualização de Dados* é um componente que representa os sistemas legados e suas fontes de dados, ou seja, esses sistemas legados possuem os dados clínicos. Esse componente funciona como um agregador de dados, dando a ideia de uma fonte única para todos os outros sistemas que interagem com ele.

O *Agente Federado* é um componente que está diretamente conectado ao sistema legado que contém os dados. Esse agente pode se conectar ao banco de dados do sistema ou se comunicar por meio de uma API e converte consultas AQL (executadas sobre arquétipos) em consultas escritas na linguagem de consulta que o sistema legado usa. As regras de conversão são executadas a partir do *Serviço Mapeador*.

Os outros quatro componentes são responsáveis pela construção da consulta, recuperação de dados e autenticação. *Serviço de Consulta* é responsável por construir e executar as consultas. Esse serviço possui uma interface web para construção de consultas onde o usuário local pode criar consultas personalizadas.

Figura 9 - Visão geral da arquitetura proposta em REIS et al. (2018).

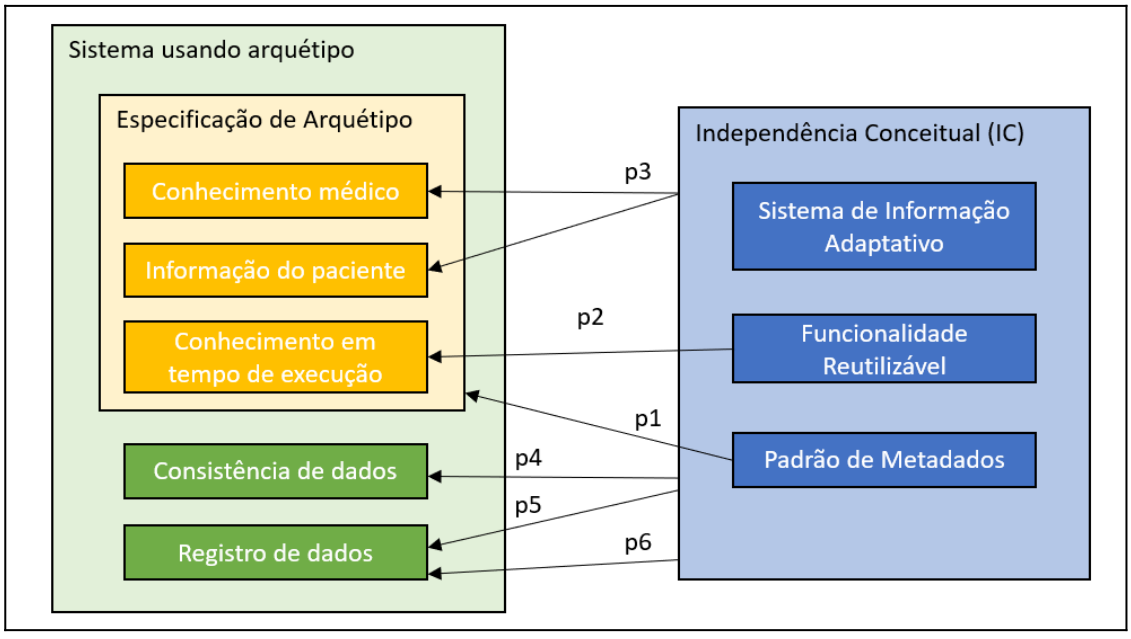


Fonte: Adaptado de REIS et al. (2018).

Serviço de Autenticação executa um fluxo de processos para autenticação e autorização de usuários locais e remotos. Os últimos dois componentes *Analisador de Consulta* e a *Camada Acesso de Arquétipos* são responsáveis por analisar as consultas que o sistema recebe, validando-as através de um conjunto de regras para verificar se o usuário que está solicitando os dados pode de fato acessá-los.

TARENSKEEN et al. (2020) apresenta uma abordagem conceitual sobre como o uso de arquétipos promove adaptabilidade em arquiteturas de software. Esse estudo mostra que o uso de arquétipos proporciona a adesão de conceitos que são utilizados em sistemas de software adaptáveis, tal como o conceito de independência conceitual. Conforme apresentado na Figura 10, a *Independência Conceitual* (IC) propõe três camadas e alguns princípios. As camadas são *Sistemas de Informação Adaptativo*, *Funcionalidade Reutilizável* e *Padrão de Metadados*. Os princípios são: recursos que podem ser reusados (p1), categorias de dados conhecidas (p2), aplicação de esquemas de dados (p3), identificação de entidades de dados (p4), aplicação de rótulos sobre os dados (p5).

Figura 10 - Estudos sobre o uso de arquétipo e a promoção da independência conceitual.



Fonte: Adaptado de TARENSKEEN et al. (2020).

Essa figura ainda ilustra alguns elementos comuns aos sistemas que usam arquétipos: conhecimento médico (o arquétipo utilizado), informação do paciente (os dados clínicos que esses arquétipos representam), conhecimento em tempo de execução (o agrupamento de arquétipos realizado pelos *templates* openEHR), consistência e registro de dados são funcionalidades do SIS. As camadas e princípios da IC são base para uma arquitetura de software flexível e adaptável. Nesse estudo, verificou-se que o uso de arquétipo pode ser visto como um meio de alcançar a IC. Os arquétipos são desacoplados da lógica de aplicação, e isso é uma característica geral da IC. Um SIS pode usar diferentes arquétipos (p1).

Os arquétipos podem ser agrupados de forma que partes específicas da interface do usuário possam ser apresentadas (p2). Arquétipos em diferentes versões podem coexistir em uma aplicação (p3). A verificação da consistência dos dados de um SIS que usa arquétipo representa p4. O registro dos dados com os diferentes arquétipos representa p5 e p6.

PALIWAL et al. (2022) apresentam uma arquitetura de software baseada nos padrões openEHR e HL7. Essa arquitetura de software tem como finalidade a recuperação de dados armazenados em sistemas administrativos. Para validação dessa arquitetura, dados do *Medical Information Mart for Intensive Care* III (MIMIC III) foram utilizados. MIMIC é um banco de dados disponível gratuitamente, o qual contém dados de quarenta mil pacientes hospitalizados (com câncer) em unidades de cuidados intensivos durante 2001 a 2012 no centro médico *Beth Israel Deaconess*.

Uma arquitetura com seis camadas foi especificada nesse trabalho: *cloud*, *persistence*, *backend services*, *virtual client*, *application logic*, *presentation logic*, as quais são apresentadas a seguir. A camada *Cloud* fornece acessibilidade e compartilhamento de dados para clientes remotos, por meio de computação em nuvem. *Persistence* permite o armazenamento e a recuperação de dados. *Backend service* implementa diferentes serviços web para extração de dados, execução de regras de segurança e privacidade, serviços demográficos, arquétipos, terminologia, localização de registros.

Virtual client atua como um middleware entre a camada de aplicativo e de serviço. Essa camada ajuda *Application logic* a acessar um conjunto pertinente de

serviços. *Application logic* contém a regras de negócios específicas para uma aplicação de usuário ou outro serviço. A camada de apresentação inclui a GUI da aplicação.

A seguir, alguns trabalhos que propõem ferramentas de construção de SIS a partir de arquétipos são apresentados.

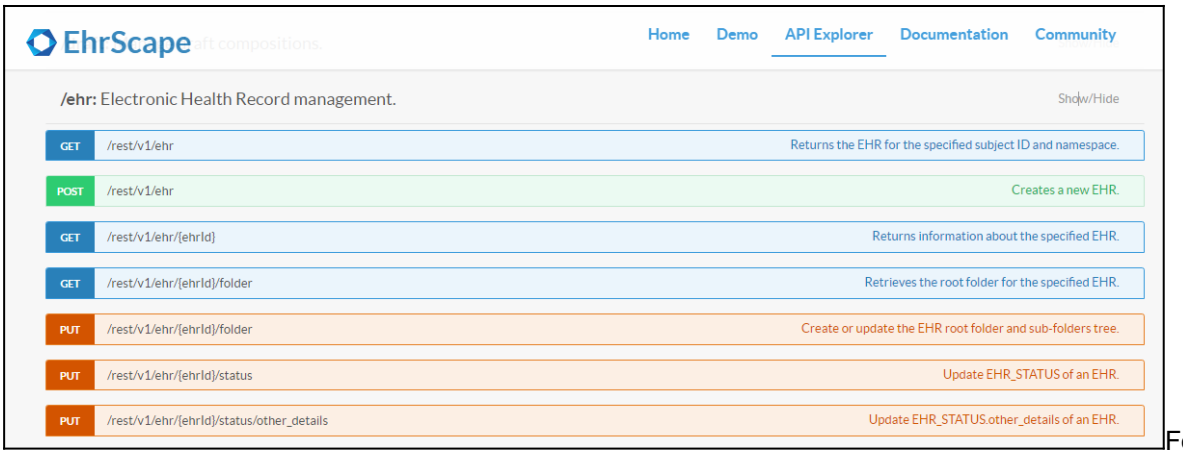
3.2 Ferramentas para Desenvolvimento de SIS que Usam Arquétipos

Nesta seção, ferramentas projetadas para construir SIS que usam arquétipos são descritas. Com essas ferramentas, é possível gerar os códigos e componentes de arquiteturas de software, os quais executam as funcionalidades dos SIS citados. Esses trabalhos são apresentados a seguir.

EhrScape (BETTER, 2017) é uma ferramenta baseada na especificação openEHR que fornece um conjunto de recursos para construir serviços web para área da saúde. Dentre suas principais características, destaca-se um conjunto de *Application Programming Interfaces* (API) que permitem consultar terminologias, integrar dados legados do RES e consultar sistemas de apoio à decisão clínica.

Como mostra a Figura 11, por meio do protocolo REST, requisições GET e POST das API disponibilizadas pela ferramenta EhrScape, é possível recuperar os elementos de um arquétipo em formato JSON e criar soluções que integrem aplicações legadas com as especificações openEHR.

Figura 11 - Ferramenta EhrScape.



Fonte: Extraída de BETTER (2017).

MARCIA (*Manejo de Registro Clínico Aplicado*) (GOMES et al., 2018) trata-se de uma solução para o desenvolvimento de SIS, utilizando o framework EHRServer e padrões OpenEHR. Essa abordagem registra e acompanha o atendimento clínico do paciente com chikungunya na UBS, no intuito de auxiliar os profissionais da atenção básica na melhoria do atendimento.

MARCIA utiliza a ferramenta EhrServer para processamento de dados de arquétipos e posterior armazenamento. Para o desenvolvimento da solução proposta, as seguintes camadas foram especificadas: requisitos, arquétipos, composição, exportação, upload, implementação, geração de formulário. Essas camadas são detalhadas a seguir.

A camada de requisitos representa a fase inicial, onde são definidos os requisitos do sistema. A camada de arquétipos contém os arquétipos. A camada de composição agrupa os arquétipos em uma estrutura única, dando origem aos *templates*. A camada de exportação exporta os *templates* criados em dois formatos: *operational template* (OPT) ou definição de esquema XML (XSD).

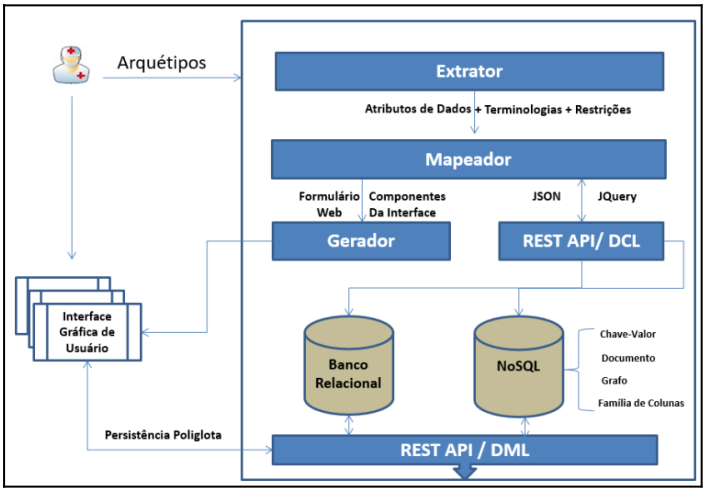
Em seguida, a camada de *upload* carrega o *template* na ferramenta EhrServer. A camada de implementação processa a leitura e a implementação de um *template*. Após a implantação da aplicação, a camada de geração do formulário constrói o formulário HTML para interação com o usuário final, disponibilizado por meio de um navegador.

Template4EHR (ARAÚJO et al., 2020) é uma ferramenta que viabiliza a construção de esquemas de dados para o armazenamento de RES em bancos de dados heterogêneos, isto é: relacional, do tipo chave-valor, orientado a documentos, colunar ou baseado em grafos. A Figura 12 mostra uma visão conceitual do funcionamento de Template4EHR.

Essa ferramenta também viabiliza a geração de GUI a partir dos atributos de dados e das restrições de domínio definidas em um conjunto de arquétipos selecionados pelo usuário. A partir da leitura de um arquétipo informado por um profissional de saúde, o framework realiza as tarefas de criação dos esquemas de dados e geração das interfaces gráficas de usuário. Para isso, o componente *Extrator* retira do arquétipo informado, os atributos que definem o RES, as

terminologias e os vocabulários da área de saúde que dão significado semântico aos dados clínicos, além das restrições especificadas sobre os atributos.

Figura 12 - Visão geral da arquitetura proposta na Template4EHR.



Fonte: Extraída de ARAÚJO et al. (2020).

Após a extração, armazena-se os elementos do arquétipo em uma estrutura de dados em formato *JavaScript Object Notation* (JSON) para geração dos artefatos de software pelo componente *Mapeador*.

De posse dos elementos do arquétipo, o componente *Mapeador* executa as seguintes regras de domínio: i) mapeia os atributos de dados em componentes de entrada de dados na interface gráfica (e.g., caixa de texto, lista de valores suspensa); ii) utiliza as restrições extraídas dos arquétipos como mecanismo de validação de entrada de dados na interface gráfica (e.g., intervalo de valores, restrição de tipo de dado); iii) disponibiliza as terminologias extraídas dos arquétipos na interface gráfica para dar significado semântico aos seus respectivos componentes.

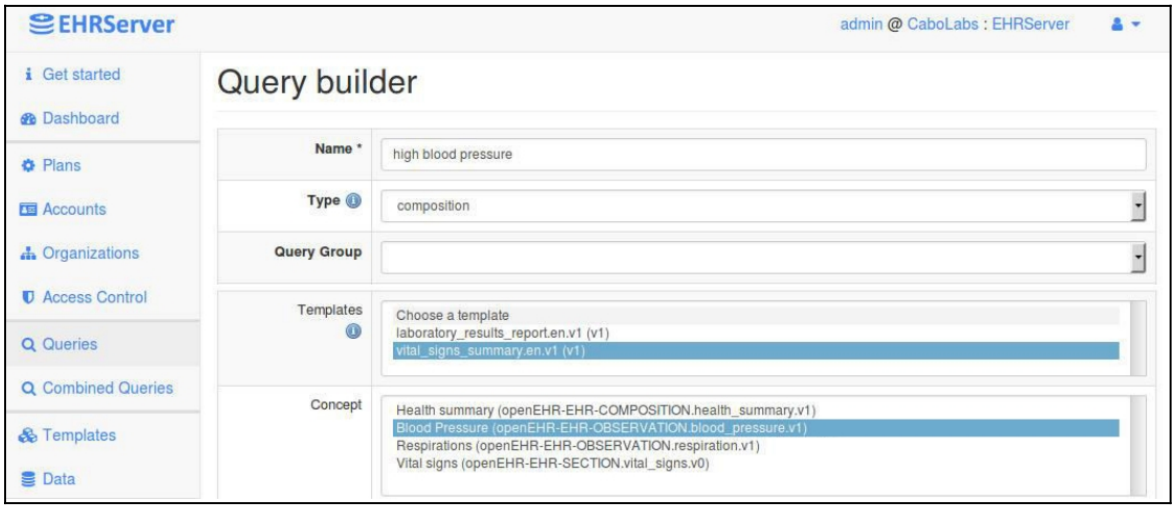
Depois que o componente *Extrator* extrai os atributos, as terminologias e as restrições, a REST API/DCL gera esquemas de dados relacionais e NoSQL. Template4EHR dispõe de uma funcionalidade que permite configurar o tipo do esquema de dado a ser gerado no momento da leitura dos arquétipos.

CloudEHRServer (CABOLABS, 2022) é uma ferramenta que usa arquétipos e é voltada para o desenvolvimento de SIS que compartilham e gerenciam dados clínicos. Suas principais características são: disponibilização de serviços web para

compartilhamento de dados de saúde entre SIS que usam arquétipos e armazenamento desses dados em uma solução proprietária.

Figura 13 mostra o ambiente de CloudEHRServer no qual é possível construir uma consulta sobre um arquétipo e seus respectivos *templates*. Esta ferramenta permite criar uma definição de documento clínico, carregar essa definição para o EHRServer, criar consultas de dados com base em definições de documentos, integrar a API do EHRServer a outros SIS, consultar e exibir dados.

Figura 13 - Ferramenta CloudEHRServer.



Fonte: Extraída de CABOLABS (2022).

EHRBASE (2022) é uma plataforma que fornece um conjunto de serviços web com os quais é possível armazenar um conjunto de dados representados por arquétipos. O código fonte dessa ferramenta está disponível na Internet gratuitamente (*open source*).

Suas principais características são: utilização de arquétipos com padrão de representação dos dados de saúde, utilização de um conjunto de API para disponibilizar as funcionalidades de consulta e armazenamento de dados, utilização de computação em nuvem, integração com outros padrões de saúde como *Fast Healthcare Interoperability Resources* (FHIR) (HL7 INTERNATIONAL, 2022).

EtherCIS (RIPPLE, 2022) é uma plataforma *open source* compatível com o padrão openEHR. Essa plataforma foi projetada para permitir interações simples com SIS usando a API REST e persistir dados clínicos em um banco de dados na Internet.

A seguir, descreve-se um outro conjunto de arquiteturas de software que dão suporte aos sistemas de software da área de saúde e possuem recursos de adaptabilidade. Esse recurso é também utilizado na arquitetura proposta nesta tese.

3.3 Arquiteturas de Software Adaptativas para Área de Saúde

Zhang (2017) propõe uma arquitetura adaptativa baseada em conceitos de aprendizagem de máquina, sistemas multiagentes e *big data*. Esta arquitetura promove a interação contínua entre agente e ambiente, e viabiliza a adaptabilidade de sistemas que usam essa arquitetura por meio do aprendizado de máquina.

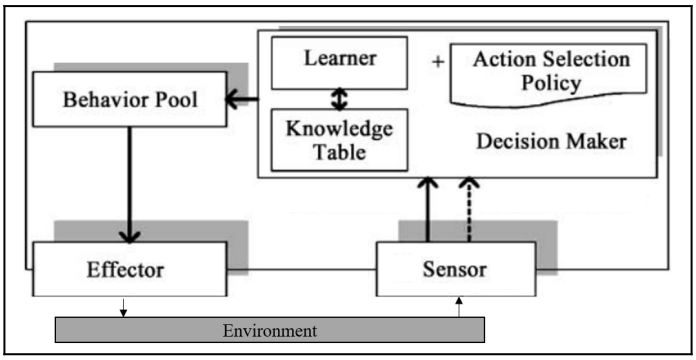
Conforme ilustrado na Figura 14, essa arquitetura possui cinco módulos (*environment*, *sensor*, *decision maker*, *behavior pool* e *effector*) e três repositórios (*knowledge table*, *action selection policy* e *behavior pool)*, os quais são explicados a seguir.

O módulo *environment* é o ambiente no qual um SIS está sendo executado. O módulo *sensor* é responsável pela percepção das mudanças ocorridas em *environment*. Essa mudança é denominada “novo comportamento”. Ao detectar uma mudança (ou novo comportamento) no módulo *environment*, S*ensor* obtém os dados relativos a esse novo comportamento e os envia ao módulo *decision maker*. No *decision maker*, o subcomponente *Learner* executa alguns processamentos:

- Recebe os dados enviados por *sensor*;
- Compara os dados do novo comportamento (recebido de *sensor*) com dados armazenados no repositório chamado *knowledge table*. Nesse repositório está o histórico de dados de mudanças (ou comportamentos) ocorridas anteriormente;
- Baseado nas regras contidas no repositório *action selection policy*, *Leaner* atualiza os dados contidos no *knowledge table*. Essa atualização significa a adição dos dados referentes ao novo comportamento detectado pelo *sensor*.

Em seguida, *decision maker* persiste esses dados (referentes ao novo comportamento) no repositóro *behavior pool*. Os dados desse repositório (*behavior pool*) são utilizados pelo módulo *effector*, o qual executa um processamento de ajuste (ou adaptação) no módulo *environment*.

Figura 14 - Sistema adaptativo baseado em aprendizagem por reforço.



Fonte: Extraído de Zhang (2017).

MALLYA & KOTHARI (2018) propõem uma abordagem de adaptabilidade aplicada ao sistema de monitoramento da saúde da mulher. Neste trabalho, propõe-se o desenvolvimento de um sistema autônomo e multiagente para o monitoramento de feto. O software é chamado *Multi-Agent Autonomic Fetus Monitoring System* (MAA-FMS) e utiliza dispositivos vestíveis (por exemplo: relógios inteligentes), os quais são usados para obter e monitorar os dados do paciente.

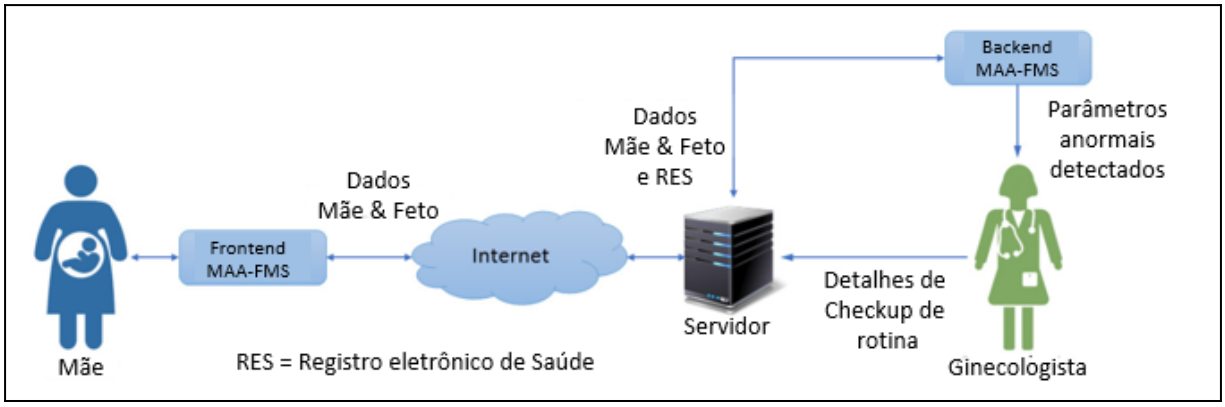
Como mostrado na Figura 15, o *frontend* MAA-FMS é um dispositivo vestível que desempenha o papel de monitoramento e diagnóstico. O dispositivo vestível está conectado com o aplicativo móvel do sistema de monitoramento do feto. Este aplicativo será instalado em celulares de gestantes, ginecologistas e caseiros para comunicação de emergência. O sistema de monitoramento do feto monitorará os batimentos cardíacos do feto, movimentos do feto e parâmetros do corpo materno, como nível de fluido embrionário e contrações uterinas.

Esse sistema pode gerar alarmes de ingestão de água e alimentos para a mãe e alertas para o ginecologista se forem observados alguns parâmetros corporais anormais. A adaptabilidade desse sistema está relacionada à configuração de dados de diferentes pacientes. Então, se uma nova paciente utiliza o relógio inteligente, uma nova configuração é carregada no software MAA-FMS, para atender as necessidades individuais da nova paciente. Desta forma, esse software se adapta a alguns requisitos ou cenários incertos.

Alguns dos cenários incertos são: (i) Ambiente imprevisível, se for um ambiente completamente novo (nova paciente utilizando o sistema), atualize a base de conhecimento do paciente; (ii) Frequência cardíaca do feto menor que 110 ou

maior que 220, conforme base de conhecimento, geram-se alertas; (iii) Mudança nos requisitos, em caso de nova paciente, altera-se a base de conhecimento da paciente e, se necessário, uma mensagem solicitando um dispositivo vestível adequado àquela paciente é enviada; (iv) Contrações maiores que 70%, conforme base de conhecimento, geram-se alertas.

Figura 15 - Ferramenta MAA-FMS.



Fonte: Adaptado de MALLYA & KOTHARI (2018).

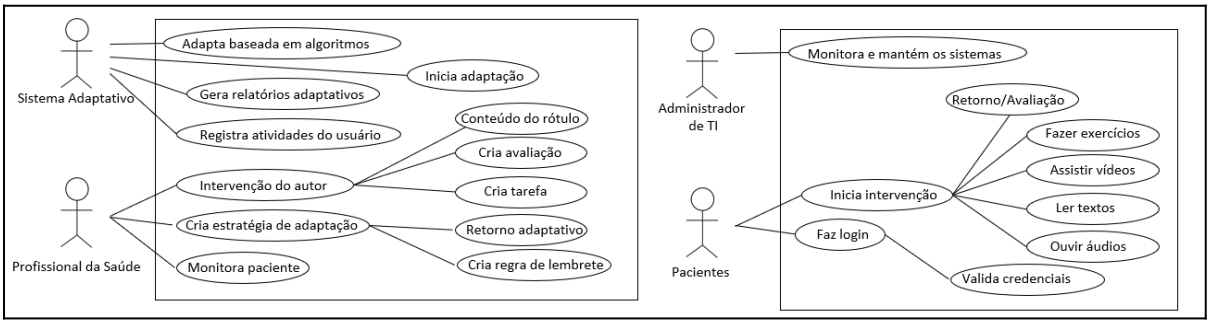
MUKHIYA et al. (2020) apresentam a adição de recursos de adaptabilidade ao sistema *internet-delivered psychological treatments* (IDPT). Esse SIS foi projetado para executar na Internet para dar suporte no tratamento de doenças mentais. Esse sistema adota uma adaptabilidade baseada nos dados obtidos dos pacientes, através do uso da PBA (*person-based approach*). A Figura 16 ilustra os diferentes tipos de atores e ações envolvidos no uso dessa proposta.

Os profissionais da saúde (a esquerda) estão ativamente envolvidos na tarefa de desencadear e criar adaptação. Os pacientes (a direita) usam o sistema adaptativo e, finalmente, o administrador de TI (a direita) monitora e mantém o sistema adaptativo para garantir o funcionamento adequado.

A arquitetura IDPT adaptativa foi especificada com: (i) Projeto arquitetônico, consiste na estrutura geral do sistema, seus componentes e respectivos relacionamentos. (ii) Projeto de banco de dados, inclui a criação das estruturas de dados do sistema e seus relacionamentos, e sua persistência. (iii) Designs de interface e de interação, requer especialistas criar interfaces de usuário (GUI). O design da interface se preocupa com a especificação de detalhes, incluindo sintaxe

e semântica dos serviços fornecidos pela interface do usuário. (iv) Seleção e projeto de componentes: refere-se ao reúso de componentes já implementados.

Figura 16 - Caso de uso do sistema IDPT adaptativo.



Fonte: Adaptado de MUKHIYA et al. (2020).

SELMI et al., 2020 propõem a "PersoDiagMedi", uma plataforma colaborativa orientada a serviços para a personalização do serviço médico e a detecção precoce de doenças. A detecção dessas doenças é efetuada através de agentes de adaptabilidade. Os principais objetivos deste projeto são a integração de dados médicos estruturados e não estruturados, o desenho de uma ontologia médica e o desenvolvimento de um módulo de raciocínio e decisão baseados em recursos de adaptabilidade.

A arquitetura global do projeto PersoDiagMedi é composta por quatro camadas: (i) Camada de dados: representa as diferentes fontes de dados farmacêuticos, médicos, meteorológicos ou de mídia social; (ii) Camada de coleta, organização e armazenamento de dados: responsável pela integração dos dados que surgem da camada de dados, a construção de uma ontologia médica, o armazenamento dos dados em um banco de dados e o mapeamento entre os dados e a ontologia projetada; (iii) Camada de raciocínio, detecção e alerta: representa a parte inteligente da plataforma.

É responsável pela análise de dados, da camada anterior, para detecção de doenças emergentes através de um sistema multiagente adaptativo; (iv) Camada de interface gráfica do usuário (GUI): esta camada permite ao usuário, utilizar a plataforma.

JUNIOR et al. (2021) propõem um processo de desenvolvimento de software para aplicativos IoHT (*Internet of healthcare things*) adaptativos, os quais são

baseados em dados de padrões de movimento para dispositivos móveis. O processo proposto adapta o processo de desenvolvimento orientado ao reúso de componentes adicionando elementos relacionados à construção de sistemas adaptativos.

Cinco artefatos para apoiar este processo são apresentados: uma taxonomia relacionando dados de sensores e estado de saúde; um gráfico de correlação entre dados de sensores, características de movimento e situações de saúde, para auxiliar na escolha de componentes e servir como base de conhecimento para aplicação adaptativa; um modelo para construir regras de adaptação; um modelo baseado no ciclo de adaptação MAPE-K; e um *framework* baseado no modelo, para auxiliar na fase de implementação.

GRUA et al. (2022) apresenta uma arquitetura de software para permitir personalização baseada em inteligência artificial e adaptação executada por um aplicativo de disposto móvel chamado e-*Health*. A arquitetura de software proposta é composta por dois macros componentes principais: o *e-Health app*, um aplicativo executado no *smartphone* do usuário, e *Back-end,* componentes de software que provêm o funcionamento geral do aplicativo e são executados na computação em nuvem.

Para executar a personalização e adaptação, essa arquitetura leva em consideração o ambiente em que o usuário vive e utiliza o aplicativo e- *Health*. Precisamente, o ambiente representa a localização física do usuário. Então, o aplicativo e-*Health* adapta as atividades programadas do usuário de acordo com seu contexto operacional atual.

Esse contexto operacional significa, por exemplo, as condições ambientais e meteorológicas do local que usuário está. Um exemplo de adaptação é mudar uma atividade programada ao ar livre (em uma praça de uma cidade) para uma atividade dentro de um ginásio (com teto para proteger da chuva).

A seguir, uma comparação sobre os trabalhos apresentados neste capítulo é mostrada. Embora esses trabalhos tragam contribuições importantes para o estado da arte, algumas lacunas, as quais foram utilizadas nesta tese, foram identificadas e são listadas como segue.

3.4 Discussão Sobre os Trabalhos Investigados

A partir dos trabalhos correlatos apresentados neste capítulo, uma análise comparativa sobre algumas características de cada estudo considerado é feita.

Para facilitar o entendimento, alguns critérios de comparação foram utilizados nesta tese: (i) arquitetura de software voltada para a área da saúde, (ii) arquitetura que usa arquétipos, (iii) arquitetura distribuída, cujos componentes são implantados em nós de em uma rede de computadores (local, Internet ou Cloud), (iv) utilização de serviços web, (v) promove adaptabilidade, (vi) a arquitetura adaptável ao conjunto de arquétipos utilizados em uma GUI. As respostas sobre esses critérios são apresentadas na Tabela 1.

Como mostra a Tabela 1, todos os trabalhos são arquiteturas para área de saúde. A maioria dos trabalhos usam arquiteturas distribuídas e serviços web, exceto os trabalhos cinco e quatorze. Do trabalho 1 ao 13, são soluções com arquiteturas que utilizam arquétipos como padrão de representação de seus dados. Porém, somente o trabalho 5 propõe adaptabilidade em sua solução.

Tabela 1 - Análise comparativa dos trabalhos correlatos.

	Trabalhos	Critérios de Comparação						
		i	ii	iii	iv	v	vi	vii
1	SILVA-FERREIRA et al. (2012)	✓	✓	✓	✓	×	×	×
2	SUNDVAL et al. (2013)	✓	✓	✓	✓	×	×	×
3	VIEIRA-MARQUES et al. (2014)	✓	✓	✓	✓	×	×	×
4	REIS et al. (2018)	✓	✓	✓	✓	×	×	×
5	TARENSKEEN et al. (2020)	✓	×	×	✓	✓	×	×
6	PALIWAL et al. (2022)	✓	✓	✓	✓	×	×	×
7	EhrScape (BETTER, 2017)	✓	✓	✓	✓	×	×	×

8	MARCIA (GOMES et al., 2018)	✓	✓	✓	✓	×	×	×
9	Template4EHR (ARAÚJO et al., 2020)	✓	✓	✓	✓	×	×	×
10	CloudEHRServer (CABOLABS, 2022)	✓	✓	✓	✓	×	×	×
11	EHRBASE (2022)	✓	✓	✓	✓	×	×	×
12	EtherCIS (RIPPLE, 2022)	✓	✓	✓	✓	×	×	×
13	Zhang (2017)	✓	×	×	×	✓	✓	×
14	MALLYA & KOTHARI (2018)	✓	✓	✓	×	✓	✓	×
15	MUKHIYA et al., 2020	✓	✓	✓	×	✓	✓	×
16	SELMI et al., 2020	✓	✓	✓	×	✓	✓	×
17	JUNIOR et al., 2021	✓	✓	×	×	✓	✓	×
18	GRUA et al., 2022	✓	✓	✓	×	✓	✓	×

Fonte: O autor (2022).

Os trabalhos 14 ao 18 não fazem uso de arquétipos como padrão de representação de seus dados. Contudo, utilizam recursos de adaptabilidade executados em tempo de execução. Nenhum desses dezenove trabalhos possui uma arquitetura de software que identifique, em tempo de execução, os arquétipos utilizados pelo SIS. Além disso, nenhum desses trabalhos adapta a arquitetura que usa arquétipos, conforme descrito nos objetivos desta tese (Seção 1.4, Capítulo 1).

3.5 Considerações Finais

Neste capítulo, trabalhos correlatos foram apresentados. Inicialmente, mostrou-se o estado da arte de arquiteturas de software que utilizam arquétipos. Em seguida, ferramentas do estado da arte e da prática que constroem SIS que usam

arquétipos foram detalhadas, assim como suas respectivas arquiteturas de software. Posteriormente, foram descritos estudos propondo arquiteturas de software que dão suporte à área da saúde e possuem recursos de adaptabilidade. Finalmente, um comparativo entre todos os trabalhos analisados foi apresentado.

Conforme apresentado no Capítulo 1, a área da saúde é um ambiente cujos protocolos estão em constante modificação. Essas modificações muitas vezes demandam recursos de adaptabilidade no SIS para dar suporte às mudanças ocorridas no conjunto de arquétipos utilizados na GUI (por exemplo).

Como visto na Tabela 1, a maioria dos trabalhos que usa arquétipo não utiliza recursos de adaptabilidade, exceto o trabalho 5 (TAKENSKEEN et al., 2020) o qual faz uso da adaptabilidade conceitualmente apenas (sem implementações) e não promove a adaptabilidade da arquitetura em função do conjunto de arquétipos utilizados no SIS (essa proposta de adaptabilidade é explicado na seção 4.3.1 no Capítulo 4).

Além disso, a Tabela 1 mostra duas lacunas no estado da arte e da prática no que diz respeito à adaptabilidade de software em SIS que usam arquétipos. Primeira lacuna: os trabalhos que fazem uso de arquétipo não possuem recursos de adaptabilidade. Segunda lacuna: os trabalhos que fazem uso de adaptabilidade não utilizam arquétipos como padrão de representação de seus dados. Desta forma, a proposta de uma arquitetura de software para SIS que usam arquétipos e seja adaptável ao conjunto de arquétipos utilizados, guiou o desenvolvimento desta tese.

4 UMA ARQUITETURA DE SOFTWARE ADAPTATIVA BASEADA EM ARQUÉTIPOS PARA SISTEMAS DE INFORMAÇÃO EM SAÚDE

A *adaptabilidade de software* é um recurso da Engenharia de Software que provê a um software a capacidade de fazer ajustes estruturais ou comportamentais mediante a ocorrência de mudanças no ambiente em que estão alocados (GARLAN et al., 2004) . Dessa forma, esse conceito pode ser utilizado para criação de soluções mais resilientes às mudanças, o que é oportuno para softwares da área da saúde (GRUA et al., 2020; SILVA et al., 2022). No entanto, existem vários desafios relacionados ao uso desse conceito nessa área.

Sobre os desafios encontrados no Capítulo 3, há uma lacuna no que diz respeito à adaptabilidade de software em SIS que usam arquétipos. Dos trabalhos apresentados sobre arquiteturas de software que usam arquétipos (seção 3.1 e 3.2), somente TARENSKEEN et al. (2020) abordam o conceito de adaptabilidade de software, cujo trabalho traz uma abordagem conceitual sobre como o uso de arquétipos promove adaptabilidade em um SIS. Esse trabalho traz importantes contribuições, mas não promove a adaptação da arquitetura consoante mudança nos arquétipos utilizados no SIS.

Por outro lado, nos trabalhos que tratam conceitos e mecanismos de adaptabilidade no domínio de saúde (seção 3.3), verifica-se ausência de soluções voltadas para os SIS que usam arquétipos para representação de seus RES. Desta forma, esta tese propõe trabalhar os seguintes desafios:

- Identificação dos arquétipos utilizados na interface do usuário do SIS, em tempo de execução;
- Utilização de um mecanismo de adaptabilidade para alocar, em tempo de execução, um conjunto de componentes de software capaz de processar os dados representados pelos arquétipos identificados.

Neste contexto, para proporcionar adaptabilidade em SIS que usam arquétipos é proposta a construção de uma arquitetura de software contemplando os desafios mencionados. Essa arquitetura serve de guia na criação de soluções, fornecendo um *framework* para o desenvolvimento de software, de modo a

padronizar os componentes de software em toda arquitetura de software, cujos componentes da camada do servidor são adaptáveis aos arquétipos utilizados no SIS, na camada do cliente.

Conforme apresentado na seção 2.1 do Capítulo 2, a especificação de RES proposta pela openEHR é realizada por meio de arquétipos (explicado na seção 2.1 no Capítulo 2). Desse modo, esta tese propõe uma arquitetura de software adaptativa baseada em arquétipo, cujo objetivo principal é fornecer um *framework* para o desenvolvimento de software que usam arquétipos, de modo a proporcionar uma adaptabilidade de software em tempo de execução.

Assim, espera-se assegurar a padronização dos componentes de software utilizados em toda arquitetura de software, e permitir a adaptação dos componentes de software da camada do servidor com relação a alterações feitas no conjunto de arquétipos utilizados na camada do cliente.

A seguir, a Seção 4.1 descreve um cenário funcional para exemplificar um contexto no qual a proposta desta tese se aplica. Na Seção 4.2 são apresentados os requisitos funcionais e não funcionais da arquitetura proposta. Em seguida, a Seção 4.3 detalha essa arquitetura e finalmente, as considerações finais são feitas na Seção 4.4.

4.1 Cenário Funcional

Esta seção descreve um cenário funcional, por meio do qual se levantaram os requisitos abordados na arquitetura de software proposta neste capítulo. Nesse cenário, Santa Luzia é uma unidade básica de saúde (UBS) situada em Recife/PE (Brasil) e possui um SIS chamado LuziaSYS, com o qual os profissionais da saúde (enfermeiros e médicos) fazem a gestão clínica dos dados dos pacientes dessa UBS. Neste caso, esses profissionais da saúde são os usuários de LuziaSYS.

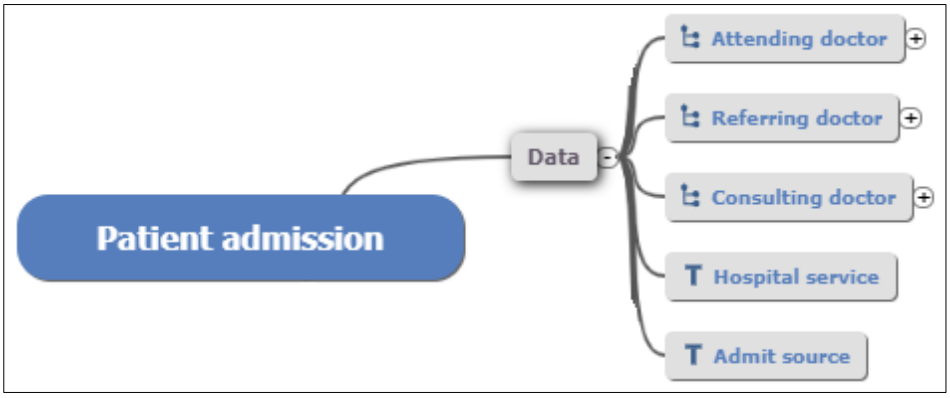
O LuziaSYS foi desenvolvido em uma arquitetura Cliente-Servidor por uma equipe interna de engenheiros de software. Na camada do cliente, utilizou-se linguagens computacionais que permitem executar o LuziaSYS em um navegador de Internet, por exemplo: linguagens HTML e Javascript. Na camada do servidor,

utiliza-se uma API (conceito explicado na seção 3.1 do Capítulo 3) construído com a linguagem Java e um banco de dados (chamado PostgreSQL).

Essa API dá suporte às requisições iniciadas pelos usuários de LuziaSYS. Dois exemplos dessas requisições são o envio de dados do registro de um paciente para persistência no banco de dados e uma consulta sobre dados de um paciente. Essas requisições trafegam em uma rede local, na qual estão todos os computadores que dão suporte aos registros administrativos da UBS Santa Luzia.

Quando um paciente novo entra nessa UBS, um profissional da recepção abre a aplicação (isto é, o LuziaSYS). Em seguida, clica-se em "*Registro de Paciente*" e inicia o cadastro de dados do paciente. Essa atividade é feita por meio de perguntas ao paciente e da solicitação de documentos. Para desenvolver os componentes de software que permitam o "*Registro de Paciente*" em LuziaSYS, a equipe de Engenharia de Software solicitou a um especialista openEHR, os arquétipos que são utilizados para modelar os RES dessa funcionalidade. Nesse caso, o especialista sugeriu utilizar o arquétipo *Patient Admission*, o qual é ilustrado na Figura 17.

Figura 17 - Arquétipo *Patient Admission.*



Fonte: Clin*ical Knowledge Manager (openEHR, 2022).*

Esse arquétipo possui diversos atributos, como: *attending doctor*, referring doctor, hospital service, dentre outros, os quais correspondem aos dados das funcionalidades de registro de paciente. Na camada do servidor, a API foi implementada com diversas funções que permitem executar as operações CRUD (*Create*, *Read*, *Update* e *Delete*) dos dados gerenciados por essa funcionalidade.

Porém, devido a pandemia de COVID-19, em março de 2019 o Ministério da Saúde do Brasil implementou modificações em alguns protocolos de saúde em todo país. Uma das modificações incluiu a adição do protocolo *fast-track* como parte do protocolo de admissão do paciente (G. BRASILEIRO, 2020). Isso significa que mais dados do paciente passaram a ser exigidos quando um paciente solicita atendimento em uma UBS. Alguns desses dados são: se tem sintomas de gripe, se tem tosse, febre, dificuldade para respirar, etc.

Desta forma, os SIS das UBS de todo Brasil (por exemplo, o LuziaSYS) ficaram obsoletos. Para atender as mudanças impostas pelo Ministério da Saúde e não ficarem obsoletos, houve a necessidade de adaptar os SIS a um conjunto novo de requisitos e funcionalidades. No caso da UBS Santa Luzia, houve um trabalho para modificar LuziaSYS para deixá-lo em conformidade com as novas regras.

Inicialmente, um engenheiro de requisito levantou os novos requisitos e funcionalidades necessários para que LuziaSYS atendesse a nova demanda (isto é, o protocolo *fast-track*). Em seguida, um especialista openEHR selecionou um conjunto de arquétipos para representar os dados da nova funcionalidade que executaria o protocolo *fast-track*, por exemplo: `story.v1`, `body_temperature.v2`, `symptom_sign-cvid.v0`, `travel_ history.v0`, `health_risk-covid.v0`. Então, esses arquétipos foram repassados para a equipe de engenharia de software, a qual iniciou a implementação das novas funcionalidades.

Ao adicionar esses arquétipos nos componentes de software na camada do cliente, a equipe de engenharia de software observou que a camada do servidor também precisava de alterações para processar os dados dos novos arquétipos adicionados ao sistema. É importante ressaltar que, esse trabalho efetuado em LuziaSYS, foi necessário porque os componentes de software da camada do servidor não eram adaptáveis às modificações que ocorrerem na camada do cliente. Desta forma, ao adicionar ou remover arquétipos nos componentes de software na camada do cliente, a camada do servidor também teve de ser modificada.

Nesse cenário funcional, verifica-se que a equipe de engenharia de software utilizou ferramentas disponíveis no estado da arte e da prática para construir componentes de software da camada do cliente a partir dos arquétipos escolhidos.

Contudo, se houver novas modificações nesse conjunto de arquétipos, os componentes na camada do servidor não sendo adaptáveis, precisarão ser modificados de novo pela equipe de engenharia de software.

Desta forma, qualquer mudança no conjunto de arquétipos da camada do cliente, implica aumento de custo de manutenção e de alocação de recursos para adaptar a camada do servidor do SIS às mudanças. Como exemplo de alocação de recurso, há a demanda de tempo para executar a etapa de escrita de código, de revisão, de teste, assim como a implantação desses componentes de software no ambiente de produção, e se houver erros, vai requerer retrabalho nas etapas anteriormente descritas.

Assim, a equipe de engenharia de software da UBS Santa Luzia resolveu desenvolver uma solução que permite ao LuziaSYS, utilizar componentes de software (na camada do servidor) adaptáveis às alterações do conjunto de arquétipos utilizados nos componentes de software na camada do cliente.

4.2 Requisitos de Software

Esta seção mostra os requisitos funcionais e não funcionais que devem ser atendidos pela arquitetura proposta nesta tese. De acordo com BASS et al. (2022), os requisitos funcionais estão relacionados às funcionalidades que devem ser proporcionadas pelas aplicações que utilizam os dados gerenciados pela arquitetura. Já os requisitos não funcionais são atributos de qualidade que devem ser atendidos pela arquitetura.

4.2.1 Requisitos Funcionais e Não Funcionais

Os Requisitos Funcionais (RF) definem o que um sistema deve fazer e como ele deve se comportar ou reagir a estímulos (BASS et al., 2022). Logo, para alcançar o objetivo da arquitetura projetada nesta tese, os requisitos funcionais devem fornecer funcionalidades que auxiliem na adaptação dessa arquitetura aos arquétipos utilizados no SIS e, assim, fazer o processamento CRUD de seus dados.

A adaptabilidade de software viabiliza que um SIS modifique seu comportamento ou estrutura para se adaptar a alguma mudança que ocorra durante sua execução, sem gerar interrupções em sua execução (GRUA et al., 2020, SILVA et al., 2022). Nesse sentido, a arquitetura proposta provê uma solução que permite a adaptação de um SIS aos arquétipos usados na interface do usuário, na camada do cliente.

Desta forma, se houver alterações no conjunto de arquétipos utilizados no SIS, os componentes de software da camada do servidor possuem funcionalidades que permitem a sua adaptação em tempo de execução. Por fim, a execução do SIS não é interrompida.

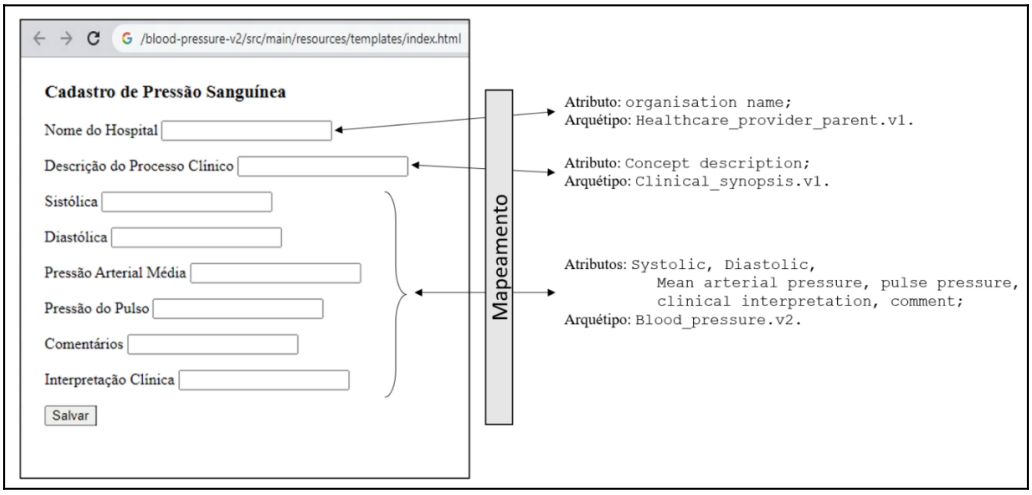
Partindo desse pressuposto, foram identificadas funcionalidades que permitem essa adaptação. Portanto, nessa solução, qualquer SIS que usa arquétipo pode utilizar uma arquitetura de software que se adapta aos arquétipos utilizados na camada do cliente. Ainda, nesta tese, pensou-se em uma abordagem de adaptação em tempo de execução. Isto é, a adaptação ocorre em cada requisição enviada da camada do cliente para a camada do servidor, na qual um conjunto de RF são executados. Desta forma, os RF identificados para esta arquitetura foram:

Cinco RF foram especificados para os componentes da camada do cliente:

- **RF1**: Na construção da interface do usuário (por exemplo: GUI), fazer o mapeamento de campos de inserção de dados com os respectivos atributos de um ou mais arquétipos. Por exemplo, conforme ilustrado na Figura 18, um formulário web pode conter campos para Cadastro de Pressão Sanguínea, os quais podem ser mapeados para os respectivos atributos e arquétipos.
- **RF2**: No uso do SIS, permitir que seus usuários façam requisições e obtenham resposta. Isto é, tal requisito corresponde a registrar ou consultar dados de pacientes. Nesta tese, usuário de SIS são profissionais da área de Saúde, por exemplo: enfermeiros, médicos, assistentes e auxiliares administrativos da Saúde.
- **RF3**: Permitir ao SIS requisitar o protocolo de comunicação (conceito apresentado na seção 2.2.1 do Capítulo 2) dos serviços web que processarão os dados enviados na requisição do usuário. Essa requisição deve ser efetuada em tempo de execução. Nessa requisição, um mecanismo de

adaptação deve ocorrer quando o dado enviado pelo usuário contiver diferentes arquétipos.

Figura 18 - Mapeamento de campos de um formulário web com os respectivos arquétipos.



Fonte: O autor (2022).

- **RF4**: Permitir ao SIS requisitar o processamento das operações CRUD (sobre os dados enviados na requisição do usuário) e a arquitetura de software executar esse processamento mesmo que haja mudança nos arquétipos utilizados na interface do usuário.

Cinco RF são especificados para o componente que executará a adaptação:

- **RF5**: Identificação de um conjunto de arquétipos utilizados na requisição do usuário;
- **RF6**: Localização do protocolo de comunicação de um ou mais componentes de software capazes de executar operações CRUD sobre os dados de arquétipos enviados na requisição do usuário. Esses protocolos de comunicação são explicados na seção 2.2, no Capítulo 2;
- **RF7**: Verificar a disponibilidade de interação dos serviços web referente aos protocolos de comunicação localizados;
- **RF8**: Verificar as permissões de acesso do usuário (que iniciou a requisição) aos componentes de software localizados;
- **RF9**: Verificar as regras de interoperabilidade dos componentes localizados em RF8, por exemplo: tipos de dados, regras de criptografia;

Por fim, um RF é especificado para a componente que executa o processamento dos dados:

- **RF10**: Processamento de operações CRUD dos dados representados pelos arquétipos contidos na requisição do usuário. O conceito de “operações CRUD” refere-se a criação, leitura, atualização e exclusão de dados em um banco de dados.

Requisitos Não Funcionais (RNF) são aspectos qualitativos da arquitetura como um todo (BASS et al., 2022). Nesta tese, abordou-se o requisito não funcional de adaptabilidade de software, ao qual se denominou **RNF1**. A seguir, descreve-se as especificações da arquitetura proposta nesta tese.

4.3 Arquitetura de Software Proposta

Esta arquitetura tem o papel de definir os componentes da implementação de soluções para SIS que usam arquétipos e cujo processamento de dados possui um mecanismo de adaptação que ocorre em tempo de execução. Essa adaptação é explicada a seguir.

Antes de detalhar como é efetuado o mecanismo de adaptação, é importante informar que o protocolo de comunicação utilizado na interação dos componentes da arquitetura proposta segue o conceito apresentado na seção 2.2, no Capítulo 2. Nos exemplos utilizados nesta seção, o protocolo HTTP é utilizado tanto nas requisições e como nas respostas, porém não é um critério obrigatório, podendo ser utilizado qualquer outro protocolo (por exemplo, REST).

A seguir, a seção 4.3.1 descreve o mecanismo de adaptação utilizado na proposta desta tese. A seção 4.3.2 descreve as visões da arquitetura. A seção 4.3.3 descreve o comportamento esperado na execução dos *containers* e componentes especificados.

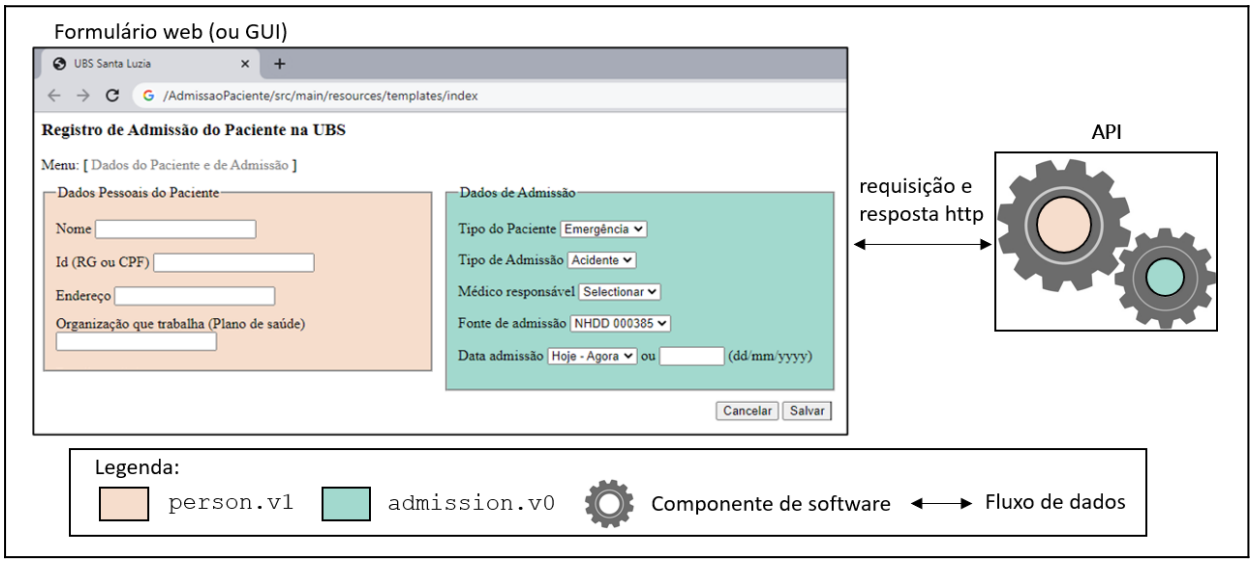
4.3.1 Mecanismo de Adaptação

Para melhor entendimento da adaptação proposta nesta tese um exemplo é explicado a seguir. A Figura 19 mostra um SIS composto por uma interface do

usuário (GUI) e uma API para processamento dos dados dessa GUI. A funcionalidade desse SIS é fazer o registro de admissão do paciente em uma UBS. Para construir esse formulário web, os arquétipos `person.v0` e `admission.v0` foram utilizados.

Desta forma, nesse formulário existem campos que se referem aos atributos desses arquétipos (por exemplo: nome, id, endereço, tipo do paciente, tipo de admissão, *etc*). A finalidade desse formulário web é permitir que um usuário desse SIS faça o registro de admissão de pacientes em uma Unidade Básica de Saúde (UBS).

Figura 19 - Formulário web e API construídos com os arquétipos `person.v0` e `admission.v0`.



Fonte: O autor (2022).

Ao utilizar esse SIS, um usuário digita nesse formulário web os dados dos respectivos campos de *Dados Pessoais do Paciente* (e.g., nome, id, endereço, *etc*) e *Dados de Admissão* (e.g., tipo de paciente, tipo de admissão, *etc*). Quando o usuário aciona o botão *Salvar*, uma requisição HTTP é iniciada, isto é, o usuário solicita que os dados sejam enviados a uma API, para executar um processamento sobre esses dados. Após esse processamento, uma resposta HTTP é retornada, isto é, a API envia uma mensagem de sucesso ou falha. Essa requisição e resposta HTTP podem ser enviadas por uma rede de computadores local ou pela Internet.

Conforme apresentado nos trabalhos correlatos, no domínio da Saúde, as APIs são construídas para receber e processar um conjunto de dados previamente especificado. Isto é, se esse conjunto de dados mudar, a API processa com erros ou falhas (BETTER, 2017; REIS et al, 2018; CABOLABS, 2022; EHRBASE, 2022). Como mostrado na Figura 19, na API, há um componente de software para processar o conjunto de dados de *Dados Pessoais do Paciente* (campos relacionados ao arquétipo `person.v1`) e um outro componente para processar os dados de *Dados de Admissão* (relacionados ao arquétipo `admission.v0`). Contudo, haverá um problema nesse SIS se houver uma mudança nos campos desse formulário web.

A Figura 20 ilustra a adição do campo *Avaliação de Risco de Infecção* ao formulário web. Esse campo é relacionado ao arquétipo `health_risk-covid.v0`. Porém, a API não foi projetada para processar os dados desse novo campo e, se o usuário acionar o botão *Salvar*, os dados referentes a esse novo campo não serão processados por essa API.

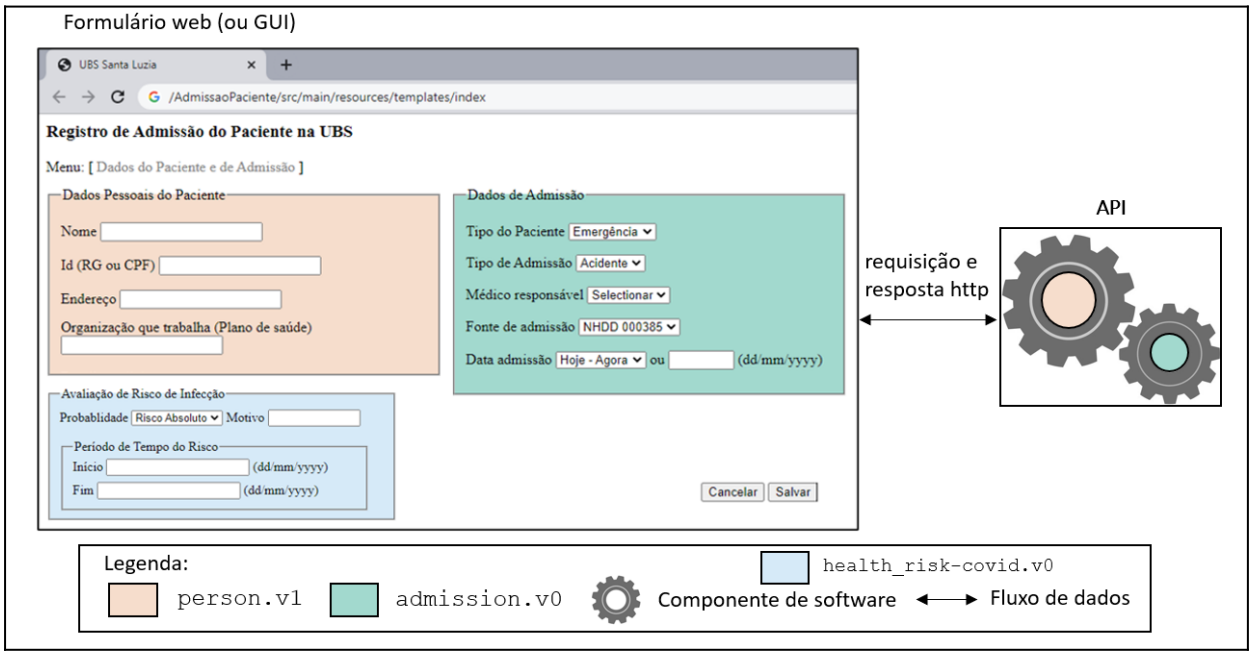
Assim, essa modificação na GUI também demanda mudanças nos componentes de software da API para que os dados do novo campo adicionado sejam processados. Este cenário significa que essa arquitetura não é adaptável às alterações do conjunto de arquétipos utilizados na GUI. Conforme apresentado na Introdução, essa falta de capacidade de adaptação demanda esforços dispendiosos (de tempo e financeiro) para deixar todos os componentes da arquitetura em concordância.

Desta forma, para mitigar esse problema de arquiteturas que não se adaptam a essas alterações de arquétipos utilizados na interface do usuário, o mecanismo de adaptação proposto nesta tese é explicado a seguir. Primeiramente, a arquitetura proposta identifica quais são os arquétipos utilizados no SIS.

Em seguida, em tempo de execução, a arquitetura identifica um conjunto de componentes de software, disponível na Internet, os quais conseguem processar os dados desses arquétipos identificados. Finalmente, a arquitetura adapta esse conjunto de componentes de software ao SIS para processamento.

Essa adaptação, baseada no conjunto de arquétipos utilizados em um SIS, é o que diferencia a proposta desta tese dos trabalhos correlatos apresentados no Capítulo 3. A seguir, a arquitetura proposta é descrita.

Figura 20 - Formulário web e API construídos com diferentes arquétipos.



Fonte: O autor (2022).

4.3.2 Níveis da Arquitetura

A arquitetura foi especificada com base no levantamento dos requisitos funcionais e não funcional listados na Seção 4.2.1. Então, os elementos da arquitetura foram pensados no intuito de prover as funcionalidades necessárias para executar esses requisitos.

Para documentar esta arquitetura, as diretrizes definidas no modelo C4 (BROWN, 2022) foram utilizadas, as quais são aplicadas em diversos estudos na Engenharia de Software (VÁZQUEZ-INGELMO et al., 2020; FOWLER, 2022; TIAN et al., 2022). Assim, a arquitetura foi descrita em diagramas que representam 4 níveis: contexto, contêiner, componentes e código. Esses níveis são descritos a seguir.

O nível de contexto é uma representação mais ampla do sistema que está sendo descrito e sua relação com outros sistemas e usuários. O público-alvo desse nível são os técnicos e não técnicos dentro e fora da equipe. O nível de contêiner

representa um serviço ou aplicativo individual. O aplicativo deve ser uma unidade executável ou implantável separadamente. Ele fornece diagramas focados em tecnologia de alto nível. O público-alvo desse nível são os desenvolvedores, arquitetos de software dentro e fora da equipe. O nível de componentes mostra como um contêiner é composto de vários componentes. Fornece mais detalhes sobre as responsabilidades e os detalhes de tecnologia/implementação dos componentes. O público-alvo desse nível são desenvolvedores e arquitetos de software. O nível de código fornece informações detalhadas sobre como cada componente é implementado e foi projetado para desenvolvedores de software.

Nível de Contexto

Conforme ilustrado na Figura 21, o nível de contexto da arquitetura proposta é composto por dois ambientes (*empresa* e *internet ou rede local*), dois tipos de pessoas (*engenheiro de software* e *profissional da saúde*), dois tipos de software (*frontend* e o *backend* do SIS) e um modelo de dados, os quais são descritos a seguir:

(i) *Empresa de saúde* é um ambiente físico representado por um hospital, clínica ou laboratório;

(ii) *Engenheiros de software* é uma pessoa que constrói software utilizado em uma *empresa de saúde*;

(iii) *Profissional da saúde* é um funcionário de uma empresa de saúde que interage com um SIS. Esse profissional pode ser uma enfermeira, um médico, um assistente administrativo;

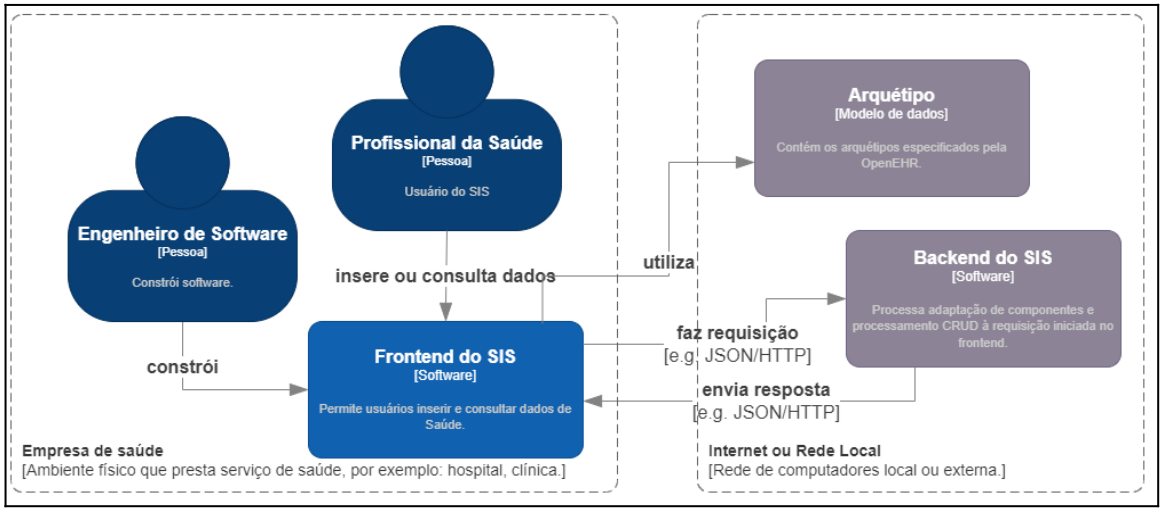
(iv) *Frontend do SIS* é um software que interage com o usuário da SIS, neste caso trata-se do *profissional da saúde;*

(v) *Internet ou rede local* é um ambiente virtual acessado por uma rede de computadores;

(vi) *Arquétipo* é um modelo de dados especificado e mantido pela organização openEHR. Esse modelo de dados está disponível na Internet e pode ser acessados pelo endereço eletrônico https://ckm.openehr.org;

(vii) *Backend do SIS* é um software responsável pela execução de adaptação dos componentes de software dessa arquitetura e, também, pelo processamento CRUD dos dados de um SIS que usa arquétipo.

Figura 21 - Nível de contexto da arquitetura proposta: pessoas, sofware, modelo de dados e suas relações.



Fonte: O autor (2022).

A relação entre esses elementos listados acima ocorre da seguinte forma: Um engenheiro de software constrói um *frontend do SIS*. Esse software é utilizado em uma *empresa de saúde* para processar os dados dos procedimentos diários executados por ela. Um ou mais *profissional de saúde* insere ou consulta dados nesse software. O *frontend do SIS* é construído baseado nas especificações de *arquétipos*. Quando um *profissional de sáude* utiliza o *frontend do SIS*, requisições são enviadas ao *backend do SIS,* o qual também envia uma resposta referente a essa requisição. O *arquétipo* e o *backend do SIS* estão disponíveis na *internet ou rede local*.

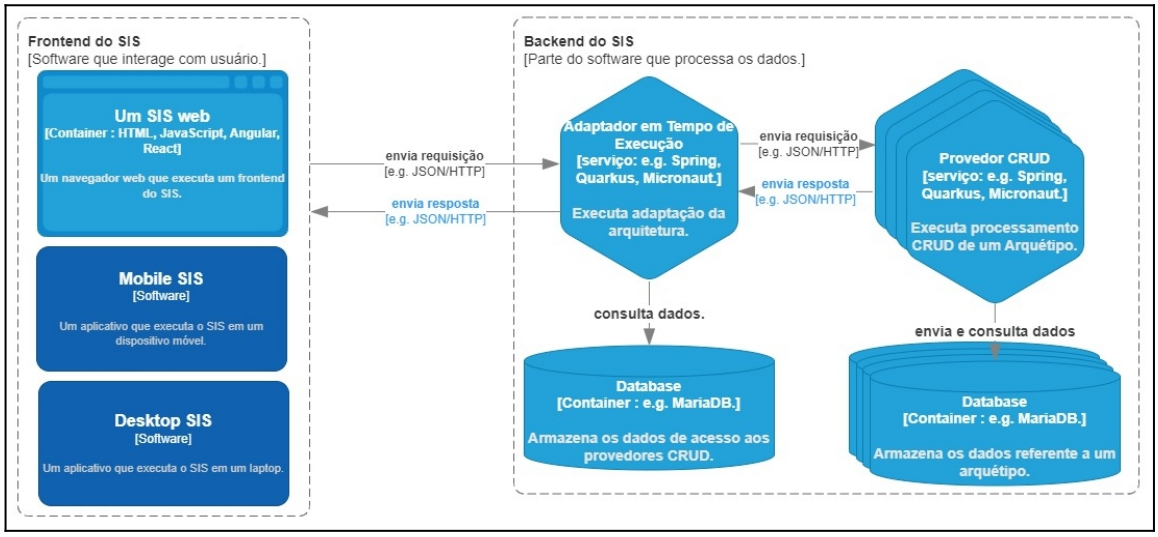
Nível de *Container*

O nível de *container* da arquitetura proposta apresenta uma visão mais detalhada do *frontend* e *backend do SIS*. Como mostrado na Figura 22, o *Frontend*

do SIS é apresentado com três containers: um SIS web, um SIS móvel e um SIS desktop.

O primeiro executa suas operações em um navegador web, o segundo, em um dispositivo móvel, o terceiro, em um laptop ou computador. O *Backend do SIS* é composto por *Adaptador em Tempo de Execução* e *Provedor CRUD, os quais são explicados a seguir.*

Figura 22 - Nível de *containers* da arquitetura proposta e fluxo de dados entre os containers e componentes.



Fonte: O autor (2022).

- ***Adaptador em Tempo de Execução***: é um serviço disponível na Internet ou em rede local que executa um processo de adaptação na arquitetura. Esse *container* tem um banco de dados associado, o qual contém protocolos de comunicação (explicado na Seção 2.2.1) dos *Provedores CRUD*, descritos a seguir.
- ***Provedor CRUD***: é um serviço disponível na Internet ou em uma rede local que executa as operações CRUD sobre dados representados por um arquétipo. A arquitetura proposta nesta tese é composta por um conjunto de *Provedores CRUD*. Cada *Provedor CRUD* executa as operações CRUD sobre dados representados por um arquétipo específico e tem um banco de dados associado, no qual se persiste e consulta dados referentes a esse arquétipo.

Se a versão do arquétipo mudar, um novo *provedor CRUD* deve ser construído.

Importante ressaltar que, para executar a arquitetura proposta nesta tese, os *provedores CRUD* tem que ser previamente construídos e disponbilizados na Internet. E seus dados de comunicação tem que ser persistidos no *adaptador em tempo de execução* para posterior processo de localização.

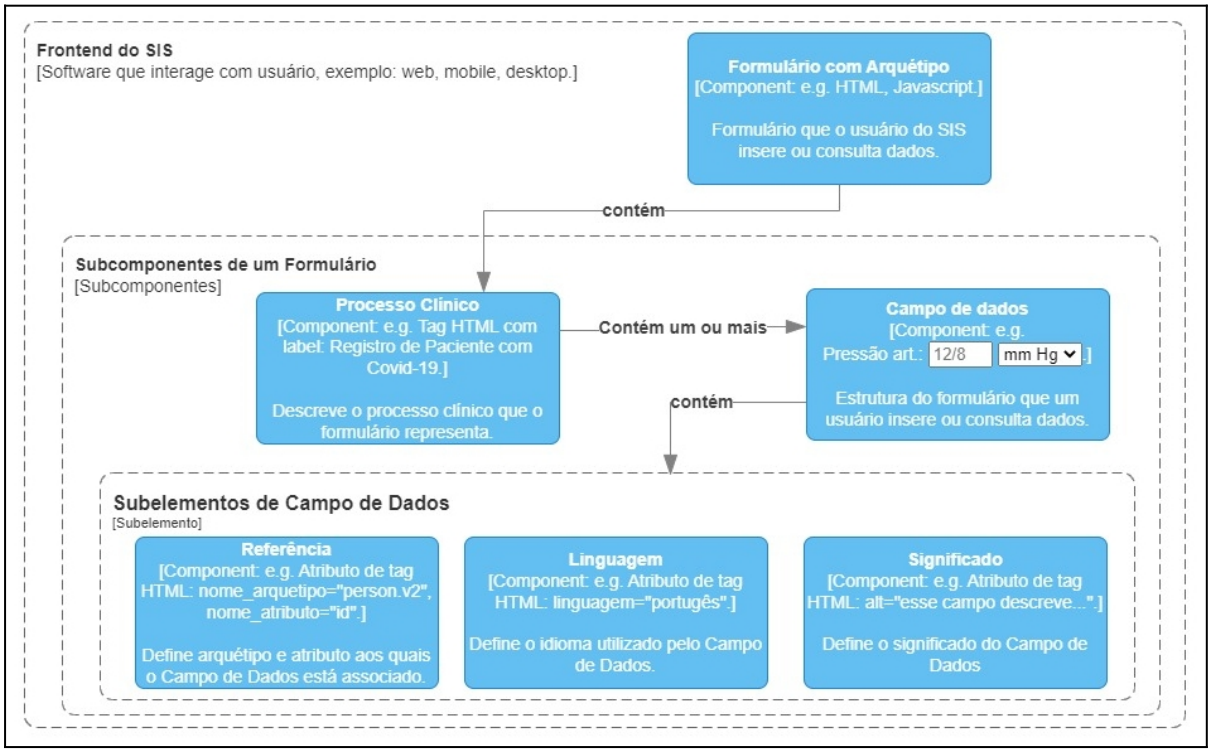
Nível de Componentes

Esse nível apresenta com mais detalhes a especificação da arquitetura proposta para o frontend e backend do SIS, conforme descrito a seguir. Um *Frontend do SIS* é composto por *formulário com arquétipo* e *controlador*.

Nível de componentes de *formulário com arquétipo*: Um *Formulário com Arquétipo* é um componente por meio do qual o usuário interage com o SIS. Nesse componente, um conjunto de arquétipos obrigatoriamente obtidos do CKM são utilizados para construção de interface gráfica (web, desktop ou mobile). Conforme mostrado na Figura 23, *formulário com arquétipo* é composto por um conjunto de componentes, os quais são descritos a seguir.

- ***Processo Clínico***: recurso que descreve o processo clínico ao qual um *formulário com arquétipo* representa, por exemplo: “Registro de Paciente com Covid-19”. Um *processo clínico* é composto por um ou mais *campos de dados*, cuja definição é descrita a seguir. A mudança (adição ou remoção) de um conjunto de *campos de dados* representa uma mudança da versão do respectivo *processo clínico*.
- ***Campo de Dados:*** recurso utilizado para compor um formulário com arquétipo. É por meio de campos de dados que um usuário do SIS insere ou consulta dados em um SIS. Um exemplo desse campo é “Pressão arterial”, conforme ilustrado na Figura 23. Cada campo de dados têm uma referênciaI, explicado a seguir.
- ***Referência***: recurso utilizado para definir um arquétipo e respectivo atributo que os *campos de dados* está associado.

Figura 23 - Nível de componentes de *Formulário com Arquétipo*.



Fonte: O autor (2022).

Para melhorar a usabilidade do usuário de SIS, *campos de dados* pode ter também, opcionalmente, outros dois componentes *linguagem* e *significado*, os quais são descritos a seguir.

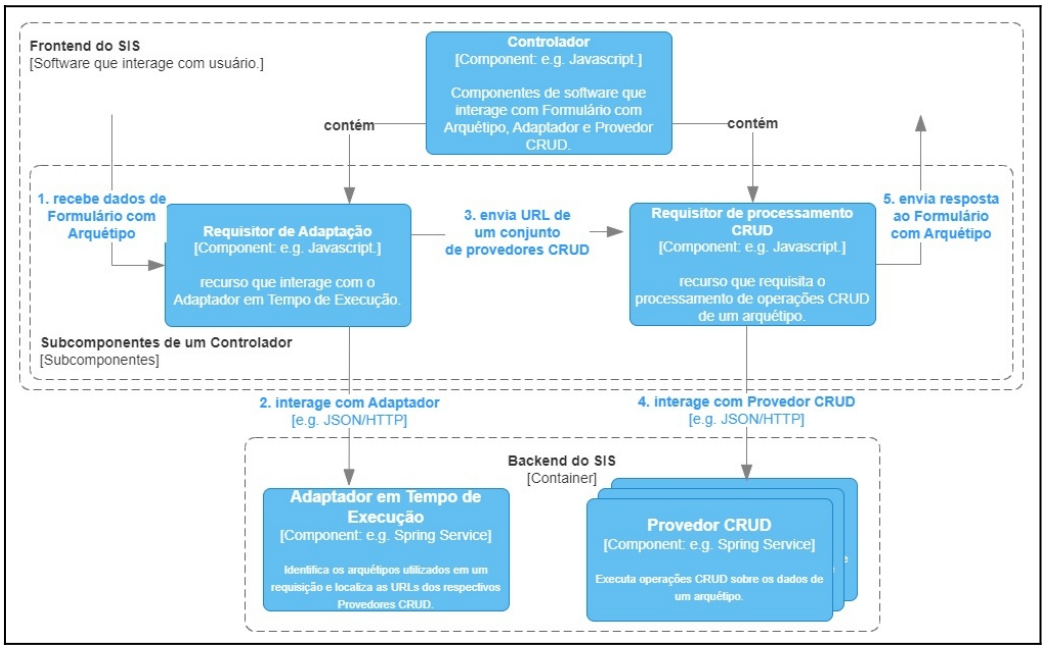
- ***Linguagem***: recurso para definir o idioma utilizado pelo *campo de dados*. Arquétipos foram especificados pela openEHR com “definições de termo” (OpenEHR, 2022), os quais contém os idiomas que seus atributos podem ser reutilizados, por exemplo: "EN" ou “PT” (idiomas inglês e português, respectivamente).
- ***Significado***: recurso que define o significado de um *campo de dados*, isto é, uma descrição breve sobre o conceito que o respectivo *campo de dados* representa. As “definições de termo” de um arquétipo contém uma descrição do conceito que o atributo representa, por exemplo: o atributo *Organisation identifier*, do arquétipo `healthcare_provider_ parent.v1`, tem como descrição “*The unique identifier of the organisation*”. Essa descrição pode ser reutilizada no subcomponente *significado*.

Nível de componentes de *controlador*: A Figura 24 ilustra o componente *controlador*, a relação entre seus subcomponentes e o fluxo de dados. O *controlador* é composto:

- *Requisitor de Adaptação* é um componente de software que interage com o componente *adaptador em tempo de execução*. Ele envia os dados inseridos em um *formulário com arquétipo* (com seus respectivos arquétipos) e recebe os dados de comunicação de um conjunto de *provedores CRUD* referente a esses arquétipos.
- *Requisitor de Processamento CRUD* é um componente de software que interage com os componentes de *provedores CRUD*. Nesse momento do processo, já se sabe a qual conjunto de *provedores CRUD* o *controlador* pode interagir.

O fluxo de dados apresentados na Figura 24 significa que, primeramente, *requisitor de adaptação* recebe os dados que o usuário inseriu e iniciou uma requisição no *formulário com arquétipo*.

Figura 24 - Nível de componentes *Controlador* e suas interações com *Adaptador e Provedor*.



Fonte: O autor (2022).

Em seguida, interage com *adaptador em tempo de execução*, do qual recebe um conjunto de dados de comunição de um conjunto de *provedores CRUD*. Em

terceiro, o *requisitor de processamento CRUD* recebe os dados de comunicação de um conjunto de *provedores CRUD*. Em quarto, interage com os respectivos *provedores CRUD*. Por fim, envia uma resposta ao *formulário com arquétipo*.

Nível de componentes de *adaptador em tempo de execução:* A Figura 25 ilustra o componente *adaptador em tempo de execução*, a relação entre seus subcomponentes e o fluxo de dados. O *adaptador em tempo de execução* é composto por:

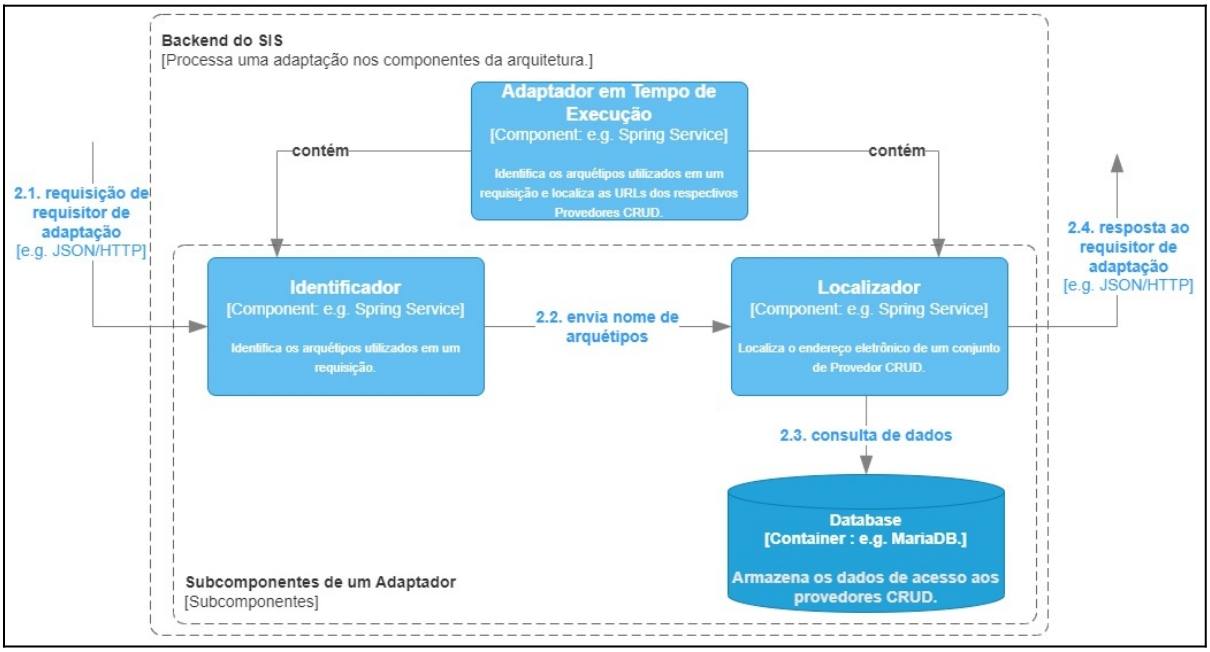
- *Identificador* é um componente de software que identifica os arquétipos utilizados nos dados de uma requisição;
- *Localizador* é um componente de software que execução a localização dos dados de comunição de um conjunto de *provedores CRUD*. Esses dados de comunicação são endereço eletrônico e o protocolo usado de comunicação (e.g., HTTP, REST).

O fluxo de dados mostrado na Figura 25 é uma derivação da interação 2 mostrada na Figura 24, por esta razão, o fluxo de dados da Figura 25 começa pelo número 2.1.

Então, primeramente, *identificador* recebe um conjunto de dados de *requisitor de adaptação* e identifica os arquétipos utilizados nessa requisição. Em segundo, após envia os nomes dos arquétipos identificados e os envia ao *Localizador*. Em terceiro, *localizador* busca os dados de comunição dos *provedores CRUD* referente aos arquétipos identificados. Por fim, envia ao requisitor de adaptação

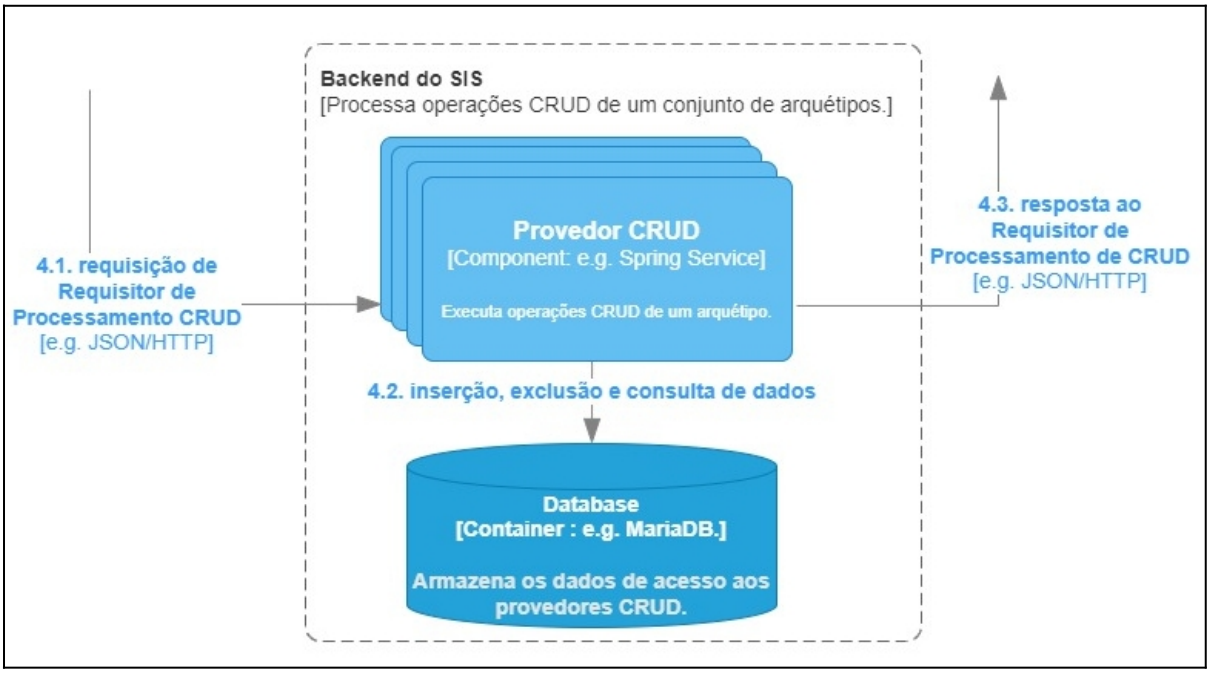
Nível de componentes de *provedor CRUD:* A Figura 26 ilustra o componente *provedor CRUD* e o fluxo de dados da arquitetura com esse componente. A arquitetura proposta é composto por um conjunto de *provedor CRUD*, cada qual executa operações CRUD sobre os dados representados por um arquétipo. Desta forma, cada provedor CRUD possui um banco de dados específico para consulta e armazenamento dos dados representados pelo seu respectivo arquétipo.

Figura 25 - Nível de componentes *Adaptador em Tempo de Execução* e seu fluxo de dados.



Fonte: O autor (2022).

Figura 26 - Nível de componentes e fluxo de dados de *provedor CRUD*.



Fonte: O autor (2022).

Nível de Código

Para representar o nível de código, nesta tese, optou-se em apresentar algoritmos e implementações dos componentes da arquitetura de software proposta.

Algoritmos de um *controlador* e de um *adaptador em tempo de execução* estão disponíveis no endereço eletrônico *https://www.cin.ufpe.br/~maps3/ahsa* (site destinado ao projeto de pesquisa desta tese), em 'Controller' e 'Runtime Adapter', respectivamente.

Implementações de *provedores CRUD,* referente ao arquétipo `blood_pressure.v2`, `family_history.v2` e `individual_personal.v0`, estão disponíveis no mesmo endereço eletrônico descrito no parágrafo anterior, na opção '*Implementations*'. Uma implementação em java do *provedor CRUD* referente ao arquétipo `blood_pressure.v2` também está disponível no Anexo I desta tese.

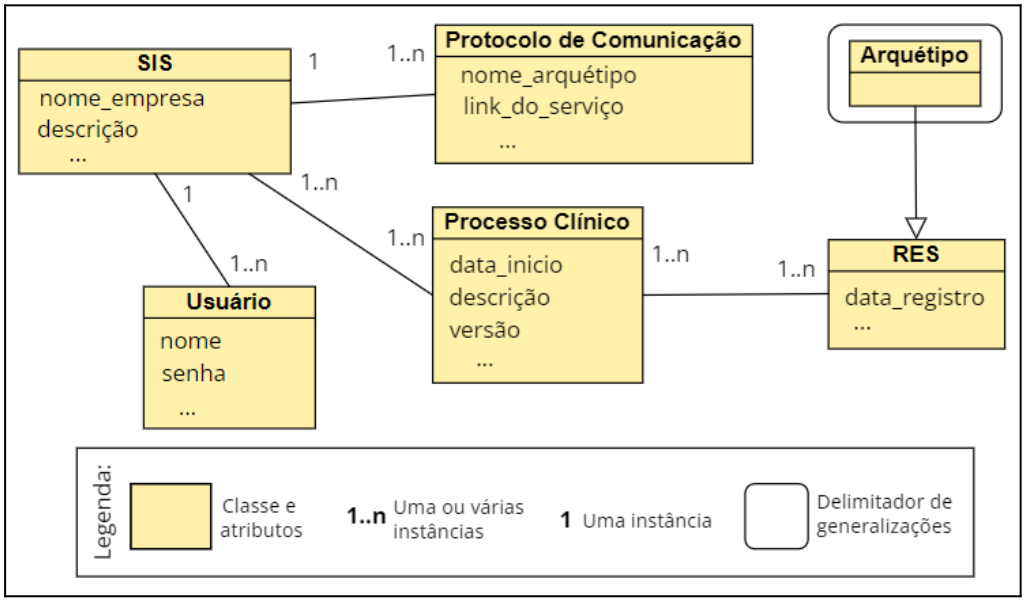
4.3.3 Visão de Modelo de Dados

A visão de modelo de dados descreve a estrutura dos dados usados no sistema como entidades e relacionamentos. Este tipo de visão ajuda a guiar a fase de implementação e melhora a modificabilidade em sistemas centrados em dados (CLEMENTS et al., 2010). A Figura 27 mostra a visão de modelo de dados da arquitetura proposta. Algumas entidades e um conjunto inicial de atributos (isto é, outros atributos podem ser adicionados posteriormente) foram especificados e são apresentados a seguir:

- **SIS**: entidade que representa uma aplicação ou sistema de informação da área de Saúde, que usa arquétipo e faz uso da arquitetura proposta nesta tese. Nessa estrutura de dados, deve conter nome da empresa e a descrição do SIS pra identificação;
- **Protocolo de Comunicação**: é uma entidade que representa os protocolos de comunicação que permite a interação do SIS com os *Provedores CRUD*. Nessa entidade, deve conter o endereço eletrônico de acesso ao *Provedor CRUD* e o nome do respectivo arquétipo. As reticências presentes nas

entidades apresentadas na Figura 27 significa que outros atributos não identificados nesta tese podem ser adicionados posteriormente;

Figura 27 - Visão de modelo de dados da arquitetura proposta nesta tese.



Fonte: O autor (2022).

- **Processo Clínico:** representa um processo clínico executado no SIS. Um processo clínico deve ter sua data de criaçã, sua descrição ou nome, e sua versão. Um mesmo processo clínico pode conter diferentes conjuntos de RES durante seu ciclo de vida. Essa mudança de RES é representado na versão do processo clínico.
- **Usuário:** representa um usuário da aplicação na arquitetura projetada nesta tese. Normalmente, esses usuários são profissionais da Saúde, por exemplo: técnicos de enfermagem, enfermeiros ou médicos. Os dados desses usuários são utilizados para identificar quem está usando o SIS e prover as devidas permissões de acesso aos componentes da arquitetura. Inicialmente, a entidade Usuário deve conter o nome e a senha de acesso.
- **RES** (ou Registro Eletrônicos de Saúde) é uma entidade que representa os dados de Saúde que trafegam no SIS, os quais são representados por arquétipos (conforme apresentado no delimitador de generalizações, na Figura 27). Esses dados de Saúde são relacionados aos pacientes e às

instituições de saúde. Além de todos os atributos inerentes ao arquétipo que representa um RES, nessa entidade também deve estar a data que o RES foi registrado pelo sistema.

A seguir, comportamentos da arquitetura proposta são mostrados. Essa documentação é um recurso adicional às visões apresentadas anteriormente. Esses comportamentos mostram como os componentes iniciam e finalizam suas execuções, durante o processamento da arquitetura.

4.3.4 Comportamentos da Arquitetura

A documentação de comportamentos complementa as descrições das visões da arquitetura porque apresenta elementos arquiteturais que interagem entre si. Os comportamentos proveem benefícios que auxiliam o desenvolvimento da arquitetura, assim como a manutenção dos sistemas (CLEMENTS et al., 2010). Duas ferramentas são utilizadas nesta tese para explicar os comportamentos da arquitetura proposta, são elas: diagramas de casos de uso e diagramas de sequência. Ambos são utilizados para documentar comportamentos, conforme explicado a seguir.

A elaboração de um diagrama de casos de uso é o primeiro estágio para documentação dos comportamentos de uma arquitetura. Este tipo de diagrama ajuda na visualização dos requisitos funcionais de um sistema e mostram como esses requisitos podem ser usados por atores (PRESSMAN, 2019). Ainda, os diagramas de sequência também são uma ferramenta útil na documentação dos comportamentos de uma arquitetura. Eles servem para demonstrar as interações existentes entre os elementos documentados (BASS et al., 2022).

As Figuras 28 e 29 mostram diagramas de caso de uso e de sequência, respectivamente. Esses diagramas ilustram os comportamentos encontrados no cenário funcional apresentado na Seção 4.1. A apresentação desses comportamentos é para exemplificar em qual contexto ocorre mudanças no conjunto de arquétipo usado em um SIS, e como arquitetura proposta se adapta mediante essas mudanças.

Conforme mostrado na Figura 28, o diagrama possui um conjunto de atores, os quais são:

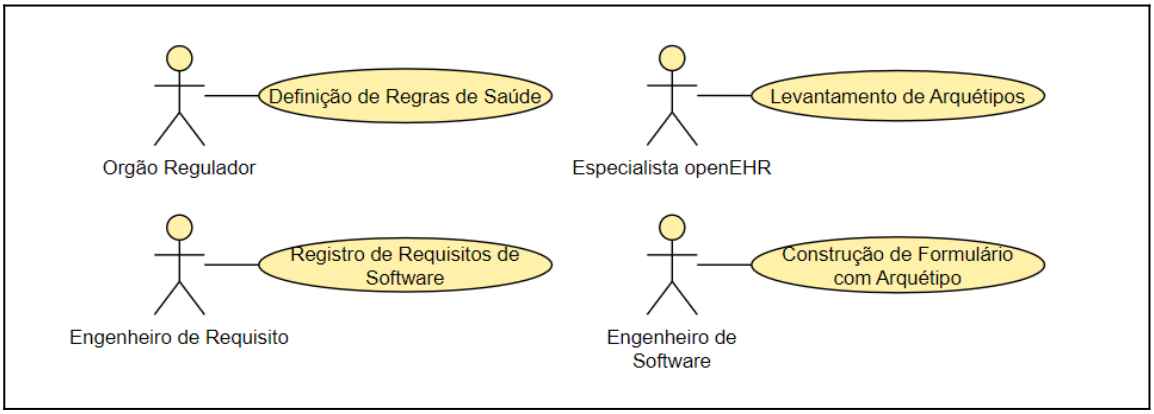
- **Órgão Regulador:** é uma organização, empresa, ou órgão de um governo, que tem autorização para definir (e redefinir) um conjunto de regras de Saúde, as quais têm de ser obedecidas por empresas que prestam serviços de Saúde (por exemplo: UBS, clínicas e hospitais). Um exemplo de órgão regulador é o Ministério da Saúde do Governo Brasileiro.
- **Engenheiro de Requisito:** é uma pessoa encarregada por especificar e fazer a manutenção dos requisitos de um software. Um requisito de software é um conjunto de atividades ou funcionalidades a serem realizadas, e define quem as executam, quais artefatos de entrada são necessários e quais artefatos de saída são produzidos.
- **Especialista openEHR:** é uma pessoa que tem conhecimento sobre os arquétipos openEHR e auxilia na escolha de quais arquétipos representarão os RES existentes nos requisitos e funcionalidades de um SIS.
- **Engenheiro de Software:** representa um conjunto de pessoas responsáveis pela construção (escrita de códigos) e manutenção (revisão e testes) dos componentes de software de um SIS.

Os casos de uso identificados para o cenário funcional, e mostrados na Figura 28, são:

- **Definição de Regras de Saúde**: caso de uso que define um conjunto de regras e protocolos de Saúde, cuja aplicação é recomendada ou obrigatória às organizações de saúde. Por exemplo, no cenário funcional (apresentado na Seção 4.1), o Ministério de Saúde definiu os protocolos de admissão do paciente como regra ao receber um paciente nas UBS e, para conter o avanço da pandemia do COVID19, definiu a aplicação do protocolo *fast-track*.
- **Registro de Requisitos de Software**: significa a especificação, alteração e manutenção de requisitos de software baseados em demandas que um SIS exige. Este caso de uso inicia quando ocorre uma mudança nas funcionalidades de um SIS. Desse modo, o engenheiro de requisito tem que registrar os novos requisitos de um SIS, os quais atenderão às novas demandas, e fazer a manutenção dos requisitos de software existentes.

- **Levantamento de Arquétipos**: caso de uso que faz a procura e identificação de um conjunto de arquétipos que atende ao modelo de dados das novas funcionalidades registrados pelo engenheiro de requisito. Esse caso de uso inicia quando ocorre uma mudança nos arquétipos utilizados em um SIS. Por exemplo, um especialista em arquétipos é responsável por fazer a procura e identificação dos arquétipos que serão usados no SIS.
- **Construção de Formulário com Arquétipo**: inicia-se quando se precisa fazer adições ou modificações na interface gráfica de usuário do SIS, por exemplo: formulário web apresentado na Figura 18. Esse caso de uso atende ao **RF1** (mostrado na Seção 4.2.1).

Figura 28 - Diagrama de casos de uso do cenário funcional mostrado na Seção 4.1.



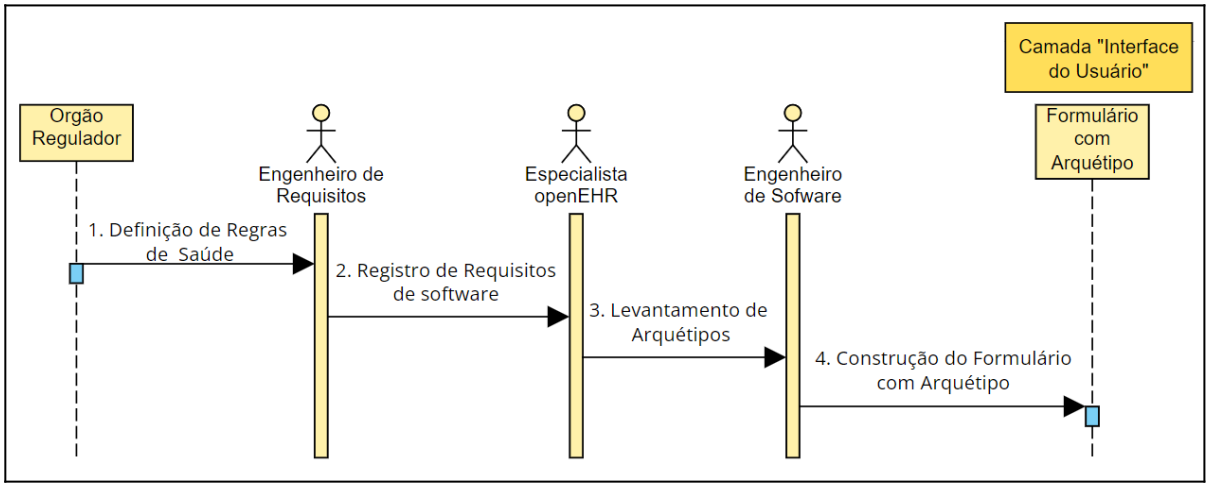
Fonte: O autor (2022).

A Figura 29 mostra o diagrama de sequência do comportamento do cenário funcional. Primeiramente, um órgão regulador define um conjunto de regras de saúde. Baseado nesse conjunto de regras, um engenheiro de requisito registra um conjunto de requisitos de software. Na sequência, um especialista openEHR faz o levantamento de arquétipos que podem ser usados no SIS. Finalmente, um engenheiro de software constrói o componente *Formulário com Arquétipo*.

A Figura 30 mostra, inicialmente, o usuário abrindo a aplicação e requisitando um processamento no *Formulário com Arquétipo*. Esse formulário envia os dados ao Controlador, o qual recebe esses dados e, em seguida, inicia a comunicação com o adaptador. O *Identificador* identifica os arquétipos utilizados nessa requisição e os envia ao Localizador.

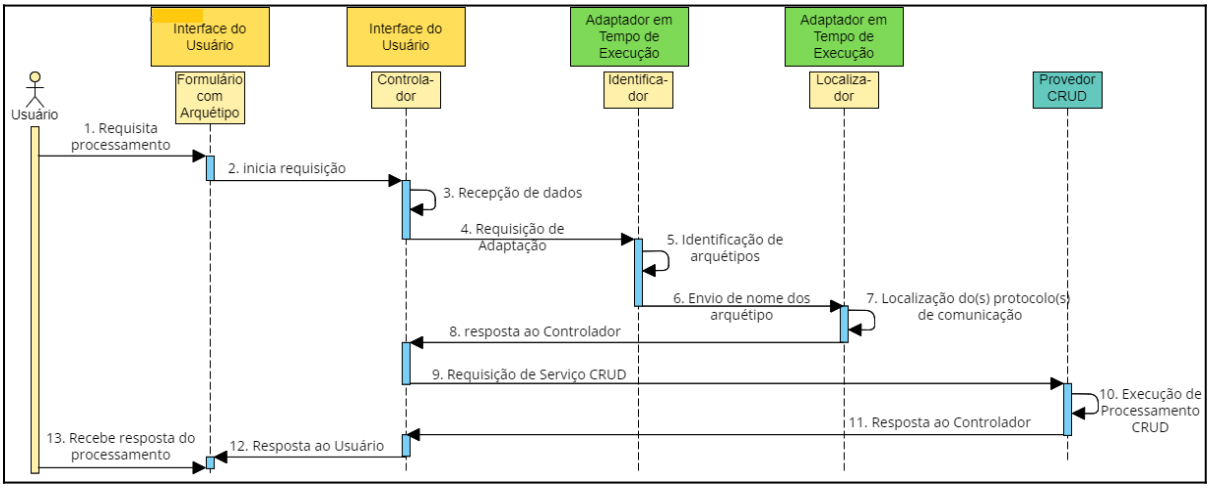
Localizador envia uma resposta ao *Controlador* que inicia uma segunda requisição solicitando o processamento de operações CRUD dos dados enviados pelo usuário. Uma resposta é retornada ao *Controlador* que, finalmente, envia essa resposta para o *Formulário com Arquétipo* para visualização do resultado pelo usuário.

Figura 29 - Diagrama de sequência do comportamento do cenário funcional.



Fonte: O autor (2022).

Figura 30 - Diagrama de sequência do comportamento da arquitetura.



Fonte: O autor (2022).

4.4. Considerações Finais

Este capítulo apresentou uma arquitetura de software adaptativa baseada em arquétipos para sistemas de informação em saúde. Esta arquitetura foi projetada para prover aspectos de adaptabilidade às modificações de funcionalidades resultantes de alterações feitas aos arquétipos utilizados no SIS.

Primeiramente, foram identificados os requisitos para a arquitetura. No que diz respeito aos requisitos funcionais descritos na Seção 4.2, eles são atendidos através da definição dos comportamentos mostrados na Seção 4.3.4. Em adição, o requisito não funcional é atendido por meio do componente *Adaptador em Tempo de Execução*, conforme descrito na Seção 4.3.2, cujos comportamentos também são mostrados na Seção 4.3.4.

Este capítulo também descreveu cada um dos elementos que compõem a arquitetura através da documentação dos níveis de contexto, *container* e de componentes do modelo C4.

Por fim, considerando a discussão sobre os trabalhos investigado na seção 3.4, do Capítulo 3, a arquitetura de software proposta nesta tese diferencia-se desses trabalhos por atender a todos critérios mostrados na Tabela 1. Isto é, trata-se de uma arquitetura de software distribuída em rede, que utiliza serviços web e arquétipos para representação de RES, tem um mecanismo de adaptabilidade em tempo de execução, o qual adapta a arquitetura se houver alterações nos arquétipos utilizados nos SIS.

5 AVALIAÇÃO E RESULTADOS

Este capítulo descreve uma avaliação sobre a adaptabilidade de arquiteturas de software que utilizam arquétipos. Particularmente, o principal objetivo desta avaliação é comparar a arquitetura de software proposta nesta tese com outras arquiteturas em uso no mercado e na Academia e verificar se a arquitetura proposta nesta tese se adapta mediante a mudança no conjunto de arquétipo utilizado na camada do cliente.

A avaliação de arquiteturas de software possui muitos desafios, aos quais diversos trabalhos têm proposto soluções (SOBHY et al., 2021). Uma solução é a avaliação de arquitetura de software baseada em cenários que propõe o desenvolvimento de um conjunto de cenários que concretiza o real significado de um requisito (KAZMAN et al., 2000; FOLMER et al., 2005).

Nesse contexto, o cenário é uma descrição da interação entre alguma fonte (como uma parte interessada) e um sistema de software (RICHARDS, 2015) e, também, ajuda a tirar conclusões sobre a adequação de uma arquitetura proposta em um determinado contexto (BABAR et al., 2019).

Nesta tese, utilizou-se a avaliação de arquitetura de software baseada em cenários com o intuito de mostrar o comportamento do requisito não funcional de adaptabilidade da arquitetura de software proposta. Para realizar essa avaliação, utilizou-se:

- Dois cenários de saúde cujos protocolos são regidos pelo Ministério da Saúde do Brasil e sofreram modificações em todo país devido a pandemia de COVID-19. Esses cenários são: A admissão de paciente em uma UBS e a triagem clínica inicial em clínicas de doação de sangue (também chamadas nesta tese de “hemocentros”).
- Um conjunto de ferramentas do estado da arte e da prática que constroem SIS utilizando arquétipos. As ferramentas utilizadas na avaliação desta tese são: EhrScape (BETTER, 2017), EhrBase (EHRBASE, 2022), MARCIA

(GOMES et al., 2018), Template4EHR (ARAÚJO et al., 2020) e Microservice4EHR (SILVA et al., 2019).

- Por fim, utilizou-se a ferramenta AdaptiveHIS (SILVA et al., 2022), com a qual é possível gerar SIS que utilizam a arquitetura proposta nesta tese.

Importante informar que não foram encontradas na literatura quaisquer ferramentas ou especificações de arquiteturas de software que utilize arquétipos e tenha adaptabilidade. Devido a isso, as ferramentas utilizadas na avaliação tiveram como critério de escolha o uso de (i) arquétipos para construção de arquitetura de software de Saúde, (ii) arquiteturas cliente-servidor ou computação em nuvem, (iii) licença gratuita, (iv) disponibilição de documentação na Internet.

Para um melhor entendimento, é necessário definir alguns conceitos utilizados nessa avaliação: “sistemas de informação em saúde” ou “SIS”, “arquitetura de software” ou “arquitetura”, e “ferramentas”. SIS é um software ou aplicativo que dá suporte às atividades administrativas do domínio de saúde, por exemplo: um aplicativo web de cadastro de novos pacientes em um hospital).

O conceito de “arquitetura de software” é a organização, estrutura e relacionamento dos componentes que compõem um SIS. “Ferramentas” referem-se aos softwares ou aplicativos utilizados para dar suporte à construção de SIS. Essas ferramentas são usadas para gerar códigos, os quais materializam a estrutura (componentes) das arquiteturas de software, por exemplo: Template4EHR, Microservice4EHR e AdaptiveHIS.

O escopo da avalição é verificar os componentes do *backend do SIS* (explicado na Seção 4.3.2.2), mediante alteração dos arquétipos do *frontend do SIS* (explicado na Seção 4.3.2.1). Para executar essa avaliação, três etapas foram realizadas:

- Na primeira etapa, o autor desta tese construiu um total de dezoito SIS com as ferramentas EHRScape, EHRBase, Template4EHR, Microservice4EHR e AdaptiveHIS. Para essa construção, considerou-se um conjunto de arquétipos (obtidos no CKM), os quais foram utilizados nas interfaces gráficas (GUI) desses SIS. Essas GUI foram desenvolvidas para atender algumas funcionalidades do SIS, exigidas antes da pandemia do COVID-19.

- Na segunda etapa, o conjunto de arquétipos utilizados nesses SIS foi modificado no *frontend do SIS*. Desta forma, as GUI desses SIS foi alterada baseada nesse novo conjunto de arquétipos. Essa nova GUI foi desenvolvida para atender as novas funcionalidades do SIS, exigidas depois da pandemia do COVID-19. Como parte dessa avaliação, a camada do servidor não foi alterada com o propósito de verificar a adaptação da arquitetura de software às modificações aplicadas ao conjunto de arquétipos utilizados na GUI desses SIS.
- Na terceira etapa, analisou-se o comportamento dos componentes do *backend do SIS*. Nessa análise, verificou-se se os SIS executam suas funcionalidades com erros, falhas ou inconsistências. Por fim, com o auxílio da métrica *level system adaptability* (LSA) (Perez-Palacin et al., 2014) foi possível medir a adaptabilidade da arquitetura de software dos SIS avaliados.

A métrica LSA, citada na terceira etapa acima, propõe um cálculo de adaptabilidade baseada em uma divisão cujo numerador é o número de componentes inicialmente usados em uma arquitetura de software, e o denominador é o número de componentes disponíveis na mesma arquitetura de software após mudanças estruturais. Na métrica LSA, se o resultado for uma fração entre zero e 0.9, significa que a arquitetura é adaptável, se o resultado for o número um ou mais, não há adaptação.

Essa avaliação (de três etapas) foi aplicada nos SIS de três centros de saúde (os quais são descritos na seção 5.1). Com o intuito de alcançar o objetivo estabelecido, essa avaliação deve responder às seguintes Questões de Pesquisa (QP):

- **QP1**: A arquitetura de software proposta utiliza arquétipos como artefato computacional para representação de seus RES?
- **QP2**: A arquitetura de software proposta identifica os arquétipos utilizados na camada do cliente em tempo de execução?
- **QP2**: A arquitetura de software proposta executa uma adaptação, na camada do servidor, referente aos arquétipos utilizados na camada do cliente?

Este capítulo está dividido sete seções, conforme descrito a seguir. A Seção 5.1 mostra o planejamento realizado nessa avaliação. A Seção 5.2 apresenta como

foi realizada a construção dos dezoito SIS. A Seção 5.3 descreve a mudança do conjunto de arquétipos utilizados nesses SIS. A Seção 5.4 exibe os resultados encontrados. As seções subsequentes mostram as respostas das questões de pesquisa, as ameaças à validade da avaliação e as considerações finais.

5.1 Planejamento

Sobre a **infraestrutura** computacional, foi utilizado uma estação de trabalho em uma rede local com a seguinte configuração: processador de núcleo i7 1.8 GHz 8550U quad-core, 16 GB de RAM DDR4, sistema operacional Win 10 PRO e memória flash 256 GB. Também, utilizou-se uma conta gratuita em Heroku (SALESFORCE, 2022), uma plataforma disponível na Internet que viabilizou a execução da arquitetura proposta utilizando a computação em nuvem. Os critérios utilizados para escolha do Heroku basearam-se na existência da opção de conta gratuita e porque o autor desta tese tem familiaridade com essa plataforma.

Sobre as **ferramentas** computacionais, foi instalado na estação de trabalho local o seguinte conjunto de ferramentas: EHRScape, EHRBase, MARCIA, Template4EHR, Microservice4EHR, AdaptiveHIS. Essas ferramentas foram escolhidas porque permitem a construção de SIS utilizando arquétipos, são arquitetura cliente-servidor e tem opções não proprietárias.

Além disso, o GIT (2022), Heroku e um banco de dados PostgreSQL foram instalados no computador local e na plataforma Heroku. Esse banco de dados foi escolhido por ser gratuitamente disponível no Heroku e por não possuir restrições quanto às atividades executadas nessa avaliação.

Os arquétipos utilizados nessa avaliação estão disponíveis no site do CKM e, opcionalmente, também disponível no endereço eletrônico do site do projeto de pesquisa dessa tese (isto é, *https://www.cin.ufpe.br/~maps3/ahsa*). Nesta avaliação, não se verificou se os arquétipos estão validados pela openEHR. Somente considerou-se estar disponível no site do CKM. Assim se procedeu porque essa validação do arquétipo pela openEHR não impacta na avaliação da adaptabilidade da arquitetura proposta.

Para tornar a arquitetura proposta executável, um conjunto de *Provedor CRUD* foi construído, o qual foi implantado na plataforma Heroku e estão disponíveis na Internet. Por exemplo, é possível acessar alguns deles pelos seguintes endereços eletrônicos: *https://blood-pressure-v2.herokuapp.com*, *https://respiration-v2.herokuapp.com*, *https://individual-personal-v1.herokuapp.com*, *https://admission-v0.herokuapp.com*. Devido ao perfil da conta gratuita do Heroku a qual tem recursos limitados de armazenamento, os dados dessa avaliação não foram persistidos. Detalhes de implementação de um *provedor CRUD* estão disponíveis no Anexo I desta tese e, também, no site do projeto: *https://www.cin.ufpe.br/~maps3/ahsa*.

Sobre os **cenários de saúde**, foi utilizado o protocolo de admissão de paciente em uma UBS e o protocolo de triagem clínica inicial em dois hemocentros. Nessa avaliação, foram considerados duas versões desses protocolos. A primeira versão, refere-se aos protocolos exigidos antes da pandemia do COVID-19. A segunda versão, refere-se aos protocolos exigidos depois dessa pandemia.

Sobre a primeira versão do protocolo de admissão de paciente em uma UBS, os SIS que atendiam a esse protocolo eram compostos por um conjunto de dados que podiam ser representados pelos arquétipos `admission.v0` e `individual_personal.v0`, por exemplo. Esses arquétipos contêm dezenas de atributos de dados referentes ao paciente, por exemplo: nome, data de nascimento, sexo, endereço, etc. Porém, devido à pandemia, o governo brasileiro adicionou o protocolo *fast-track*, que inclui outras informações do paciente, como: sintomas tipo influenza, tosse, febre, dificuldade para respirar, etc. Para apoiar essa modificação, novos requisitos e funcionalidades foram adicionados aos SIS, cujos dados podem ser representados pelos arquétipos: `story.v1`, `body_temperature.v2`, `symptom_sign-cvid.v0`, `travel_history.v0` e `health_risk-covid.v0`.

Sobre a primeira versão da triagem clínica inicial em um hemocentro, os SIS que atendiam a esse protocolo eram compostos por um conjunto de dados que podiam ser representados por doze arquétipos: `person.v0`, `height.v1`, `body_weight.v1`, `blood_pressure.v2`, `lab_test-blood_glucose.v1`, `pathology_test.v1`, `pulse.v1`, `smoking_use_summary.v1`, `alcohol_use_summary.v1`, `family_history.v2`, `medicine_substance.v0`, `clinic_synopsis.v1`.

Esses arquétipos contêm centenas de atributos de dados sobre informações pessoais, testes hematológicos, consumo de álcool, uso contínuo de medicamentos, histórico familiar, etc. Entretanto, devido à pandemia, o governo brasileiro modificou esse protocolo.

Candidatos que foram infectados ou tiveram contato com pacientes infectados com o vírus Corona-19, passaram a ser considerados incapazes de doar sangue (temporariamente). Para dar suporte a essa nova diretriz, novos requisitos e funcionalidades foram adicionados aos SIS, aos quais também foram adicionados novos arquétipos, por exemplo: `story.v1`, `travel_history.v0`, `health_risk-covid.v0`. Nessa avaliação, foram avaliados os SIS dos hemocentros dos estados de Sergipe (Hemose) e Pernambuco (Hemope). Nesse cenário dos hemocentros, também foi avaliada a retirada de arquétipos dos SIS. Os dois arquétipos escolhidos para serem retirados foram: `body_weight.v1` e `lab_test-blood_glucose.v1`.

O critério utilizado para essa retirada, é a possibilidade de o doador não consentir na divulgação desses dados por se tratar de dados pessoais, conforme descrito na Lei Geral de Proteção de Dados (G. BRASILEIRO, 2018). Importante informar que outros arquétipos também representam dados pessoais, os quais podem ser retirados em outras avaliações futuras.

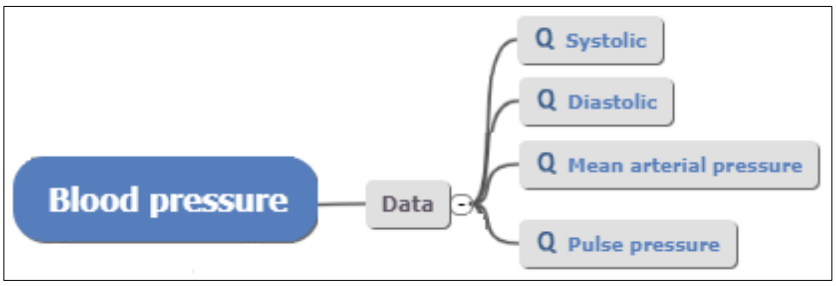
5.2 Primeira Etapa: Construção dos SIS

Um total de dezoito SIS foram gerados com as ferramentas descritas na Seção 5.1. Inicialmente, esses SIS foram construídos considerando os protocolos exigidos antes da pandemia. Um mesmo conjunto de arquétipos foi utilizado para construir todos esses SIS, conforme explicado a seguir. No cenário da UBS, foram utilizados os arquétipos `person.v0` e `admission.v0`, mostrados nas Figuras 2 e 17, respectivamente.

No cenário dos hemocentros, utilizou-se um outro conjunto de arquétipos nos SIS as clínicas de doação de sangue. Para exemplificação, a Figura 31 mostra o arquétipo `blood_pressure.v2`, o qual foi utilizado nesta avaliação. Esse arquétipo

é composto pelos atributos sistólica, diastólica, pressão arterial, pulso, interpretação clínica e comentários gerais.

Figura 31 - Arquétipo *Blood Pressure*.



Fonte: *Clinical Knowledge Manager*.

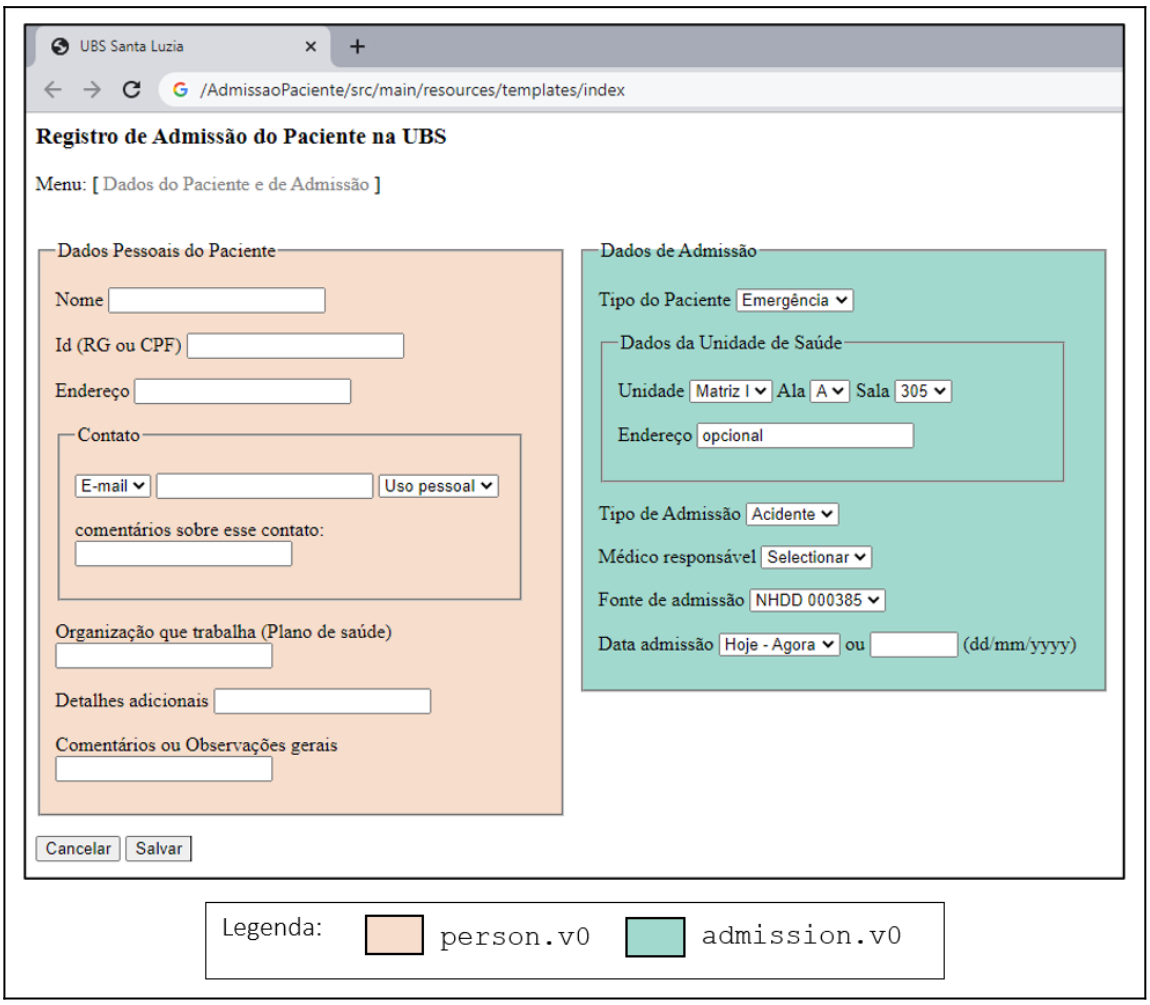
Para o melhor entendimento do funcionamento da camada do cliente, algumas ilustrações de GUI construídas com esses arquétipos são mostradas. Além disso, diagramas do nível de componentes do modelo C4 (BROWN, 2022) são usados para compreensão do fluxo de dados e da visualização das arquiteturas de software geradas por ferramenta. A seguir, a avaliação feita no cenário da UBS é descrita.

5.2.1 Cenário da UBS

Para representar os dados contidos no registro de admissão do paciente em uma UBS, antes da pandemia, os arquétipos `person.v0` e `admission.v0` foram usados. A Figura 32 exemplifica uma GUI de um SIS web construída com esses dois arquétipos. Nessa figura, o título "*Registro de Admissão do Paciente na* UBS" e um menu de opções são mostrados no topo da página.

Essa GUI também mostra campos referentes aos dados do paciente e de admissão, os quais foram construídos com os arquétipos `person.v0` e `admission.v0`, respectivamente. Na aba *Dados Pessoais do Paciente* da Figura 32, o usuário do SIS pode registrar o nome, id, endereço, contato, a organização que o paciente trabalha, detalhes adicionais, comentários ou observações gerais sobre o paciente.

Figura 32 - Uma GUI construída com os arquétipos `person.v0` e `admission.v0`.



Fonte: O autor (2022).

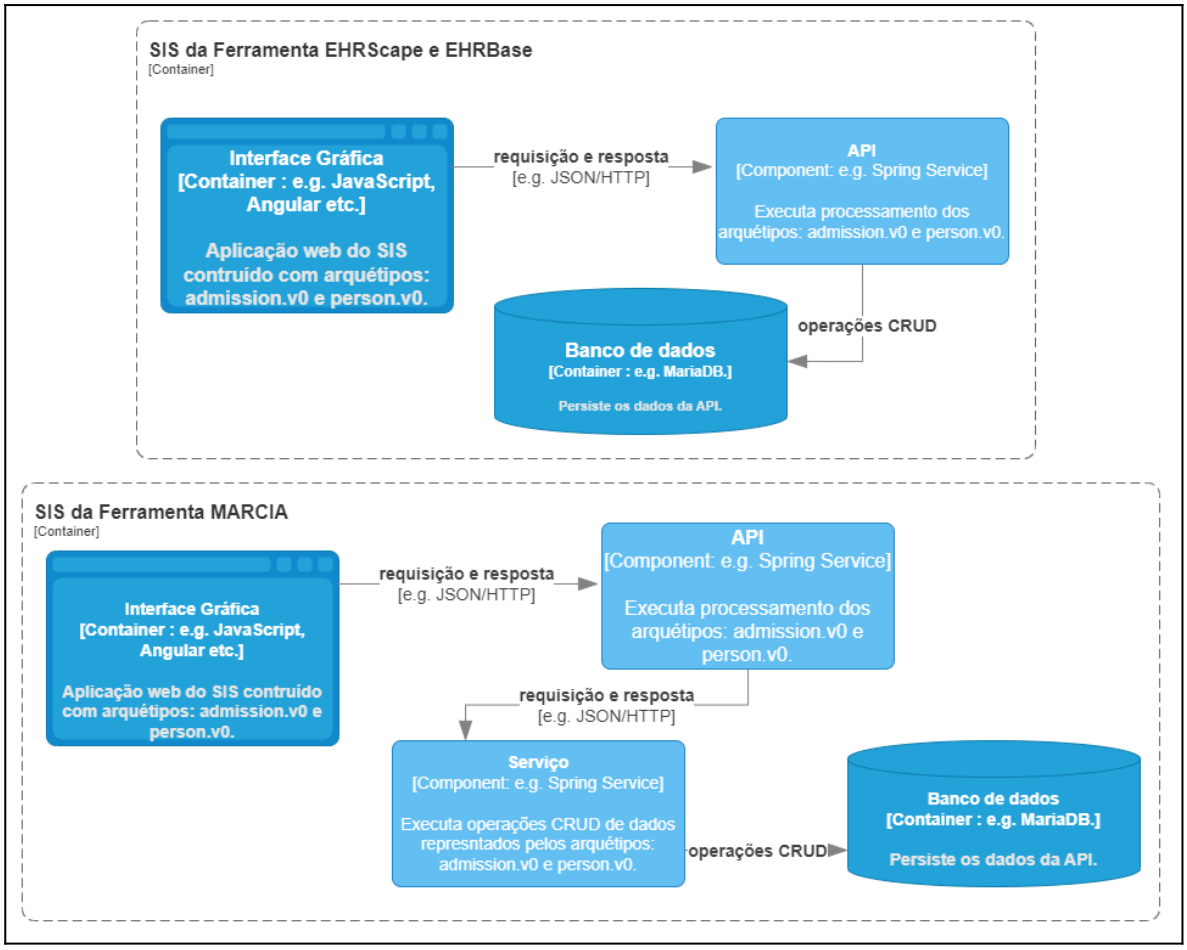
Já na aba *Dados de Admissão* dessa figura, o usuário do SIS pode registrar o tipo de paciente que está sendo admitido (por exemplo: emergência), os dados da unidade de saúde que o paciente será alocado (por exemplo: Unidade Matriz I, Ala A, Sala 305). Com relaçao ao tipo de admissão (por exemplo: acidente), o médico responsável por aquela admissão, o código médico referente a fonte de admissão (por exemplo: NHDD 000385) e a data da admissão que está sendo registrada.

Nesta seção, utiliza-se o diagrama de componentes para apresentar as arquiteturas de software dos SIS gerados para atender ao cenário da UBS. Essas arquiteturas são compostas pela camada do cliente e do servidor. Neste capítulo, a camada do servidor pode ser um servidor local ou um nó da computação em nuvem

(conceito explicado no Capítulo 2). Os componentes contidos na camada do servidor podem ser executados em uma rede local ou na Internet.

A Figura 33 mostra que os SIS e suas respectivas arquiteturas de software foram gerados para processar dados dos arquétipos `admission.v0` e `person.v0`. Isto significa que, nessa etapa, considerou-se os requisitos de software antes da pandemia do COVID.

Figura 33 - Diagrama das arquiteturas de software geradas pelas ferramentas EhrScape, EhrBase, e MARCIA, para o cenário da UBS.

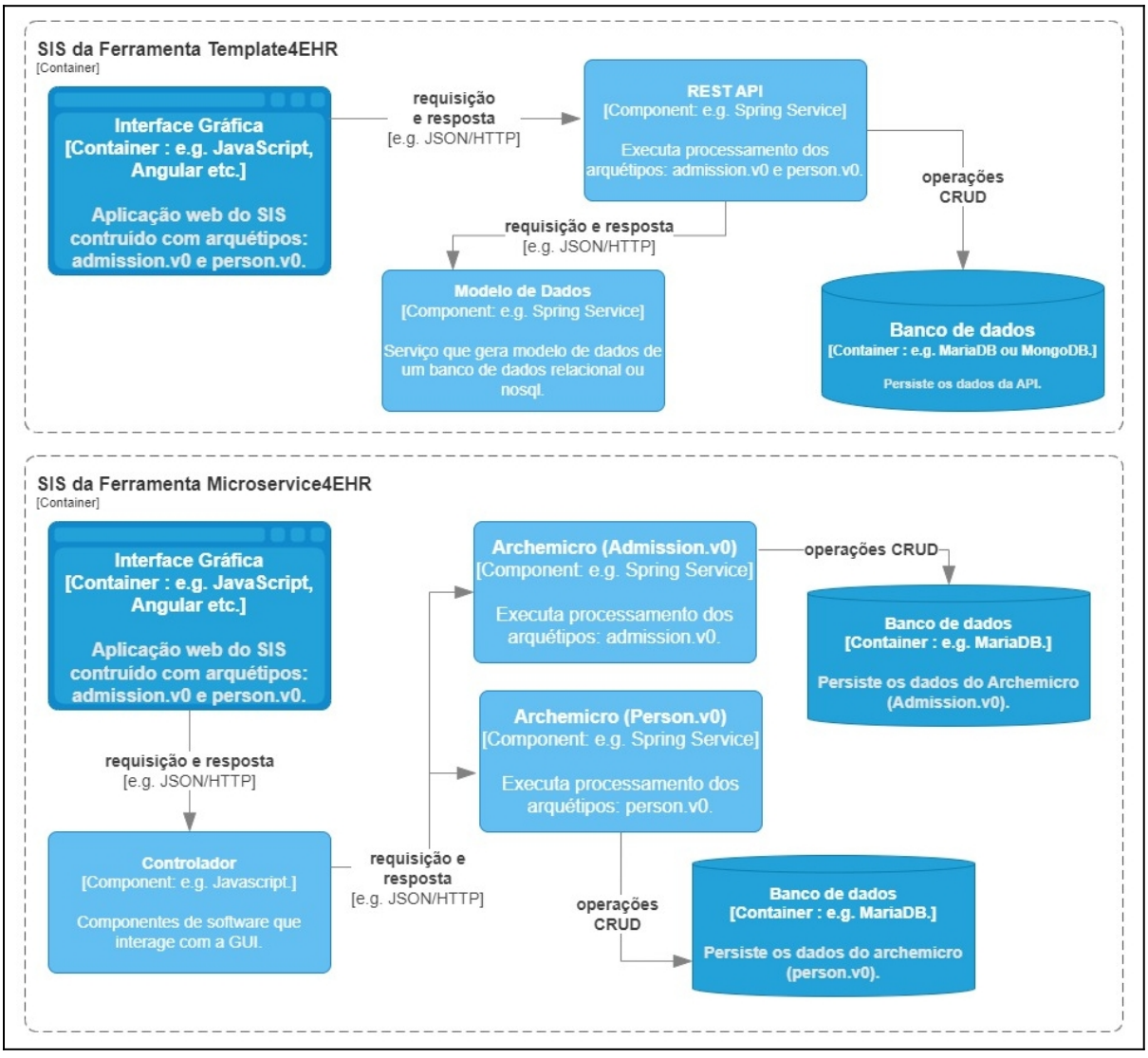


Fonte: O autor (2022).

A arquitetura de software dos SIS gerados pelas ferramentas EhrScape e EhrBase são similares. Cada SIS possui um componente que representa uma GUI. Esse componente faz requisição de processamento a uma API na camada do servidor. Essa API executa operações CRUD em um banco de dados que, por sua

vez, persiste dados dos dois arquétipos utilizados no SIS. A arquitetura de software do SIS gerado pela ferramenta MARCIA difere-se da anterior na camada do servidor. Isto é, nessa camada, a API executa as operações CRUD por meio de um Serviço, o qual interage com um banco de dados. A Figura 34 exibe duas arquiteturas de software.

Figura 34 - Diagrama das arquiteturas geradas pelas ferramentas Template4EHR e Microservice4EHR, no cenário da UBS.



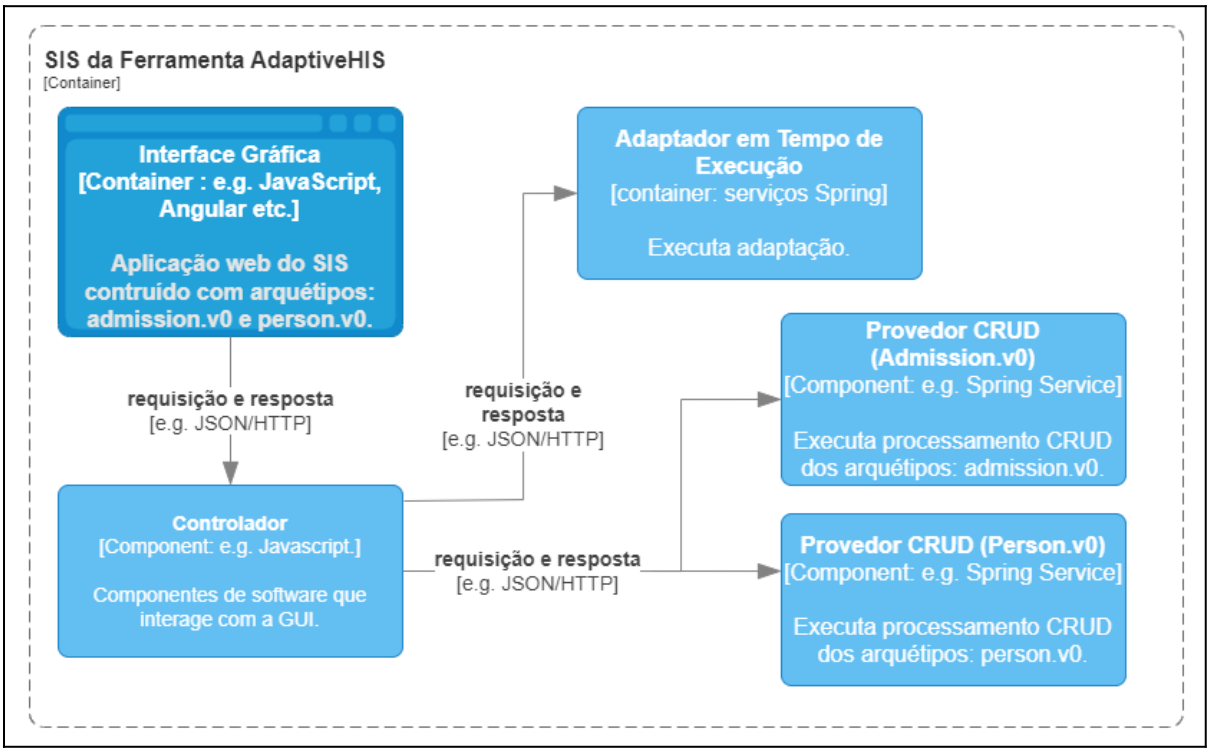
Fonte: O autor (2022).

A primeira é do SIS gerado pela ferramenta Template4EHR, cuja arquitetura tem o componente GUI na camada do cliente a qual interage com uma REST API na

camada do servidor. Essa API interage com o componente Data Model para definição do modelo de dados a ser utilizado no banco de dados e, posteriormente, interage com um banco de dados para persistência dos arquétipos. A segunda arquitetura é descrita como segue. A arquitetura do SIS gerado pela Microservice4EHR tem os componentes GUI e *Connector*, na camada do cliente. Na camada do servidor, ela difere das demais arquiteturas porque possui dois *archemicros* (conceito explicado no Capítulo 3). Cada *archemicro* executa as operações CRUD de seu respectivo arquétipo e, também, interage com seu respectivo banco de dados.

A Figura 35 mostra a arquitetura de software de um SIS gerado pela ferramenta AdapiveHIS. Essa arquitetura é proposta nesta tese. A camada do cliente é composta por dois componentes, a *Formulário com Arquétipo* (representada na figura pela GUI) e o *Controlador*, ambos construídos utilizando os arquétipos `admission.v0` e `person.v0`. O *Adaptador em Tempo de Execução* adaptou dois *Provedores CRUD* referente aos arquétipos `admission.v0` e `person.v0`.

Figura 35 - Diagrama da arquitetura gerada pela AdaptiveHIS, no cenário da UBS.

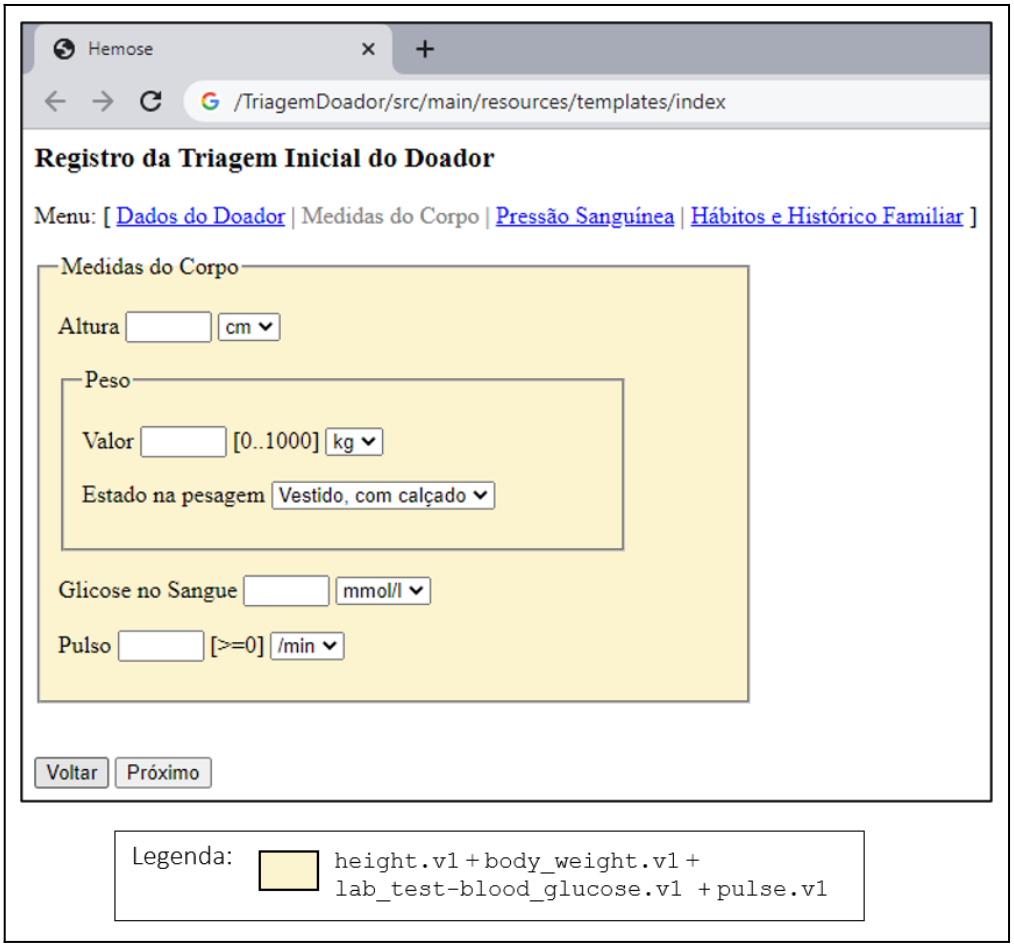


Fonte: O autor (2022).

5.2.2 Cenário dos Hemocentros

Para representar os dados no registro de triagem clínica inicial em um hemocentro, antes da pandemia, utilizou-se os arquétipos `person.v0`, `height.v1`, `body_ weight.v1`, `blood_pressure.v2`, `lab_test-blood_glucose.v1`, `pathology_ test.v1`, `pulse.v1`, `smoking_use_summary.v1`, `alcohol_use_summary.v1`, `family_ history.v2`, `medicine_substance.v0` e `clinic_synopsis.v1`. A Figura 36 ilustra uma GUI de um SIS web construída com os arquétipos `height.v1`, `body_weight.v1`, `lab_test-blood_glucose.v1` e `pulse.v1`.

Figura 36 - Uma GUI construída com os arquétipos `height.v1`, `body_weight.v1`, `lab_test-blood_glucose.v1` e `pulse.v1`.

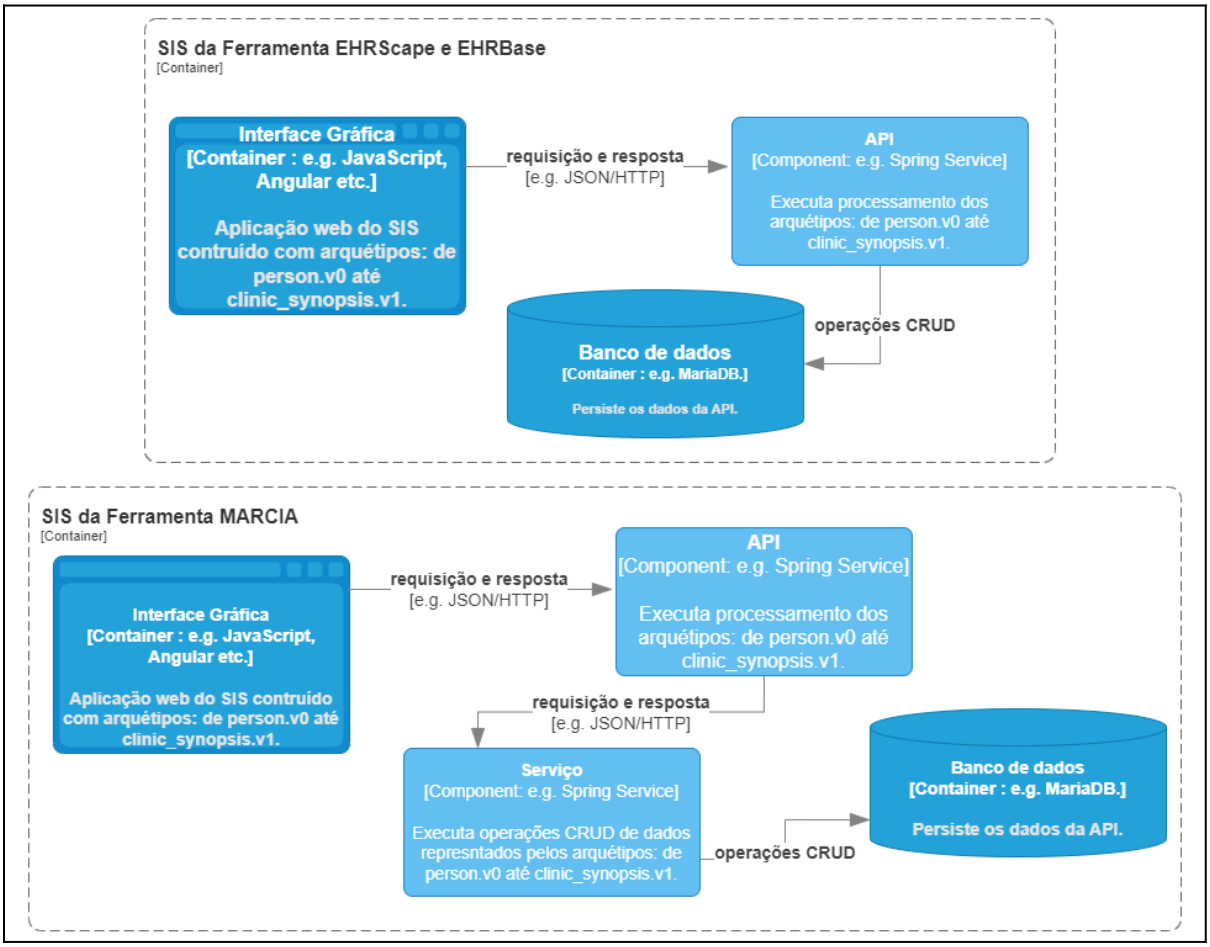


Fonte: O autor (2022).

Essa figura mostra o título “*Registro da Triagem Inicial do Doador*”, um menu de opções e o conteúdo da opção de menu “*Medidas do Corpo*”. Esse SIS dividiu o protocolo de triagem clínica inicial em um formulário web de quatro partes: *Dado do Doador*, *Medidas do Corpo*, *Pressão Sanguínea* e, por fim, *Hábitos e Histórico Familiar*. O acesso a essas partes é disponibilizado no menu.

A parte do SIS referente às *Medidas do Corpo* foi construída utilizando vários arquétipos, conforme descrito a seguir. Para representar a altura do doador e sua unidade de medida (cm) foi utilizado o arquétipo `height.v1`.

Figura 37 - Diagrama das arquiteturas geradas por EhrScape, EhrBase e MARCIA, no cenário dos hemocentros.



Fonte: O autor (2022).

Para representação do peso, da unidade de medida (kg) e do estado na pesagem (‘vestido, com calçado’) foi utilizado o arquétipo `body_weight.v1`. Para

representar os dados de glicose no sangue e sua unidade de medida (mmol/l), e os dados de pulso e sua unidade de medida (‘/min’) foram utilizados os arquétipos `lab_test-blood_glucose.v1` e `pulse.v1`, respectivamente.

A Figura 37 mostra as arquiteturas de software dos SIS gerado pelas ferramentas EhrScape, EhrBase e MARCIA para o cenário dos Hemocentros. Essas arquiteturas são semelhantes às apresentadas no cenário da UBS (Figura 33). Contudo, os componentes foram construídos para atender ao processamento de outro conjunto de arquétipos, os quais são descritos na legenda da Figura 37.

A Figura 38 mostra as arquiteturas de software dos SIS gerado pelas ferramentas Template4EHR e Microservice4EHR para o cenário dos Hemocentros.

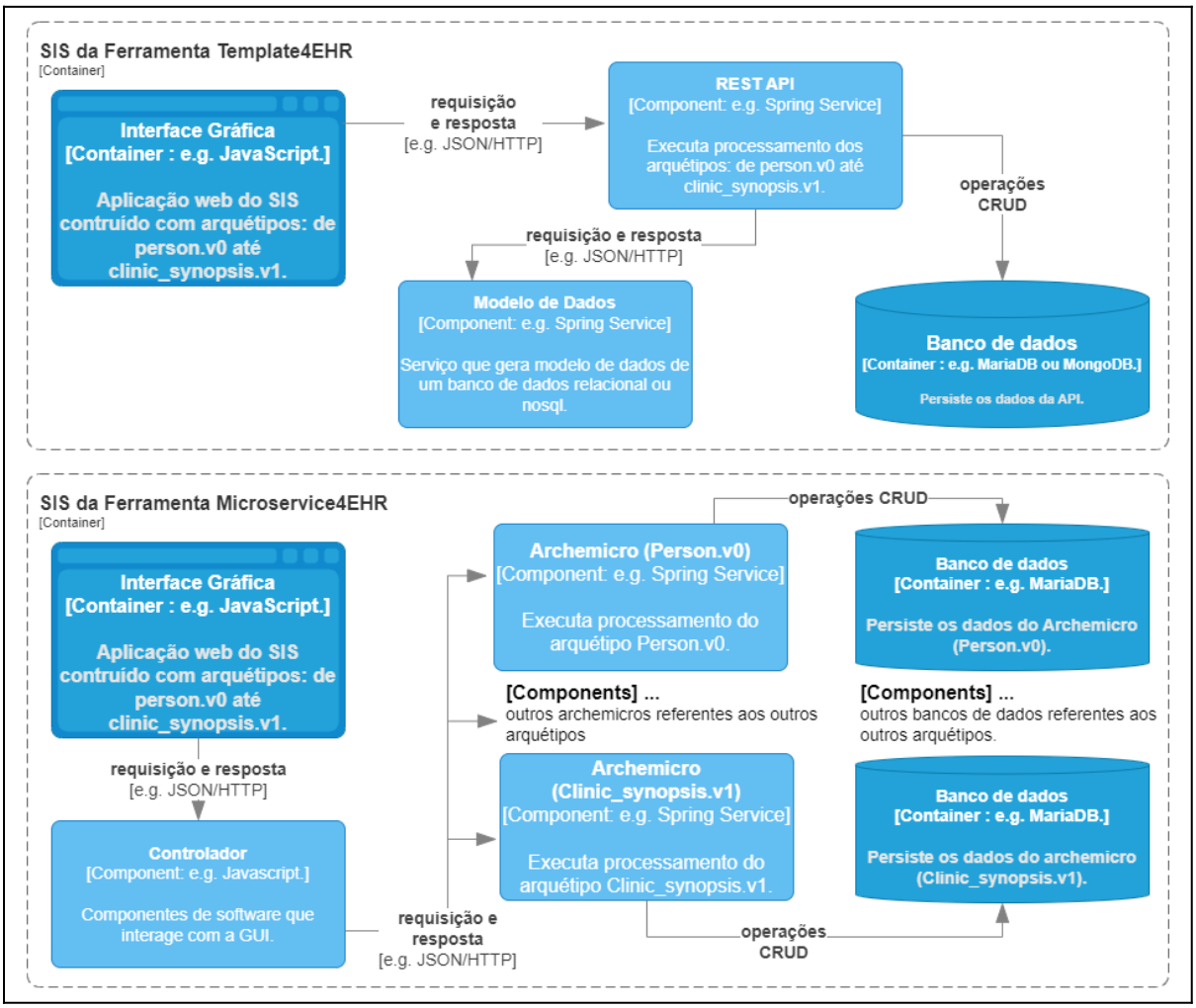
A arquitetura gerada pela Template4EHR é semelhante à arquitetura também gerada pela Template4EHR para o cenário da UBS (na Figura 34). Porém, os componentes foram construídos para atender ao processamento de outro conjunto de arquétipos, os quais são descritos na Figura 38.

Há uma diferença na arquitetura do SIS gerado pela Microservice4EHR, a qual instanciou um conjunto maior de *archemicros* e *databases* (doze cada). Esse aumento é esperado porque o número desses componentes é proporcional ao número de arquétipos utilizados no SIS.

A Figura 39 mostra a arquitetura de software dos SIS gerado pela ferramenta AdaptiveHIS, para o cenário dos Hemocentros. As interfaces gráficas e o *Adaptador em Tempo de Execução*) são similares à arquitetura gerada pela AdaptiveHIS apresentada no cenário da UBS, ilustrada na Figura 33. O que difere é a identificação do conjunto de arquétipos utilizados (feita por *identificador*), a localização de componentes de software (feita pelo *localizador*) e a interação com os componentes adaptados.

No atual cenário (hemocentros), a arquitetura instanciou doze *provedores CRUD*. Esse comportamento é também esperado para essa arquitetura para executar o processamento de dados referente ao conjunto de arquétipos utilizados no SIS, sem erros ou falhas.

Figura 38 - Diagrama das arquiteturas geradas pelas ferramentas Template4EHR e Microservice4EHR, no cenário dos hemocentros.



Fonte: O autor (2022).

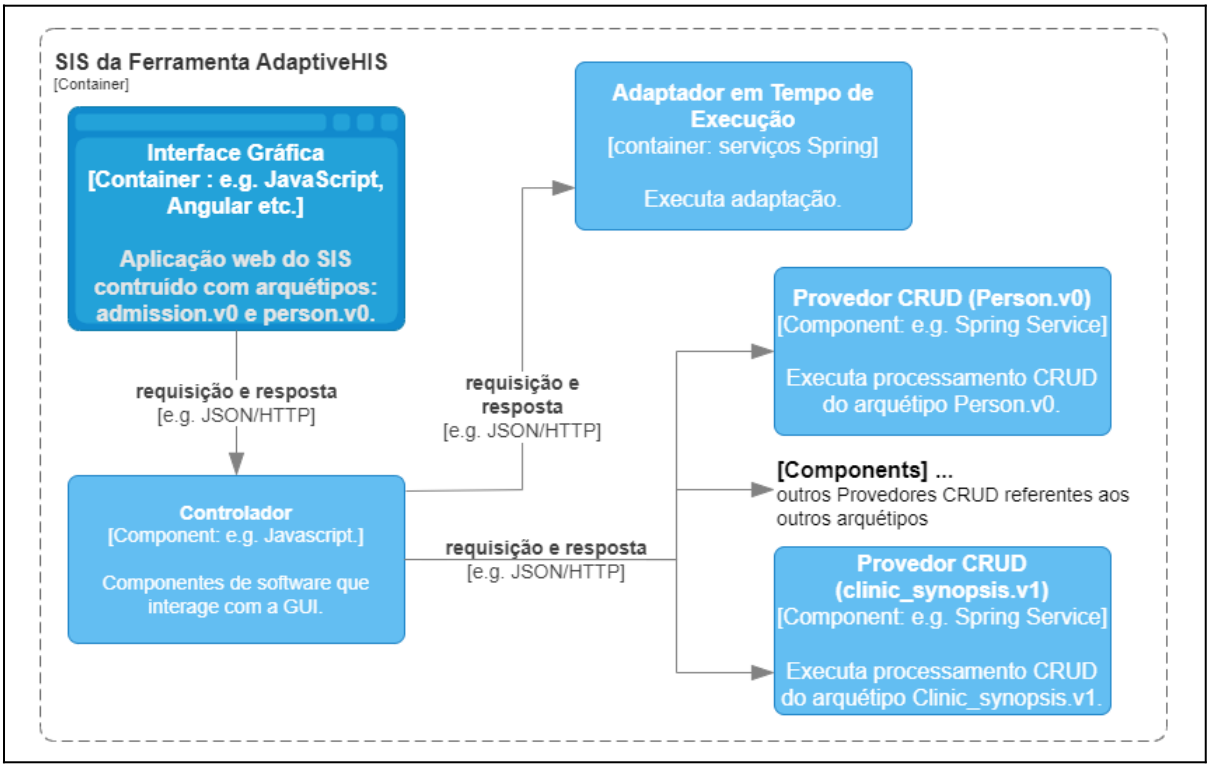
A seguir, alterações no conjunto de arquétipos usados pelos SIS são mostradas. Essa alteração é para que esses SIS atendam aos novos protocolos impostos pelo Ministério da Saúde do governo brasileiro, devido a pandemia do COVID-19.

5.3 Segunda e Terceira Etapa: Mudança de Arquétipos e Análise

Nesta tese, mudança de arquétipos é a alteração do conjunto de arquétipos utilizados na camada do cliente do conjunto de SIS avaliados nessa tese. Essa etapa é realizada por um engenheiro de software. O caso de uso *Construção de Formulário com Arquétipo* ilustrado nas Figuras 28 e 29 (na Seção 4.3.6) exemplifica essa etapa. Ao finalizar

essa mudança, esse conjunto de SIS foi executado e o comportamento de cada arquitetura foi analisado.

Figura 39 - Diagrama da arquitetura gerada pela AdaptiveHIS, no cenário dos hemocentros.



Fonte: O autor (2022).

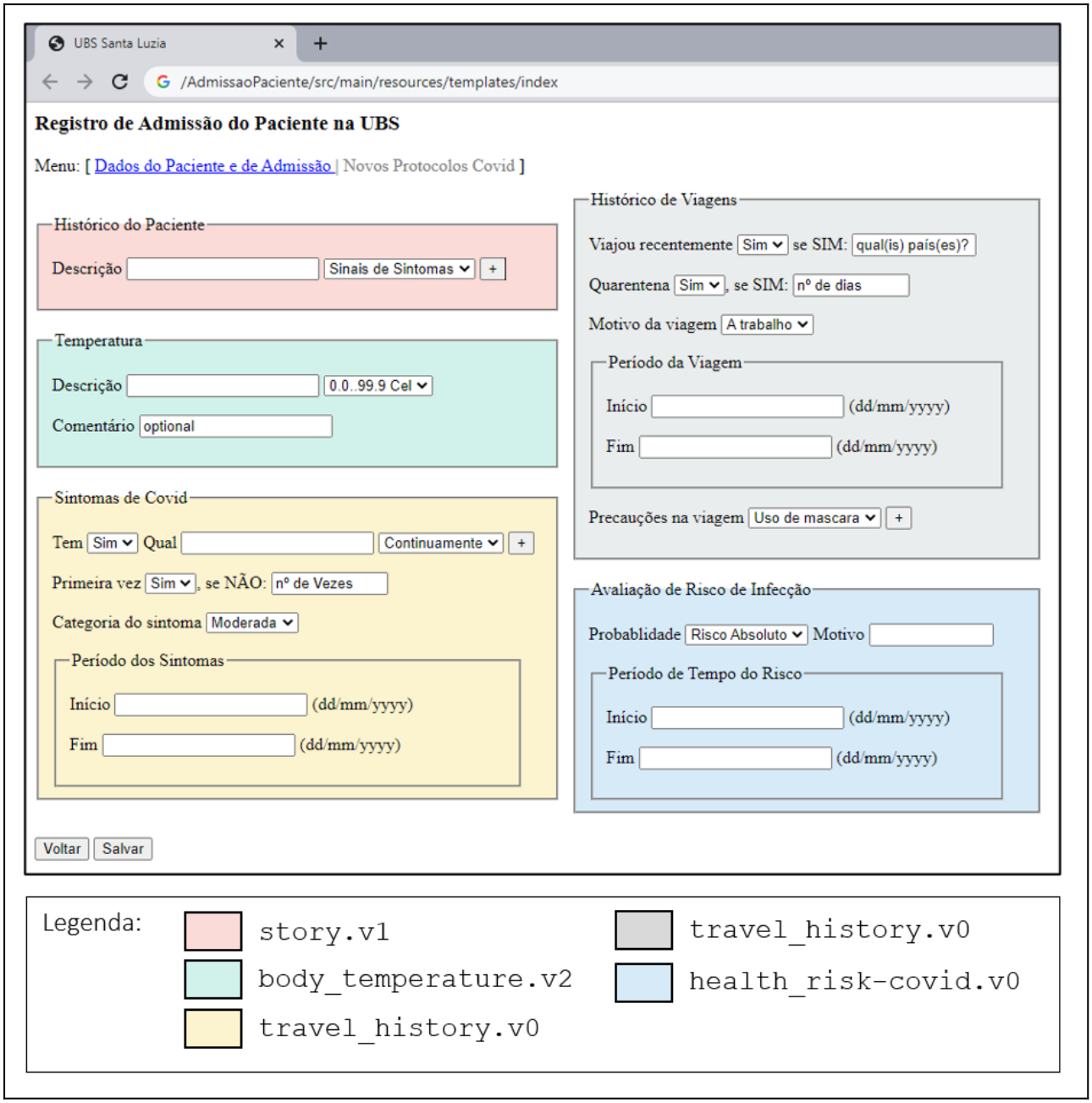
No Cenário da UBS, para atender ao protocolo *fast-track*, uma nova GUI do SIS foi construída com os arquétipos: `story.v1`, `body_temperature.v2`, `symptom_sign-cvid.v0`, `travel_history.v0` e `health_risk-covid.v0`. A Figura 40 mostra uma nova versão do SIS mostrado na Figura 32.

Nessa nova versão, a opção *Novos Protocolos Covid* foi adicionada no menu, cujo conteúdo é composto por *Histórico do Paciente*, *Temperatura*, *Sintomas de Covid*, *Histórico de Viagens* e *Avaliação de Risco de Infecção*. Cada um desses campos foi construído com base no seu respectivo arquétipo, conforme ilustrado na legenda da Figura 40.

No cenário dos hemocentros, modificações foram feitas para conformidade com o novo protocolo que invalida a doação de um candidato que foi infectado ou teve contato com pacientes infectados com Covid-19. Para dar suporte a esse novo

protocolo, novas funcionalidades foram adicionadas aos SIS cujos dados são representados pelos arquétipos `story.v1`, `travel_history.v0`, `health_risk-covid.v0`.

Figura 40 - Nova versão do SIS da UBS.



Fonte: O autor (2022).

A Figura 41 mostra uma nova versão do SIS mostrado na Figura 36. Então, esse formulário de triagem inicial agora possui cinco partes. Isto é, a opção *Novos Protocolos Covid* foi adicionada no menu, através do qual o usuário tem acesso ao registro de *Histórico do Paciente*, *Histórico de Viagens* e *Avaliação de Risco de Infecção*.

Uma outra mudança foi feita na aba *Medidas do Corpo*, a qual é mostrada na parte inferior da Figura 41. Para avaliar o comportamento das arquiteturas de software com a retirada de arquétipos, retirou-se os campos *Glicose no Sangue* e *Pulso* (ilustrada na Figura 36). Desta forma, a nova versão da aba *Medidas do Corpo* não utiliza os arquétipos `lab_test-blood_glucose.v1` e `pulse.v1`.

Após efetuar essas modificações, uma requisição de persistência de dados foi solicitada e o comportamento das arquiteturas foi observado. É importante informar que, nessa avaliação, nenhuma modificação foi feita nos componentes de software na camada do servidor. Essa abordagem busca averiguar qual arquitetura seria adaptável às mudanças feitas no conjunto de arquétipos utilizados na camada do cliente.

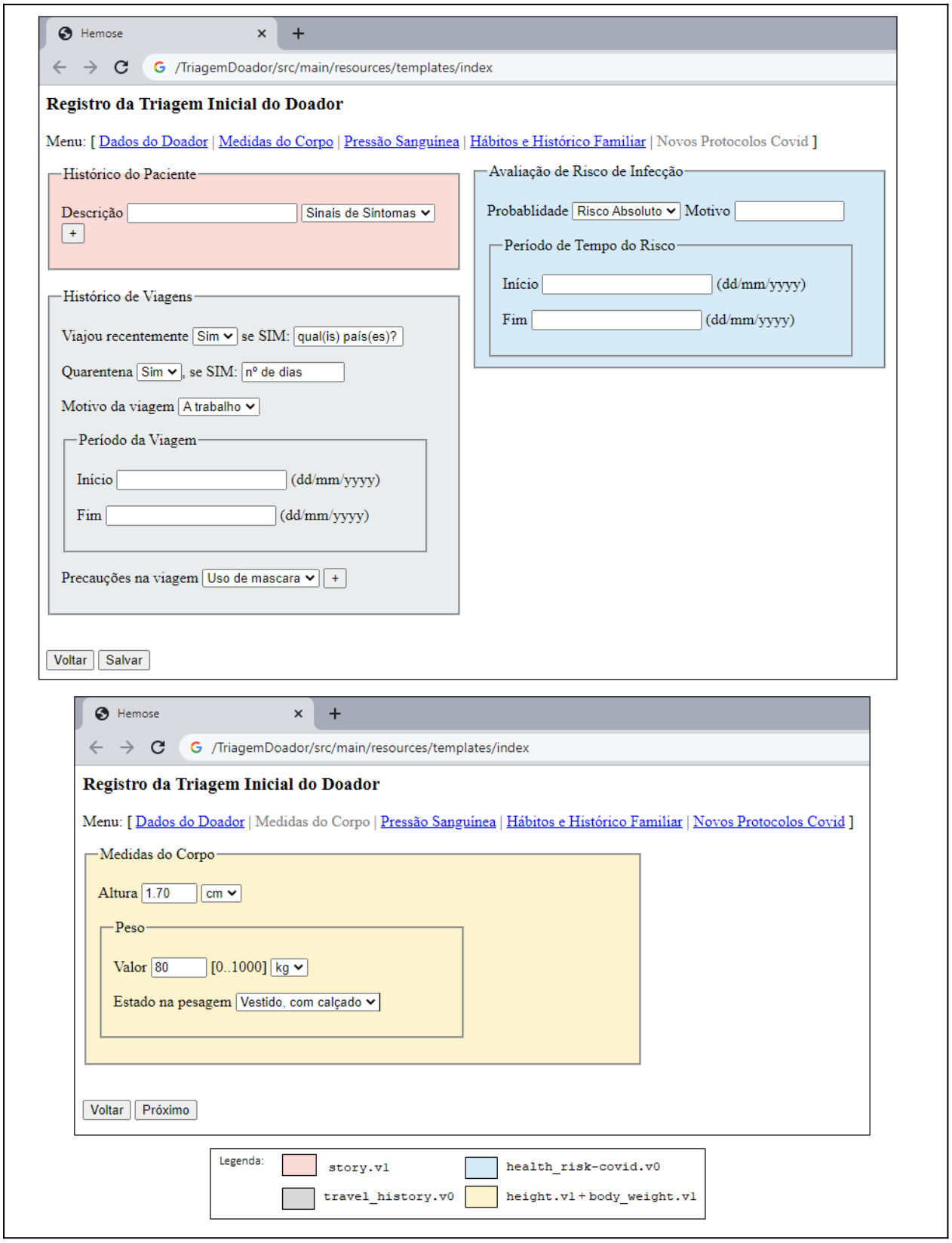
Então, após análise dos comportamentos das arquiteturas dos SIS, os resultados são apresentados a seguir.

5.4 Resultados

Esta seção mostra os resultados obtidos na avaliação proposta nesta tese. Para compreender as figuras apresentadas nesta seção, é importante saber que os nomes das ferramentas (mostrados nas figuras dessa seção) representam as arquiteturas de software gerada por elas.

Nesta tese, o conceito de “erros, falhas ou inconsistências” basea-se em três situações: (i) se um arquétipo não teve o seu dado processado; (ii) se a execução do SIS foi interrompida; e, por fim, (iii) se o SIS processa os dados com alguma falha ou inconsistência (por exemplo: a não persistência dos dados de um arquétipo ou a persistência de dados nulos ou vazios de um arquétipo que não é utilizado no SIS).

Figura 41 - Nova versão do SIS dos hemocentros.



Fonte: O autor (2022).

A Figura 42 mostra os dados de erros, falhas ou inconsistências que ocorrem, no cenário da UBS, nas arquiteturas de software dos SIS gerados pelas ferramentas EhrScape, EhrBase, MARCIA, Template4EHR, Microservice4EHR e AdaptiveHIS.

Desta forma, observa-se que as arquiteturas de software geradas pelas ferramentas EhrScape e EhrBase (ilustrada na Figura 35), após a adição de cinco arquétipos na camada do cliente (conforme mostrado na Figura 40), executaram com cinco erros ou falhas. Isto é, os cinco arquétipos adicionados não tiveram os seus dados processados. Contudo, essas arquiteturas executaram o processamento dos dois primeiros arquétipos.

A arquitetura de software gerada pela ferramenta MARCIA (também ilustrada na Figura 33), após a adição de cinco arquétipos (conforme mostrado na Figura 40), executou com sete erros ou falhas. Isto é, todos os arquétipos não tiveram os seus dados processados devido a um erro de execução no SIS. Isso significa que a adição de novos arquétipos também comprometeu o processamento dos arquétipos existentes antes da modificação.

As arquiteturas de software geradas pela ferramenta Template4EHR e Microservice4EHR (ilustrada na Figura 34) tiveram o mesmo comportamento das ferramentas EhrScape e EhrBase.

Após a adição de cinco arquétipos, elas executaram com cinco erros, falhas ou inconsistência, isto é: não processando os dados dos novos arquétipos adicionados. Porém, executando os dados dos dois primeiros arquétipos.

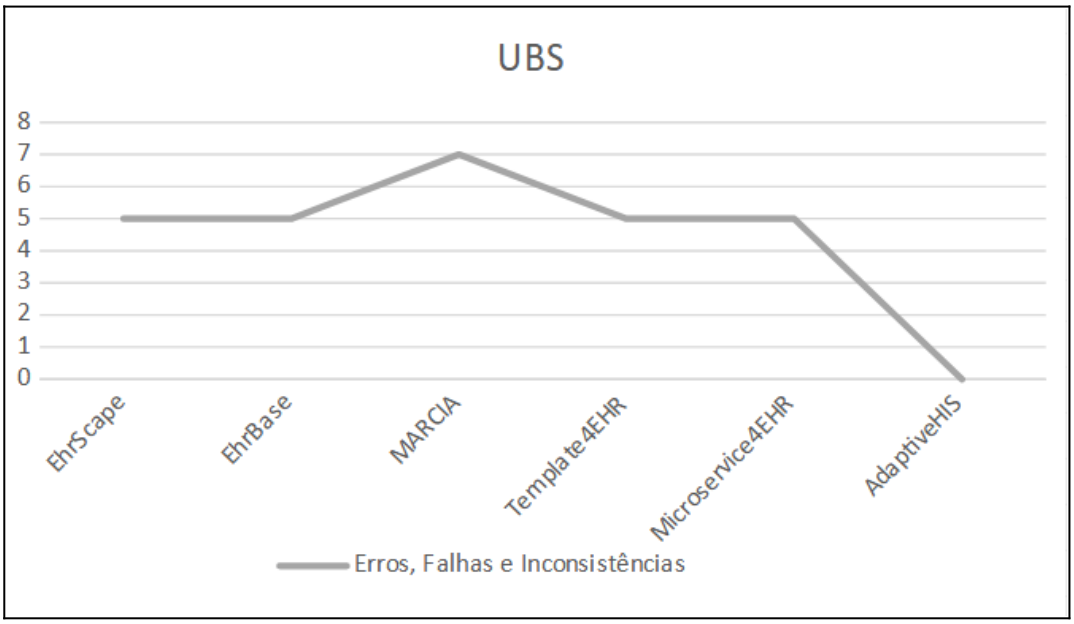
A arquitetura proposta nesta tese e gerada pela ferramenta AdaptiveHIS (mostrada na Figura 35), após a adição de cinco arquétipos, executou sem erros ou falhas, isto é: todos os sete arquétipos (os dois de antes da pandemia do COVID-19 e os cinco adicionados depois da pandemia) tiveram os seus dados processados.

A Figura 43 mostra os dados de erros, falhas ou inconsistências que ocorrem nas arquiteturas de software dos SIS gerados pelas mesmas ferramentas que foram consideradas no cenário dos hemocentros. Isto é, inicialmente elas foram construídas com doze arquétipos. No entanto, na segunda versão, adicionou-se três arquétipos e retirou-se dois.

As arquiteturas de softwares dos SIS gerados pelas ferramentas EhrScape, EhrBase, Template4EHR e Microservice4EHR apresentaram cinco erros, falhas e

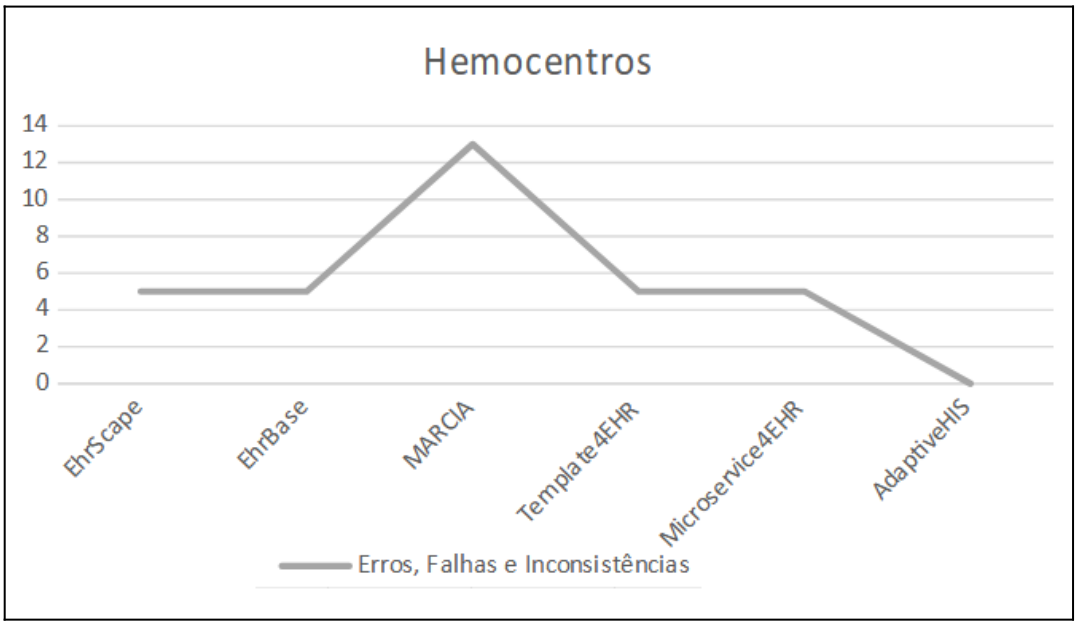
inconsistências. Dois deles referem-se ao processamento dos dois arquétipos retirados como dados nulos, enquanto três deles referem-se ao não processamento dos dados dos arquétipos adicionados.

Figura 42 - Erros, falhas ou inconsistências, no cenário da UBS.



Fonte: O autor (2022).

Figura 43 - Erros, falhas ou inconsistências, no cenário dos hemocentros.



Fonte: O autor (2022).

A arquitetura de software gerada pela ferramenta MARCIA teve treze erros, falhas ou inconsistências. Isto é, todos os arquétipos não tiveram os seus dados processados devido a um erro de execução no SIS. Isso significa que, a adição e remoção de arquétipos compromete o processamento do SIS que utiliza a arquitetura de software de MARCIA.

A arquitetura gerada pela ferramenta AdaptiveHIS, no cenário dos hemocentros, executou sem erros ou falhas. Isto é, todos os treze arquétipos tiveram os seus dados processados, assim como os dados dos arquétipos retirados não foram processados.

A Figura 44 mostra o número de novos componentes adaptados, na camada do servidor, após a modificação de arquétipos. Essa figura mostra que, no cenário da UBS, nenhuma arquitetura de software apresentou adaptação (adição ou retirada) de componentes, exceto a arquitetura de software gerada pela AdaptiveHIS.

Isto é, as arquiteturas geradas pelas ferramentas EhrScape, EhrBase, MARCIA, Template4EHR, Microservice4EHR mantiveram o número de componentes mesmo com a modificação de arquétipos. Contudo, a arquitetura de software gerada pela AdaptiveHIS tinha oito componentes na execução feita com os arquétipos usados antes da pandemia, mas com a modificação de arquétipos, houve uma adaptação de cinco componentes, os quais foram adicionados.

A Figura 45 mostra que, no cenário dos hemocentros, as arquiteturas tiveram comportamento similar ao apresentado no cenário da UBS. Isto é, nenhuma arquitetura de software apresentou adição ou remoção de componentes, exceto a arquitetura de software gerada pela AdaptiveHIS.

A arquitetura proposta nesta tese, para o cenário dos hemocentros, inicialmente, era composta por dezoito componentes de software quando executou suas funcionalidades com os arquétipos usados antes da pandemia. Com a modificação de arquétipos, essa mesma arquitetura ficou com dezenove componentes.

Esse resultado considera que a arquitetura adicionou três componentes, para processar os dados dos novos arquétipos, e removeu dois componentes, os quais processavam os dados de dois arquétipos existentes antes da mudança.

Esta remoção se refere à exclusão dos campos *Glicose no Sangue* e *Pulso* e seus respectivos arquétipos `lab_test-blood_glucose.v1` e `pulse.v1`, da aba *Medidas do Corpo* (conforme ilustrado na Figura 41). Então, isto significa que essa arquitetura adaptou cinco componentes.

Fazendo uma relação entre o número de componentes existentes antes da mudança de arquétipos e o número de componentes que foram adaptados (adicionados ou removidos) após as mudanças, pode-se inferir que:

- Na arquitetura proposta nesta tese (construída pela ferramenta AdaptiveHIS), conforme ilustrado na Figura 44, havia um total de oito componentes antes da mudança. Após a mudança de arquétipos, cinco componentes foram adaptados. Aplicando o cálculo de porcentagem {8*n* = 5 ◈ 100 / *n* é a porcentagem de adaptação}, verifica-se uma adaptação de 62%.
- No cenário dos hemocentros, após aplicar o cálculo de porcentagem {18*n* = 5 ◈ 100}, verifica-se uma adaptação de 27.7%.

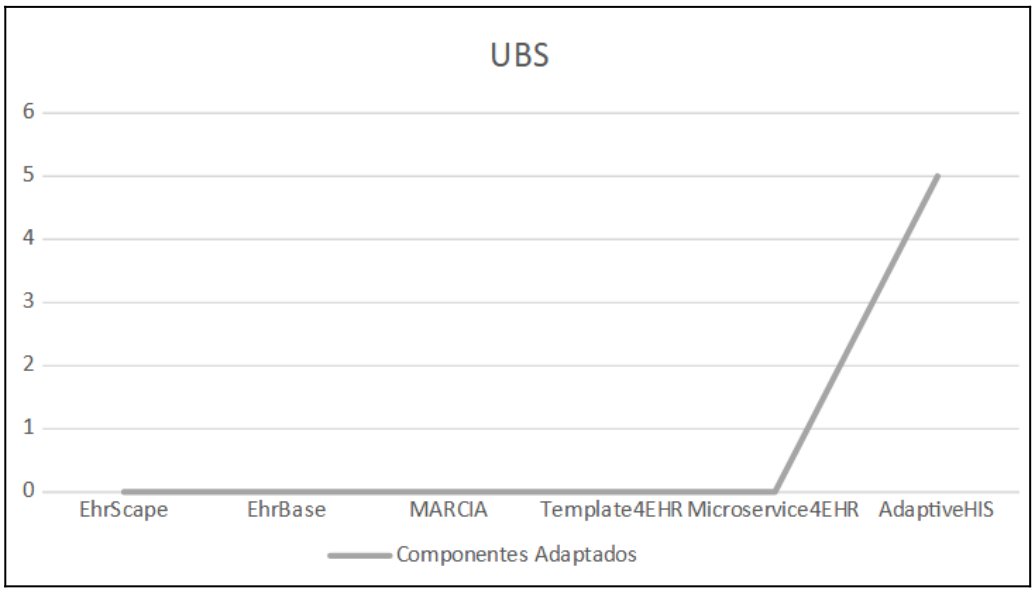
Constata-se também que o número de componentes adaptados é diretamente proporcional ao número de arquétipos modificados. Isto é, para cada arquétipo modificado (adicionado ou removido) na *interface com arquétipo*, um *Provedor CRUD* é também modificado (adicionado ou removido).

Para verificar o resultado utilizando outro cálculo, a métrica LSA (Perez-Palacin et al., 2014) foi utilizado. Esse cálculo propõe uma divisão cujo numerador é o número de componentes inicialmente usados em uma arquitetura de software, e o denominador é o número de componentes disponíveis na mesma arquitetura de software após mudanças estruturais.

Na métrica LSA, quando o resultado não for o número um, mas uma fração entre zero e 0.9, significa que a arquitetura é adaptável; e por outro lado, se o resultado for o número um, não há adaptação.

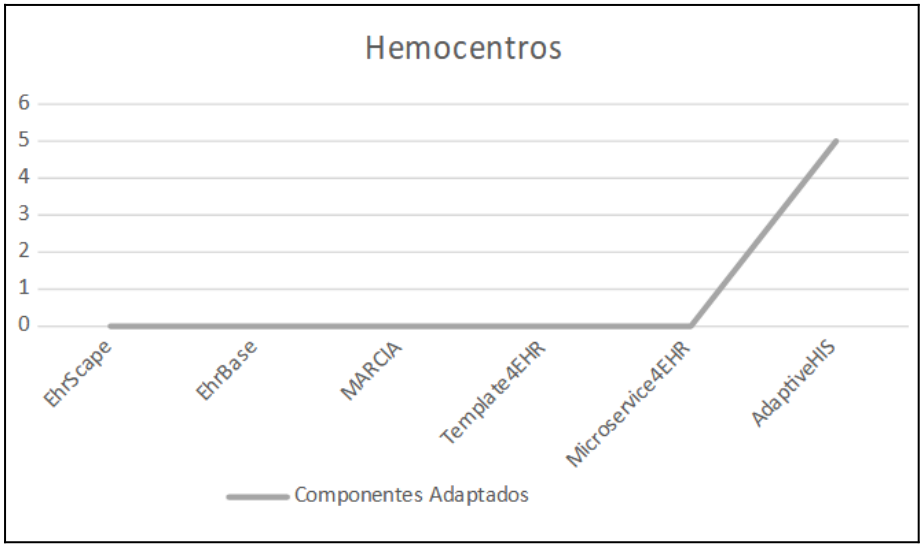
Para visualização do cálculo LSA de cada SIS, a Tabela 2 apresenta esse cálculo relacionando-o com os respectivos cenários, SIS e a ferramenta que gerou a arquitetura analisada. Desta forma, quinze SIS possuem a LSA igual ao número um (da linha 1 a 5 e da linha 7 a 16, na Tabela 2).

Figura 44 - Novos componentes após mudanças dos SIS, no cenário da UBS.



Fonte: O autor (2022).

Figura 45 - Novos componentes após mudanças dos SIS, no cenário do hemocentro.



Fonte: O autor (2022).

Isso significa que os componentes das arquiteturas de software desses SIS permaneceram os mesmos após as alterações. Essas arquiteturas de softwares não criam novos componentes para processar os dados dos novos arquétipos adicionados. Considerando o cenário da UBS e dos hemocentros, os cálculos LSA das arquiteturas de software geradas pelas ferramentas são, respectivamente:

- EhrScape: {LSA = 3🕑3 = 1} e {LSA = 3🕑3 = 1};

- EhrBase: {LSA = 3🕑3 = 1} e {LSA = 3🕑3 = 1};
- MARCIA: {LSA = 4🕑4 = 1} e {LSA = 4🕑4 = 1};
- Template4EHR: {LSA = 4🕑4 = 1} e {LSA = 4🕑4 = 1};
- Microservice4EHR: {LSA = 6🕑6 = 1} e {LSA = 26🕑26 = 1};
- AdaptiveHIS: {LSA = 8🕑13 = 0.61} e {LSA = 18🕑19 = 0.94}.

Observa-se que, nos dois cenários, os cálculos LSA da arquitetura gerada pela AdaptiveHIS é uma fração entre o número zero e 0.9, conforme mostrado nas linhas 6, 17 e 18 da Tabela 2. Baseada na métrica LSA, esse resultado significa que essa arquitetura é mais adaptável que as demais arquiteturas analisadas que se mantiveram no número um.

O cálculo LSA prova que a arquitetura proposta nesta tese é mais adaptável às mudanças dos arquétipos que as demais arquiteturas. Contudo, apesar desta prova, verificou-se uma limitação da LSA em não contabilizar os componentes removidos da arquitetura de software após as mudanças. Essa limitação impactou os resultados no cenário dos hemocentros, cujos resultados só contabilizaram a adição de um componente e não contabilizou a remoção de dois componentes. Esta remoção se refere à exclusão dos campos *Glicose no Sangue* e *Pulso* e seus respectivos arquétipos `lab_test-blood_glucose.v1` e `pulse.v1`, da aba *Medidas do Corpo* (conforme ilustrado na Figura 41). Verifica-se que, muito embora o HEMOSE e HEMOPE são companhias distintas, os resultados obtidos são os mesmos, não havendo diferença entre ambas.

Tabela 2 - Análise de adaptabilidade dos cenários da UBS e Hemocentros.

Linha	Cenário	SIS	Arquitetura da Ferramenta	LSA
1	UBS	SIS-1 UBS	EhrScape	1
2		SIS-1 UBS	EhrBase	1
3		SIS-2 UBS	MARCIA	1
4		SIS-3 UBS	Template4EHR	1
5		SIS-4 UBS	Microservice4EHR	1
6		**SIS-5 UBS**	**AdaptiveHIS**	**0.61**
7	Hemocentros	SIS-1 Hemose	EHR Scape	1
8		SIS-1 Hemope		1
9		SIS-2 Hemose	EHR Base	1

10		SIS-2 Hemope		1
11		SIS-3 Hemose	MARCIA	1
12		SIS-3 Hemope		1
13		SIS-4 Hemose	Template4EHR	1
14		SIS-4 Hemope		1
15		SIS-5 Hemose	Microservice4EHR	1
16		SIS-5 Hemope		1
17		**SIS-6 Hemose**	**AdaptiveHIS**	**0.94**
18		**SIS-6 Hemope**		**0.94**

Fonte: O autor (2022).

5.5 Limição da Métrica LSA

Duas limitações foram observadas, na avaliação efetuada nesta tese, com relação a métrica LSA. A primeira limitação é quando há adição e remoção de mesmo número de componentes da arquitetura, é possível que o resultado final da metrica seja o número um. Isso leva a falsa conclusão de que não houve adaptação da arquitetura. Desta forma, quando há remoção de componentes da arquitetura de software se mostra limitada.

Nesse caso, a avaliação quantitativa mostrada nas Figuras 42, 43, 44 e 45 parece ser mais adequada. No entanto, a métrica LSA mostra-se aplicável em contexto que há somente adição de componentes na arquitetura de software. A segunda limitação da métrica LSA é não permite medir se um SIS é mais adaptável que outro. No uso dessa métrica, a comparação é entre duas versões de um mesmo software. A primeira versão, refere-se ao software antes de uma mudança. A segunda, depois de uma mudança.

5.6 Replicabilidade da Avaliação

Conforme mostrada na Tabela 3, são apresentadas algumas diretrizes com as quais os pesquisadores podem replicar a avaliação apresentada neste capítulo.

Tabela 3 - Diretrizes para replicação da avaliação desta tese.

Passo 1	Identificar um conjunto de novos requisitos de saúde para construir um SIS.
Exemplo	Registro da pressão arterial de um paciente.
Quem	Um profissional da área da Saúde.
Passo 2	Identifique um conjunto de arquétipos para representar o RES desses requisitos de saúde descrito no primeiro passo.
Exemplo	`healthcare_provider_parent.v1, clinical_synopsis.v1, blood_pressure.v2, exam.v1.`
Quem	Um especialista nos padrões openEHR.
Passo 3	Desenvolver uma interface gráfica para interagir com usuário do SIS (e.g., HTML).
Exemplo	Exemplo de código está disponível no endereço eletrônico: *https://cin.ufpe.br/~maps3/ahsa/txt/blood_pressure.txt*.
Quem	Um engenheiro de software (ou desenvolvedor *frontend*).
Passo 4	Implementar uma conexão da GUI (do passo três) com a arquitetura proposta nesta tese.
Como	Primeira opção: Usar a ferramenta AdaptiveHIS para gerar todo o SIS. Uma implemenção em java da AdaptiveHIS está disponível no Anexo II; Segunda opção: Execute as seguintes escritas de código, as quais são apenas sugestões (não obrigatórias): 1. No arquito HTML, importe um framework javascript (e.g., Angular); 2. Implemente e importe o arquivo que executa o *Controlador* dessa GUI (passo 5); 3. Nos campos de dados HTML, defina o modelo de dadas de acordo com os campos dos arquétipos utilizados. Um exemplo é

	utilizar o recurso ng-model da ferramenta Angular.
Exemplo	Exemplo de código está disponível no endereço eletrônico: *https://cin.ufpe.br/~maps3/ahsa/txt/gen_blood_pressure.txt*.
Quem	Ferramenta AdaptiveHIS or um engenheiro de software.
Passo 5	Implemente o arquivo que executa o *controlador*.
Como	Primeira opção: Usar a ferramenta AdaptiveHIS para gerar todo o SIS; Segunda opção: Execute as seguintes escritas de código, as quais são apenas sugestões (não obrigatórias): • Construa um *controlador* (por exemplo, em javascript) para controlar todos os campos HTML; • Implemente uma comunicação desse *controlador* com o *adaptador em tempo de execução*; • No arquivo javascript, crie um modelo de dados (por exemplo, JSON) que represente o arquétipo descrito no arquivo HTML; • Envie esse dado JSON para o *adaptador*.
Exemplo	Exemplo de código está disponível no endereço eletrônico *https://cin.ufpe.br/~maps3/ahsa/txt/blood_pressure.js* e no Anexo I desta tese.
Quem	Ferramenta AdaptiveHIS or um engenheiro de software.
Passo 6	Checar se há um *Provedor CRUD* para cada arquétipo utilizado na GUI.
Como	Primeira opção: Usar a ferramenta AdaptiveHIS para gerar todo o SIS; Segunda opção: Cheque os *provedores CRUD* existentes. Isso pode ser efetuado através da API web do *adaptador em tempo de execução*. A documentação dessa API é disponível em: *https://cin.ufpe.br/~maps3/ahsa/pdf/runtimeLocator_ documentation.pdf* ou no endereço eletrônico *https://runtimelocator.herokuapp.*

	com/swagger-ui.html, no *endpoint* “*url/archetypes*”;
Quem	Ferramenta AdaptiveHIS or um engenheiro de software.
Passo 7	Se não houver um *Provedor CRUD* para dar suporte ao arquétipo que a GUI usa, é necessário criar um *provedor CRUD* para atender ao processamento CRUD desse arquétipo.
Como	Primeira opção: Usar a ferramenta AdaptiveHIS para gerar todo o SIS; Segunda opção: Execute as seguintes escritas de código, as quais são apenas sugestões (não obrigatórias): • Escolha uma linguagem de programação; • Escolha um framework para construir uma API; • Disponibilize esta API na Internet como um serviço.
Exemplo	Há exemplos de implementações de provedores CRUD no seguinte endereço eletrônico: *https://cin.ufpe.br/~maps3/ahsa* e no Anexo I desta tese.
Quem	Ferramenta AdaptiveHIS or um engenheiro de software.
Passo 8	Armazene todos os dados de conexão do provedor CRUD criado no passo 7 no *adaptador em tempo de execução*.
Como	Primeira opção: Usar a ferramenta AdaptiveHIS para gerar todo o SIS; Segunda opção: Envie os dados pela API web do adaptador em tempo de execução. A documentação dessa API é disponível em: *https://cin.ufpe.br/~maps3/ahsa/pdf/runtimeLocator_documentation.pdf* ou no endereço eletrônico *https://runtimelocator.herokuapp.com/swagger-ui.html,* no *endpoint* “*adaptivehis_request*”;
Quem	Ferramenta AdaptiveHIS or um engenheiro de software.

Fonte: O autor (2022).

5.7 Respostas às Questões de Pesquisa

Esta seção responde as questões de pesquisa definidas no início deste capítulo.

- **QP1**: **A arquitetura de software proposta utiliza arquétipos como artefato computacional para representação de seus RES?**

A arquitetura proposta oferece recursos que permite a construção de uma interface que utiliza arquétipos como artefato computacional para representação de RES. A *Interface do Usuário* e seus componentes *Formulário com Arquétipo*, *Campo de Dados* e *Referência* são fundamentais nesse processo, conforme descrito a seguir.

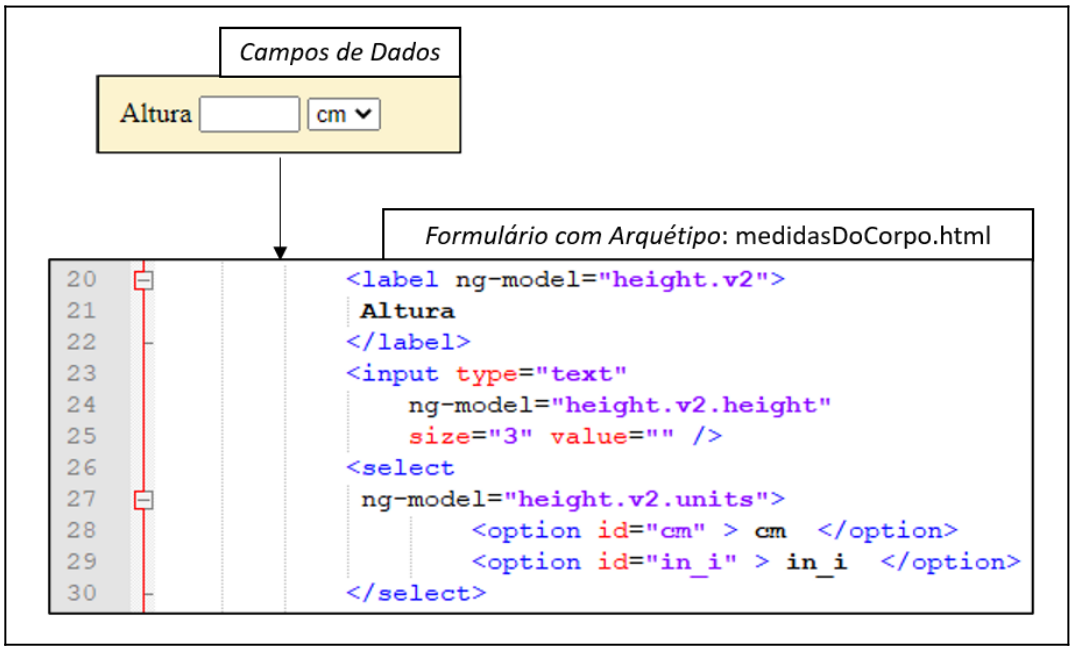
A Figura 46 mostra um fragmento de um formulário com arquétipo “Medidas do Corpo” e seu respectivo código fonte HTML. Esse fragmento é composto por três Campos de Dados: um rótulo (“Altura”), um campo para inserir dados, e um campo para seleção de dado (“cm”). No código fonte, o rótulo é representado da linha 20 a 22 pelo elemento HTML <label>. Para estabelecer uma relação entre um campo e um arquétipo, o subcomponente *Referência* foi especificado nesta tese, conforme mostrado na Figura 21.

Esse subcomponente faz referência ao arquétipo e, se necessário, ao atributo que o respectivo campo representa. Para implementar esse subcomponente, nesta tese, utilizou-se um recurso Javascript chamado ng-model (GOOGLE, 2020). Desta forma, na linha 20 é descrito que esse rótulo se refere ao arquétipo `height.v2`.

O campo para inserir dados é mostrado da linha 23 a 25. O elemento HTML `<input>` foi usado para construir esse campo na GUI. A referência desse campo (linha 22) é composta pelo nome do arquétipo e, também, por um atributo. Isso significa que esse campo se refere a um atributo específico desse arquétipo.

O campo para selecionar dados é mostrado da linha 26 a 30, o qual foi construído com dois elementos HTML: `<select>` e `<option>`. A referência desse campo (linha 27) descreve uma relação desse campo com o atributo units do arquétipo `height.v2`.

Figura 46 - Três Campos de Dados do formulário Medidas do Corpo, e seus respectivos códigos fontes em HTML.



Fonte: O autor (2022).

- **QP2**: **A arquitetura de software proposta identifica os arquétipos utilizados na camada do cliente em tempo de execução?**

A arquitetura proposta possui um recurso que permite identificar os arquétipos utilizados na camada do cliente, em tempo de execução. Os componentes *Controlador* e *Identificador* são fundamentais nesse processo de identificação dos arquétipos utilizados no SIS. Ambos estão, respectivamente, na *Interface do Usuário* e *Adaptador em Tempo de Execução*, ambos descritos na seção 4.3.2 no Capítulo 4.

A Figura 47 mostra uma implementação do processo de comunicação entre *Controlador* e *Identificador*. Nessa figura há fragmentos de código fonte HTML e código Javascript. Esse fragmento de código HTML refere-se ao arquivo medidasDoCorpo.html, o qual refere a um formulário web para registro de medidas do corpo (também mostrado na Figura 46). O fragmento de código Javascript, ilustrado na Figura 47, é uma implementação do componente *Controlador*, o qual é responsável por controlar os dados do arquivo medidasDoCorpo.html.

Para implementar *Controlador*, nesta tese, utilizou-se o recurso ng-controller (GOOGLE, 2020). Optou-se pelo uso desse recurso porque ele é disponibilizado gratuitamente na Internet. Porém, não é um critério obrigatório. Com esse recurso, é

possível ter controle sobre os dados digitados pelo usuário e sobre as requisições solicitadas, no formulário web (representado pelo arquivo medidasDoCorpo.html). Esse formulário web é representado na arquitetura proposta pelo componente *Formulário com Arquétipo*.

Para fazer uso do recurso ng-controller, o formulário importa dois arquivos externos (linhas 7 e 8, do arquivo medidasDoCorpo.html mostrado na Figura 47). A primeira importação viabiliza a infraestrutura necessária para executar o recurso ng-controller. A segunda importação viabiliza a interação do formulário com o arquivo que contém o respectivo controlador. Na linha 10, define-se no <body> o nome do controlador que interagirá com esse formulário (isto é, controlador-medidas-do-corpo).

Esse controlador é construído para interagir com *Formulário com Arquétipo* e seus subcomponentes *Campos de Dados* e *Referência*, nos quais estão descritos os arquétipos e seus respectivos atributos.

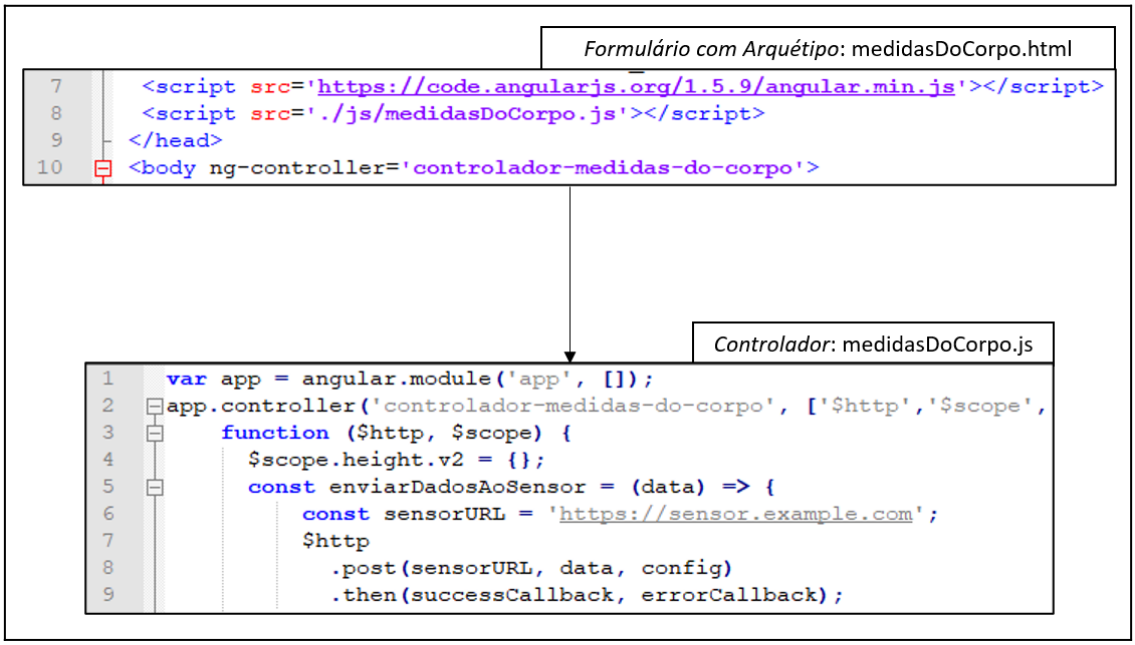
Esse controlador é implementado no arquivo medidasDoCorpo.js, cujo fragmento de código também é mostrado na Figura 47. A criação desse controlador é apresentada nas linhas 1 e 2 da Figura 47. Esse controlador possui uma função (mostrada na linha 3) cujos parâmetros são os recursos $scope e $http. O recurso $scope viabiliza a manipulação dos arquétipos descritos nos recursos ng-model.

O recurso $http permite a interação desse controlador com um componente de software externo, disponível na Internet. A Figura 47 ilustra o uso do recurso $http (da linha 3 a 9 do controlador), o qual viabiliza a interação de *Controlador* com *Identificador*. Então, *Identificador* executa a identificação dos arquétipos utilizados.

- **QP3**: **A arquitetura de software proposta executa uma adaptação, na camada do servidor, referente aos arquétipos utilizados na camada do cliente?**

Os componentes *Identificador*, *Localizador* e *Provedor CRUD* são fundamentais na adaptação da arquitetura proposta, todos esses componentes são descritos na seção 4.3.2 no Capítulo 4. A execução desses três componentes propõe a adaptação de um conjunto de serviços web disponíveis na Internet. Inicialmente, *Identificador* identifica os arquétipos que são utilizados na camada do cliente.

Figura 47 - Fragmentos de códigos fontes do formulário “*Medidas do Corpo*” e seu respectivo *Controlador*.



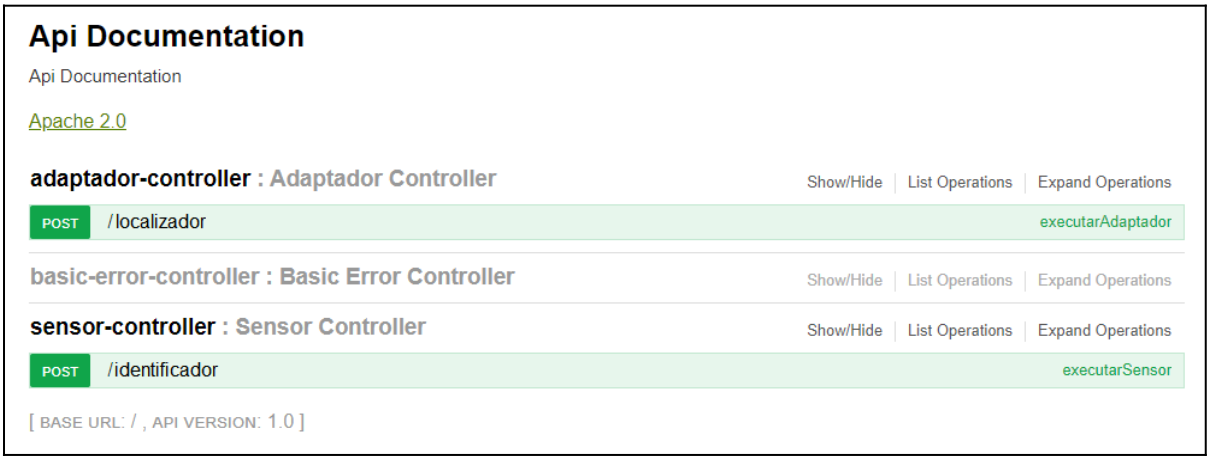
Fonte: O autor (2022).

Em seguida, *Localizador* faz uma busca em seu banco de dados para localizar os protocolos de comunicação dos *Provedores CRUD* aptos para processar os dados dos arquétipos identificados. Então, *Localizador* retorna para o *Controlador* quais serviços web serão utilizados na requisição solicitada. Finalmente, *Controlador* envia os dados para o conjunto de *Provedores CRUD* processados pelo *Adaptador em Tempo de Execução*.

A Figura 48 mostra uma documentação da API construída nesta tese para executar o *Adaptador em Tempo de Execução*. Esta figura mostra um *Identificador* e um *Localizador*. Desta forma, esses dois subcomponentes são acessíveis utilizando as regras do protocolo REST (mostrado na seção 2.2 do Capítulo 2).

Conforme mostrado na Figura 48, para acessa essa API deve utilizar o verbo “POST”, cada subcomponente tem rotas de acesso específicas (“/localizador” e “/identificador”), cada uma dessas rotas inicia a execução de uma respectiva função (“*executarSensor*” e “*executarAdaptador*”). Essa API foi utilizada na avaliação feita neste capítulo, com a finalidade de executar a adaptação proposta nesta tese.

Figura 48 - Documentação de API construída para executar o *Adaptador em Tempo de Execução*.



Fonte: O autor (2022).

5.8 Ameaças à Validade da Avaliação

Devido a escassez de estudos sobre adaptabilidade em SIS que usam arquétipo, o conjunto de ferramentas do estado da arte e da prática utilizado nesta avaliação foi limitado, conforme descrito na Seção 3.5 do Capítulo 3. No conjunto de ferramentas utilizados nessa avaliação, não há soluções que tragam adaptabilidade às arquiteturas de software.

No entanto, esse conjunto de ferramenta foram consideradas porque essas ferramentas constroem arquiteturas de SIS que usam arquétipos. Desta forma, os resultados dessa avaliação limitam-se a comprovar o ganho de adaptabilidade nesse contexto. Contudo, é importante informar que novas avaliações são necessárias, quando houver novas arquiteturas ou ferramentas que proponham adaptabilidade em SIS que usam de arquétipos.

Ao julgar a qualidade de um estudo de pesquisa, é muito importante considerar as ameaças à validade do estudo e dos resultados. Isso é particularmente importante para a pesquisa empírica, onde muitas vezes há uma infinidade de possíveis ameaças.

Com um foco crescente em métodos de pesquisa empírica em engenharia de software é importante que haja um consenso na comunidade sobre essa importância, que a análise de validade seja feita por todos os pesquisadores (Feldt &

Magazinius, 2010). A seguir seguem-se três ameaças à validade dos resultados apresentados nesta avaliação:

- **Validade de construção**: o foco deste estudo é a adaptabilidade do software em um ambiente mutável onde novos requisitos são fornecidos por arquétipos. Contudo, outras circunstâncias imprevisíveis, como mudanças em um ambiente do sistema e falhas do sistema podem desencadear ações adaptativas em um sistema de software.
- **Validade interna**: A avaliação e resultados apresentados neste estudo estão restritos a um contexto local com conjunto limitado de requisitos funcionais, recursos computacionais, arquétipos e ferramentas. Portanto, vale ressaltar que os resultados apresentados aqui não podem ser generalizados.
- **Validade externa**: A solução, aqui apresentada, opera em ambiente de computação em nuvem e não tem controle sobre alta latência de rede ou falhas no provedor de Internet, o que pode dificultar a comunicação entre os componentes da arquitetura proposta.

Para mitigar essas ameaças e promover maior replicabilidade nessa pesquisa, os componentes desenvolvidos, os arquétipos utilizados e orientações de teste e desenvolvimento foram disponibilizadas na Seção 5.6. Essas diretrizes incluem orientações para pesquisadores que desejam aplicar a arquitetura proposta nesta tese para novos domínios de aplicação.

5.9 Considerações Finais

Neste capítulo, construiu-se a ferramenta AdaptiveHIS para construção de um SIS com a arquitetura pospota nesta tese. Um algoritmo e implementação dessa ferramenta está disponível no endereço eletrônico *https://www.cin.ufpe.br/~maps3/ahsa*, na opção "*Algorithm/AdaptiveHIS Tool*" e "*Implementation/AdaptiveHIS Tool*", repectivamente. No Anexo II desta tese, há também uma implementação em Java dessa ferramenta.

Também, a utilização da arquitetura em dois cenários de Saúde para dois períodos de tempo distintos foi apresentada. Os dois cenários abordam a admissão de paciente em uma UBS e a triagem clínica inicial em hemocentros. Os períodos

referem-se ao período antes e depois da pandemia do COVID-19. Dessa forma, o capítulo mostrou os componentes implementados para o funcionamento dos cenários e as interações entre eles.

Todas as arquiteturas de software analisadas foram apresentadas por meio de um diagrama de componentes. Consequentemente, é mostrado como os componentes e subcomponentes especificados interagem mutualmente para proverem o funcionamento da arquitetura proposta nesta tese. Além disso, o uso de diagramas do modelo C4 permite a visualização dos *containers*, componentes e subcomponentes da arquitetura. Também, como eles se comportam e a função de cada um deles para promover a adaptação proposta nesta tese.

Com essa avaliação também foi possível visualizar a utilização dos comportamentos projetados na arquitetura proposta. Os resultados dessa avaliação também mostram que a arquitetura foi capaz de prover um aumento de até 62% na capacidade de adaptação nos cenários da UBS e hemocentros, respectivamente.

6 CONCLUSÃO

Este capítulo apresenta as considerações finais sobre os tópicos abordados nesta tese, incluindo as principais contribuições alcançadas, as limitações deste estudo e, por fim, as indicações de trabalhos futuros.

6.1 Considerações Finais

De acordo com o que foi apresentado no Capítulo 1, engenheiros de software (ou desenvolvedores) gastam a maior parte de seu tempo em atividades de retrabalho para ajustar software a modificações necessárias ao longo do ciclo de vida de um sistema de informação. Esse cenário resulta em um prejuízo bilionário. Ao mesmo tempo, para diminuir esse retrabalho, foi mostrado que alguns estudos propõem técnicas que permitem que arquiteturas de software incorporem dinamicamente novas características estruturais ou comportamentais. Isto é, essas arquiteturas são adaptáveis.

Descreveu-se também que alguns trabalhos sugerem que o uso de arquétipos facilita a alteração de funcionalidades de um SIS (mais rapidamente e com menos código) e a construção de uma arquitetura de software adaptável. Desse modo, o requisito de adaptabilidade e o uso de arquétipos são recursos que aparecem como soluções para lidar com o retrabalho.

Conforme mostrado no Capítulo 2, arquétipo (atributos e restrições de domínios), serviço web, computação em nuvem e adaptabilidade de software são recursos em uso na academia e no mercado, cujos trabalhos dão suporte aos sistemas de informação em geral e da área da saúde também. Esse capítulo mostra, ainda, como a adaptabilidade de software é trabalhada em conjunto com outros recursos, por exemplo: tempo de execução, serviços web, arquitetura de software e computação em nuvem.

O Capítulo 3 descreve trabalhos que usam as especificações openEHR em arquiteturas de software. Outros trabalhos abordam o uso de adaptabilidade em

arquiteturas de software para área da saúde. Porém, pouca atenção tem sido dada na construção de uma arquitetura adaptável, com a qual é possível uma adaptação decorrente de mudanças feitas no conjunto de arquétipos utilizados no SIS. Portanto, uma arquitetura de software que consegue executar uma adaptação, em tempo de execução, baseada nos arquétipos utilizados em um SIS, foi considerada um desafio em aberto e a ser trabalhado nesta tese.

Neste contexto, uma arquitetura de software projetada para ser adaptativa e baseada em arquétipos foi apresentada no Capítulo 4. Essa proposta tem como objetivo executar uma adaptação, em tempo de execução, em relação aos arquétipos de um SIS que usa as especificações openEHR. Para isso, alguns níveis (de contexto, de *container* e de componente) e visões (de uso e de modelo de dados), como também alguns comportamentos descritos em diagrama de casos de uso e de sequência, ilustram os conceitos e relacionamentos dos elementos que compõem a arquitetura proposta.

Além disso, para garantir que essa arquitetura possa ser usada sem dependência de tecnologia, utilizou-se um padrão arquitetural que permite que SIS acessem a arquitetura proposta sem que tenha conhecimento da tecnologia utilizada (serviços web). Por meio das funcionalidades desenvolvidas nos componentes da arquitetura, foi criada uma instância de SIS utilizando arquétipos e a arquitetura proposta.

Em seguida, o Capítulo 5 exibiu por meio de uma avaliação, as características de adaptabilidade e fez uma análise do comportamento da arquitetura consoante mudança nos arquétipos na camada do cliente de um conjunto de SIS. Nesta avaliação, utilizou-se a arquitetura para executar operações CRUD do novo conjunto de arquétipos utilizados na GUI, executando uma adaptação e dispensando a necessidade de retrabalho na escrita de códigos, testes ou implantação de software.

6.2 Contribuições

A principal contribuição científica desta tese é a definição de uma arquitetura de software adaptável que processa dados representados por um ou mais arquétipos, utilizados na GUI de um SIS. Essa arquitetura foi especificada com foco

no uso de arquétipo e com recurso de adaptabilidade. Desta forma, ela consegue se adaptar aos arquétipos utilizados em um SIS, mesmo que haja a alteração (retirada ou a adição) de arquétipos na GUI ao longo do ciclo de vida do SIS.

Na avaliação apresentada no Capítulo 5, os SIS que utilizaram a arquitetura proposta obtiveram um aumento percentual de adaptabilidade entre 27.7% e 62%, as quais variam conforme o número de arquétipos utilizados na GUI.

A partir dos resultados apresentados nesta tese, pode-se responder positivamente às questões de pesquisa desta tese, as quais são:

- A arquitetura de software proposta utiliza arquétipos como artefato computacional para representação de RES?
- A arquitetura de software proposta permite a modificação de arquétipos na camada do cliente?
- A arquitetura de software proposta executa uma adaptação, na camada do servidor, referente aos arquétipos utilizados na camada do cliente?

Também, verifica-se que os objetivos desta tese também foram alcançados, os quais são:

- Projetar componentes de uma arquitetura de software, os quais estejam em conformidade com as especificações definidas em arquétipos;
- Projetar uma arquitetura de software que consiga identificar o conjunto de arquétipos que é utilizado em um SIS, em tempo de execução;
- Projetar uma arquitetura de software que consiga adaptar, em tempo de execução, componentes de software para processamento dos dados representados pelos arquétipos identificados.

6.3 Limitações

Adaptabilidade é um conceito que extrapola o escopo abordado nesta tese. Contudo, a arquitetura proposta introduz o uso desse recurso aos SIS que utilizam arquétipos e que, se precisar alterar esses arquétipos (adicionando ou excluindo), ao longo de seu ciclo de vida, a arquitetura consegue se adaptar. Limitações de adataptabilidade com relação à arquietura proposta foram identificadas no desenvolvimento desta tese e trabalhos futuros são propostos (na Seção 6.4).

Há limitações nessa avaliação, as quais são descrita na Seção 5.8. Com isso, considera-se que a proposta apresentada nesta tese é uma solução limitada de adaptabilidade aos SIS que usam arquétipo. Novos estudos e pesquisas são necessários para que essa proposta evolua a novos patamares e atenda com maior completude os SIS dos mais diversos segmentos. Alguns trabalhos futuros são descritos no próximo capítulo.

A implementação da arquitetura proposta utilizou a computação em nuvem para execução de suas funcionalidades. Uma limitação identificada nesta tese é a obrigatoriedade de SIS estarem conectados com a Internet, para usufruir dessas funcionalidades. Entretanto, como o acesso à Internet é uma realidade em uma parcela significativa das instituições de saúde, esta limitação é atenuada nesse contexto.

Outra limitação se refere ao processamento do *Adaptador em Tempo de Execução*. Até a versão apresentada nesta tese, esse componente é executado a cada requisição enviada pela camada do cliente, independente do conjunto de arquétipo utilizado na GUI de um SIS. Esse processamento demanda um maior tempo de execução, o que é uma perda de recursos em SIS que não modificaram seus arquétipos.

6.4 Trabalhos Futuros

Como investigação futura, sugere-se adicionar outras abordagens de adaptabilidade na arquitetura proposta, por exemplo:

- Adaptabilidade ao ambiente do SIS: em caso de ausência de Internet, a arquitetura se adaptaria por meio de um sistema multiagentes (locais e remotos), por exemplo;
- Adaptabilidade ao uso de arquétipos inválidos ou não validados pelo CKM. Desta forma, a arquitetura garante o uso de um arquétipo válido e seus respectivos *Provedores CRUD*;
- Adaptabilidade à ausência de *Provedor CRUD*, isto é, em caso de ausência de um *Provedor CRUD* para processar dos dados de um determinado arquétipo, a arquitetura proposta inicia um processo de construção, testes e

implantação de um novo *provedor CRUD* para executar as operações CRUD desse arquétipo.

Devido as leis de proteção sobre dados de saúde a nível mundial (VUKONIC et al., 2022; FORNASIER, 2022), um trabalho futuro pertinente é a adição de um componente, com o qual seja possível deixar a arquitetura proposta nesta tese em conformidade com os critérios de segurança e de proteção de dados exigidos na área de Saúde.

Trabalhos futuros referentes ao componente *Formulário com Arquétipo*:

- Especificar um componente que verifique o status do arquétipo utilizado do CKM, para evitar a utilização de arquétipos não validados;
- Investigar um mecanismo que garantam dependências entre os *campos de dados* relacionados, para preservar a semântica dos dados no HTML;
- Estudar uma forma de adaptar a interface do usuário em caso de mudanças das terminologias utilizadas nos arquétipos;

Trabalhos futuros referentes ao componente *Adaptador em Tempo de Execução*:

- Especificar um mecanismo de *cache* pode evitar o processamento de adaptação quando não necessário. Isto é, se a requisição envia o mesmo conjunto de arquétipo de requisições anteriores, então, processar uma adaptação pode não ser necessário;
- Em *Identificador*, adicionar um mecanismo de identificação de arquétipos de GUI que usam *templates* openEHR (ou somente "*templates*") (openEHR, 2022);

Trabalhos futuros referentes aos *Provedores CRUD*:

- Estender os *Provedores CRUD* para processamentos mais complexos que as operações CRUD, por exemplo: serviços que façam processamento de imagem ou processe o diagnóstico de um paciente;
- Implementar novos *Provedores CRUD* referente aos arquétipos não contemplados na avaliação desta tese;

Trabalhos futuros referentes a avaliação efetuada nesta tese:

- Utilização de uma métrica de adaptabilidade que permita comparar a adaptabilidade entre SIS adaptáveis;
- Estudar o desempenho e a usabilidade das arquiteturas estudadas;

- Validar a aplicação da arquitetura proposta em uma aplicação inloco (em um hospital, clínica ou laboratório) para ser avaliado por outros profissionais, por exemplo: os profissionias da Saúde.
- Avaliar a aplicação da arquitetura proposta nesta tese em empresas que utilizam sistemas legados. Avaliando os desafios de executar a arquitetura em SIS construídos com tecnologias antigas, por exemplo: sistemas monolíticos;

Muito embora esta arquitetura tenha como objetivo o uso de arquétipos, uma investigação futura é a adição de outros padrões da área de saúde na arquitetura, com a finalidade de abranger um maior número de SIS, em todo mundo. Nesse contexto, outros padrões que podem ser considerados são o HL7 e FHIR (HL7 INTERNATIONAL, 2022), *European Healthcare Data Space* (EHDS) (HORGAN et al., 2022), *Formato Observational Medical Outcomes Partnership* (OMOP) (BYUN et al., 2022). Devido a isso, é pertinente a extensão da arquitetura proposta nesta tese para uso desses padrões.

REFERÊNCIAS

ABBAS, N., ANDERSSON, J. and WEYNS, D. (2020) ASPLe: A methodology to develop self-adaptive software systems with systematic reuse, Journal of Systems and Software, Volume 167, 2020, 110626, ISSN 0164-1212, https://doi.org/10.1016/j.jss.2020.110626.

ADAMKO, A., GARAI, A. and PENTEK, I. (2017) "Interaction-dependent e-health hub-software adaptation to cloud-based electronic health records," 2017 8th IEEE International Conference on Cognitive Infocommunications (CogInfoCom), 2017, pp. 000339-000344, doi: 10.1109/CogInfoCom.2017.8268267.

AFFONSO, F.J.; NAKAGAWA, E.Y. (2015) Self-adaptive Software: Development Approach and Automatic Process for Adaptation at Runtime. Revista Brasileira De Computacao Aplicada. Passo Fundo: Univ Passo Fundo, v. 7, n. 1, p. 68-84, 2015. Available at: <http://hdl.handle.net/11449/160610>.

ANDERSON J. et al. (2013) Software Engineering Processes for Self-Adaptive Systems. In: de Lemos R., Giese H., Müller H.A., Shaw M. (eds) Software Engineering for Self-Adaptive Systems II. Lecture Notes in Computer Science, vol 7475. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-35813-5_3.

ANDRIKOS, C., RASSIAS, G., TSANAKAS, P. and MAGLOGIANNIS, I. (2019) "An Enhanced Device-Transparent Real-Time Teleconsultation Environment for Radiologists," in IEEE Journal of Biomedical and Health Informatics, vol. 23, no. 1, pp. 374-386, Jan. 2019, doi: 10.1109/JBHI.2018.2824312.

ANWER, S., WEN, L. and Wang, Z. (2019) A Systematic Approach for Identifying Requirement Change Management Challenges: Preliminary Results. In Proceedings of the Evaluation and Assessment on Software Engineering (EASE '19). Association for Computing Machinery, New York, NY, USA, 230–235. DOI: https://doi.org/10.1145/3319008.3319031.

AMAZON (2022). Amazon Web Services. Disponível em: https://aws.amazon.com. Acessado em: 12/03/2022.

ARAÚJO, A., TIMES, V., SILVA, M. (2016) "PolyEHR: A Framework for Polyglot Persistence of the Electronic Health Record, " 2016 Int. Conf. Internet Computing and Internet of Things. ISBN: 1-60132-439-1. CSREA Press.

ARAÚJO, A., TIMES, V., SILVA, M. (2018) "A Cloud Service for Graphical User Interfaces Generation and Electronic Health Record Storage". In: Latifi S. (eds) Information Technology - New Generations. Advances in Intelligent Systems and Computing, vol 558. Springer. 2018. Print ISBN: 978-3-319-54977-4.

ARAÚJO, A., TIMES, V., SILVA, M, (2020) "A Tool for Generating Health Applications Using Archetypes," in IEEE Software, vol. 37, no. 1, pp. 60-67, Jan.-Feb. 2020, doi: 10.1109/MS.2018.110162508.

ATALAG, K., YANG, H.Y., TEMPERO, E. and WARREN, J.R. (2014) Evaluation of software maintainability with openEHR – a comparison of architectures, International Journal of Medical Informatics, Volume 83, Issue 11, 2014, Pages 849-859, ISSN 1386-5056, https://doi.org/10.1016/j.ijmedinf.2014.07.006.

AYALA, I., HORCAS, J.M., AMOR, M. and Fuentes, L. (2016). Using Models at Runtime to Adapt Self-managed Agents for the IoT. In Multiagent System Technologies: 14th German Conference, MATES 2016, Klagenfurt, Österreich, September 27-30, 2016. Proceedings (Vol. 9872, p. 155). Springer.

BABAR, M. A., SHEN, H., BIFFL, S. and WINKLER, D. (2019) "An Empirical Study of the Effectiveness of Software Architecture Evaluation Meetings," in IEEE Access, vol. 7, pp. 79069-79084, 2019, doi: 10.1109/ACCESS.2019.2922265.

BACELAR-SILVA, G.M., CÉSAR, H., BRAGA, P. and GUIMARÃES, R. (2013) "OpenEHR-based pervasive health information system for primary care: First Brazilian experience for public care," Proceedings of the 26th IEEE International Symposium on Computer-Based Medical Systems, 2013, pp. 572-873, doi: 10.1109/CBMS.2013.6627881.

BALIGAND, F. and MONFORT, V. (2004) A concrete solution for web services adaptability using policies and aspects. In Proceedings of the 2nd international

conference on Service oriented computing (ICSOC '04). Association for Computing Machinery, New York, NY, USA, 134–142. DOI: https://doi.org/10.1145/1035167.1035187.

BARBOSA, D.M., LIMA, R.G.M., MAIA, P.H.M. and COSTA E. (2017) "Lotus@Runtime: A Tool for Runtime Monitoring and Verification of Self-Adaptive Systems," 2017 IEEE/ACM 12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS), 2017, pp. 24-30, doi: 10.1109/SEAMS.2017.18.

BASS, L.; CLEMENTS, P.; KAZMAN, R. (2022) Software Architecture in Practice. Fourth Edition. United States: Addison-Wesley Professional.

BASSENE, A. and GUEYE, B. (2022) "A Self-adaptive QoS-management Framework for Highly Dynamic IoT Networks," 2022 IEEE Multi-conference on Natural and Engineering Sciences for Sahel's Sustainable Development (MNE3SD), 2022, pp. 1-8, doi: 10.1109/MNE3SD53781.2022.9723303.

BEALE, T. (2002) "Archetypes: Constraint-based domain models for future-proof information systems", Proc. 11th OOPSLA Workshop Behavioral Semantics: Serving the Customer, pp. 16-32, 2002.

BEALE, T., and HOVENGA, E. (2022). The knowledge-driven platform: Strategic technologies for a platform ecosystem approach. In Roadmap to Successful Digital Health Ecosystems (pp. 115-141). Academic Press.

BETTONI, G.N., LOBO, T.C., FLORES, C.D., SANTOS, B.G.T. and SILVA, F.S. (2021) "Application of HL7 FHIR in a Microservice Architecture for Patient Navigation on Registration and Appointments," 2021 IEEE/ACM 3rd International Workshop on Software Engineering for Healthcare (SEH), 2021, pp. 44-51, doi: 10.1109/SEH52539.2021.00015.

BETTER Platform (2017) EHRScape. [Online]. Disponível em: https://www.ehrscape.com. Acessado em: 20/03/2022.

BEYER, D., CHAKRABARTI, A. and HENZINGER, T.A. (2005) Web service interfaces. In Proceedings of the 14th international conference on World Wide Web (WWW '05). Association for Computing Machinery, New York, NY, USA, 148–159. DOI: https://doi.org/10.1145/1060745.1060770.

BROWN, S. (2022). The C4 model for visualising software architecture. [Online]. Disponível em: https://c4model.com. Acessado em: 10/09/2022.

BUCHANA, Y. and SEYMOUR, L. (2015) Theorising inter-organisational health information systems as complex adaptive systems in the context of emergency medical services. In Proceedings of the 2015 Annual Research Conference on South African Institute of Computer Scientists and Information Technologists (SAICSIT '15). Association for Computing Machinery, New York, NY, USA, Article 7, 1–7. DOI:https://doi.org/10.1145/2815782.2815785.

BUYYA, R., BROBERG, J. and GOSCINSKI, A. M., (2010) Cloud Computing: Principles and Paradigms. Hoboken, NJ, USA: John Wiley & Sons.

BYUN, J., LEE, D.Y., JEONG, C.W. et al. (2022) Analysis of treatment pattern of anti-dementia medications in newly diagnosed Alzheimer's dementia using OMOP CDM. Sci Rep 12, 4451 (2022). https://doi.org/10.1038/s41598-022-08595-1.

CABOLABS (2022) CloudEHRServer. [Online]. Disponível em: https://cloudehrserver.com. Acessado em 26/fev/2022.

CABRERA, C. and CLARKE, S. (2022) "A Self-Adaptive Service Discovery Model for Smart Cities," in IEEE Transactions on Services Computing, vol. 15, no. 1, pp. 386-399, 1 Jan.-Feb. 2022, doi: 10.1109/TSC.2019.2944356.4.

CALDERÓN-GOMES, H. et al. (2020) "Telemonitoring System for Infectious Disease Prediction in Elderly People Based on a Novel Microservice Architecture," in IEEE Access, vol. 8, pp. 118340-118354, 2020, doi: 10.1109/ACCESS.2020.3005638.

CALINESCU, R., WEYNS, D., GERASIMOU, S. et al. (2018) "Engineering Trustworthy Self-Adaptive Software with Dynamic Assurance Cases," in IEEE

Transactions on Software Engineering, vol. 44, no. 11, pp. 1039-1069, 1 Nov. 2018, doi: 10.1109/TSE.2017.2738640.

CARDELLINI, V., PRESTI, F., NARDELLI, M., and RUSSO, G. (2022) Run-time Adaptation of Data Stream Processing Systems: The State of the Art. ACM Comput. Surv. Just Accepted (January 2022). DOI:https://doi.org/10.1145/3514496.

CARLINI, E., COPPOLA, M., DAZZI, P., MORDACCHINI, M. and PASSARELLA, A. (2016) "Self-Optimising Decentralised Service Placement in Heterogeneous Cloud Federation," 2016 IEEE 10th International Conference on Self-Adaptive and Self-Organizing Systems (SASO), 2016, pp. 110-119, doi: 10.1109/SASO.2016.17.

CÉSAR, H.V. (2013) "OpenEHR-based pervasive health information system for primary care: First Brazilian experience for public care," 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2013), 2013, pp. 1483-1483, doi: 10.1109/ASONAM.2013.6785914.

CHA, J., Kim, J., and Jeong, Y. (2016). "Architecture Based Approaches Supporting Flexible Design of SelfAdaptive Software," 2016 Int. Conf. on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, pp. 1424-1425. doi: 10.1109/CSCI.2016.0280.

CHATTERJEE, S. (2003) "Developing enterprise Web services and applications: opportunities and best practices for the healthcare industry," Proceedings 5th International Workshop on Enterprise Networking and Computing in Healthcare Industry (HealthCom), 2003, pp. 159-, doi: 10.1109/HEALTH.2003.1218734.

CHENG SW., POLADIAN, V.V., GARLAN D., SCHMERL, B. (2009) Improving Architecture-Based Self-Adaptation through Resource Prediction. In: Cheng B.H.C., de Lemos R., Giese H., Inverardi P., Magee J. (eds) Software Engineering for Self-Adaptive Systems. Lecture Notes in Computer Science, vol 5525. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-02161-9_4.

CHOU, W. (2008) "Web Services: Software-as-a-Service (SaaS), Communication, and Beyond," 2008 IEEE Congress on Services Part II (services-2 2008), 2008, pp. 1-1, doi: 10.1109/SERVICES-2.2008.46.

CHRISTENSEN, E., CURBERA, F., MEREDITH, G. and WEERAWARAN, S. (2000) “Web Services Description Language (WSDL) version 1.0”, https://www.w3.org, September 2000.

CIANCIARUSO, L., FORENZA, F., and NITTO, E. et al. (2014) "Using Models at Runtime to Support Adaptable Monitoring of Multi-clouds Applications," 2014 16th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, 2014, pp. 401-408, doi: 10.1109/SYNASC.2014.60.

CLEMENTS, P. et al. (2010) Documenting Software Architectures: Views and Beyond. Second Edition. United States: Addison-Wesley.

COUTO, H., ARAÚJO, A., SOARES, R., RODRIGUES, G. (2022). The Use of Blockchain Technology in Electronic Health Record Management: An Analysis of State of the Art and Practice. In: Latifi, S. (eds) ITNG 2022 19th International Conference on Information Technology-New Generations. Advances in Intelligent Systems and Computing, vol 1421. Springer, Cham. https://doi.org/10.1007/978-3-030-97652-1_22

D'ANGELO, M. (2018) "Decentralized Self-Adaptive Computing at the Edge," 2018 IEEE/ACM 13th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS), 2018, pp. 144-148.

D'ANGELO, M. et al. (2019) "On Learning in Collective Self-Adaptive Systems: State of Practice and a 3D Framework," 2019 IEEE/ACM 14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS), 2019, pp. 13-24, doi: 10.1109/SEAMS.2019.00012.

DEMSKI, H., GARDE, S. and HILDEBRAND, C. (2016) Open data models for smart health interconnected applications: the example of openEHR. BMC Med Inform Decis Mak 16, 137 (2016). https://doi.org/10.1186/s12911-016-0376-2.

DETRO, S.P., Santos, E.A.P., Panetto, H., Loures, E.F.R., LEZOCHE, M. and BARRA, C.M.C.M. (2020) Applying process mining and semantic reasoning for process model customisation in healthcare. Enterp. Inf. Syst. 14(7): 983-1009 (2020).

ELLOUZEA, A. S., Tlilia, S. H., and Bouazizb, R. (2017). A Model-Driven Based Methodology for the Generation of ContextAware Medical Interfaces from OpenEHR Archetypes. J Health Med Informat, 8(279), 2.

EHRBASE (2022). [Online]. Disponível em: https://ehrbase.org. Acessado em: 25/06/2022.

ESPOSITO, C., CASTIGLIONE, A., TUDORICA, C. and POP, F. (2017) "Security and privacy for cloud-based data management in the health network service chain: a microservice approach," in IEEE Communications Magazine, vol. 55, no. 9, pp. 102-108, Sept. 2017, doi: 10.1109/MCOM.2017.1700089.

FAYAD, M. and CLINE M.P. (1996) Aspects of Software Adaptability. Communications of the ACM, vol. 39, n. 10.

FAYAD, M.E., HAMZA, H.S. and SANCHEZ H.A. (2005) "Towards scalable and adaptable software architectures," IRI -2005 IEEE International Conference on Information Reuse and Integration, Conf, 2005., 2005, pp. 102-107, doi: 10.1109/IRI-05.2005.1506457.

FELDT, R., MAGAZINIUS, A. (2010) Validity Threats in Empirical Software Engineering Research - An Initial Survey. Proceedings of the 22nd International Conference on Software Engineering & Knowledge Engineering (SEKE'2010), Redwood City, San Francisco Bay, CA, USA, July 1 - July 3, 2010.

FELHI, F. and AKAICHI, J. (2013) "Pervasive e-healthcare system based on self-adaptability of SOA to the context," 2013 3rd International Conference on Information Technology and e-Services (ICITeS), 2013, pp. 1-5, doi: 10.1109/ICITeS.2013.6624070.

FENSEL, d. and BUSSLER, C. (2002) The Web Service Modeling Framework WSMF. Electronic Commerce Research and Applications, Volume 1, Issue 2, 2002, Pages 113-137, ISSN 1567-4223, https://doi.org/10.1016/S1567-4223(02)00015-7.

FIELDING, R.T. (2000). REST: architectural styles and the design of network-based software architectures. Doctoral dissertation, University of California.

FOLMER, E., GURP, j. and BOSCH, J. (2005) Scenario-based Assessment of Software Architecture Usability. Engineering Human Computer Interaction and Interactive Systems, 2005, Volume 3425. ISBN : 978-3-540-26097-4.

FORNASIER, M. O. (2022). The use of AI in digital health services and privacy regulation in GDPR and LGPD: between revolution and (dis) respect. Revista de Informação Legislativa, 59(233), 201-220.

FOWLER, M. (2022). [Online]. Disponível em: https://martinfowler.com/articles/building-infrastructure-platform.html#CommunicateYourTechnicalVision. Acessado em: 01/09/2022.

G. BRASILEIRO (2018). Lei Geral de Proteção de Dados Pessoais (LGPD). Disponível em: http://www.planalto.gov.br/ccivil_03/_ato2015-2018/2018/lei/l13709.htm. Acessado em: 10/02/2022.

G. BRASILEIRO (2020a). Covid-19: ministério muda protocolo de atendimento em posto de saúde. [Online]. Disponível em: https://agenciabrasil.ebc.com.br/saude/noticia/2020-03/covid-19-ministerio-muda-protocolo-de-atendimento-em-posto-de-saude. Acessado em: 27/01/2022.

G. BRASILEIRO (2020b). Atualização dos critérios técnicos para triagem clínica de dengue, chikungynya, zika e coronavírus nos candidatos à doação de sangue. [Online]. Disponível em: https://www.saude.go.gov.br/files//ostransparencia/hemogo-idtech/informacoesgerais/legislacao-aplicavel/NOTA%20TECNICA%20N%205.2020-CGSH.DAET.SAES.MS.pdf. Acessado em 28/01/2022.

GADIOLI, D., PALERMO, G. and SILVANO, C. (2015) "Application autotuning to support runtime adaptivity in multicore architectures," 2015 International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS), 2015, pp. 173-180, doi: 10.1109/SAMOS.2015.7363673.

GARAI, Á. and PÉNTEK, I. (2015) "Adaptive services with cloud architecture for telemedicine," 2015 6th IEEE International Conference on Cognitive Infocommunications (CogInfoCom), 2015, pp. 369-374, doi: 10.1109/CogInfoCom.2015.7390621.

GARLAN, D. , S. Cheng, A. Huang, B. Schmerl and P. Steenkiste, (2004) "Rainbow: architecture-based self-adaptation with reusable infrastructure," in Computer, vol. 37, no. 10, pp. 46-54, Oct. 2004, doi: 10.1109/MC.2004.175.

GIL, E.B., CALDAS, R., RODRIGUES, A., SILVA, G.L.G., et al. (2021) "Body Sensor Network: A Self-Adaptive System Exemplar in the Healthcare Domain," 2021 International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS), 2021, pp. 224-230, doi: 10.1109/SEAMS51251.2021.00037.

GIT (2022) Git--local-branching-on-the-cheap. Disponível em: https://git-scm.com. Acessado em: 12/02/2022.

GOMES DC, Abreu N, Sousa P, Moro C, Carvalho DR, Cubas MR. (2021) Representation of Diagnosis and Nursing Interventions in OpenEHR Archetypes. Appl Clin Inform. 2021 Mar;12(2):340-347. doi: 10.1055/s-0041-1728706. Epub 2021 Apr 14. PMID: 33853142; PMCID: PMC8046594.

GOMES, F., Paiva, J., Bezerra, A. et al. (2018) "MARCIA: Applied Clinical Record Management: Electronic Health Record Applied with EHRServer,"2018 IEEE 20th Int. Conf. on e-Health Networking, Applications and Services (Healthcom), Ostrava, 2018, pp. 1-6. doi: 10.1109/HealthCom.2018.8531096.

GOOGLE (2020). Service components in NG. Disponível em: https://docs.angularjs.org/api/ng/service. Acessado em: 13/02/2022.

GOOGLE (2022). Google Cloud Platform. [Online]. Disponível em: https://cloud.google.com. Acesso em: 12/03/2022.

GRUA E.M., De Sanctis M., Lago P. A (2020) Reference Architecture for Personalized and Self-adaptive e-Health Apps. In: Muccini H. et al. (eds) Software Architecture. ECSA 2020. Communications in Computer and Information Science, vol 1269. Springer, Cham.

GRUA, E.M., SANCTIS, M., MALAVOLTA, I et al. (2022) An evaluation of the effectiveness of personalization and self-adaptation for e-Health apps, Information and Software Technology, Volume 146, 2022, 106841, ISSN 0950-5849, https://doi.org/10.1016/j.infsof.2022.106841.

HAK, F., OLIVEIRA, D., ABREU, N., LEUSCHNER, P., ABELHA, A. and SANTOS, M. (2020) An OpenEHR Adoption in a Portuguese Healthcare Facility, Procedia Computer Science, Volume 170, 2020, Pages 1047-1052, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2020.03.075.

HAN, J.H. and LEE, J.Y. (2021) "Digital Healthcare Industry and Technology Trends," 2021 IEEE International Conference on Big Data and Smart Computing (BigComp), 2021, pp. 375-377, doi: 10.1109/BigComp51126.2021.00083.

HL7 INTERNATIONAL (2022). Health Level Seven International. Disponível em: https://www.hl7.org. Acessado em: 23/maio/2022.

HOG, C.E., DJEMAA, R.B., and AMOUS, I. (2012) Profile annotation for adaptable Web Service description. In Proceedings of the 27th Annual ACM Symposium on Applied Computing (SAC '12). Association for Computing Machinery, New York, NY, USA, 1935–1940. DOI: https://doi.org/10.1145/2245276.2232096.

HORGAN, D., HAJDUCH, M., VRANA, M. Soderberg J, Hughes N, Omar MI, Lal JA, Kozaric M, Cascini F, Thaler V, Solà-Morales O, Romão M, Destrebecq F, Sky Gross E. (2022) European Health Data Space—An Opportunity Now to Grasp the Future of Data-Driven Healthcare. Healthcare. 2022; 10(9):1629. https://doi.org/10.3390/healthcare10091629.

HORIKOSHI, H., Nakagawa, H., Tahara, Y. and Ohsuga, A. (2012). Dynamic reconfiguration in self-adaptive systems considering non-functional properties. In Proceedings of the 27th Annual ACM Symposium on Applied Computing (SAC '12). Association for Computing Machinery, New York, NY, USA, 1144–1150. DOI: https://doi.org/10.1145/2245276.2231956.

HOUMANI, Z., BALOUEK-THOMERT, D., CARON, E. and PARASHAR, M. (2020) "Enhancing microservices architectures using data-driven service discovery and QoS guarantees," 2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID), 2020, pp. 290-299, doi: 10.1109/CCGrid49817.2020.00-64.

IANCULESCU, M., ALEXANDRU, A., NEAGU, G. and POP, F. (2019) "Microservice-Based Approach to Enforce an IoHT Oriented Architecture," 2019 E-Health and Bioengineering Conference (EHB), 2019, pp. 1-4, doi: 10.1109/EHB47216.2019.8970059.

IBM (2019) What is a web service? Disponível em: https://www.ibm.com/docs/en/cics-ts/5.1?topic=services-what-is-web-service. Acessado em: 05/03/2022.

IGLESIAS, N., JUAREZ, J.M. & CAMPOS, M. (2022) Business Process Model and Notation and openEHR Task Planning for Clinical Pathway Standards in Infections: Critical Analysis. J Med Internet Res. 2022 Sep 15;24(9):e29927. doi: 10.2196/29927. PMID: 36107480.

ISO/IEC/IEEE (2020) "International Standard for Health informatics--Device interoperability--Part 20701:Point-of-care medical device communication--Service oriented medical device exchange architecture and protocol binding," in ISO/IEEE 11073-20701:2020(E), vol., no., pp.1-48, 30 March 2020, doi: 10.1109/IEEESTD.2020.9052096.

ISO (2019) ISO 13606-2:2019 Health informatics — Electronic health record communication — Part 2: Archetype interchange specification. Disponível em: https://www.iso.org/standard/62305.html. Acessado em 26/fev/2022.

JAYBHAYE, S. M. & ATTAR, V. Z., (2017) "A review on scientific workflow scheduling in cloud computing", 2nd International Conference on Communication and Electronics Systems (ICCES), pp.218-223.

JONES, M. (2022) Chapter 24 - Caribbean/PAHO—Jamaican case study, Editor(s): Evelyn Hovenga, Heather Grain, Roadmap to Successful Digital Health Ecosystems, Academic Press, 2022, Pages 523-535, ISBN 9780128234136, https://doi.org/10.1016/B978-0-12-823413-6.00026-4.

JUNIOR, E.C., ANDRADE, R.M.C. and ROCHA, L.S. (2021) "Development Process for Self-adaptive Applications of the Internet of Health Things based on Movement Patterns," 2021 IEEE 9th International Conference on Healthcare Informatics (ICHI), 2021, pp. 437-438, doi: 10.1109/ICHI52183.2021.00073.

KAKIVAYA, G. et al., "Service fabric: a distributed platform for building microservices in the cloud," in Proceedings of the 13 EuroSys Conf. (EuroSys '18). ACM, New York, NY, USA, 2018, Article 33, 15 pages. DOI: 10.1145/3190508.3190546.

KAUR H, ALAM MA, JAMEEL, R, Mourya AK, Chang V. (2018) A proposed solution and future direction for blockchain-based heterogeneous medicare data in cloud environment. J Med Syst. 2018;42(8):1–11.

KAZMAN, R., CARRIÈRE, S.J. nad WOODS, S.G. (2000) Toward a discipline of scenario-based architectural engineering. Annals of Software Engineering 9, 5–33 (2000). https://doi.org/10.1023/A:1018964405965.

KEMPA, B., Zhang, P., Jones, P. H., Zambreno, J., & Rozier, K. Y. (2020). Embedding online runtime verification for fault disambiguation on Robonaut2. In International Conference on Formal Modeling and Analysis of Timed Systems (pp. 196-214). Springer, Cham.

KIRDA, E. (2001) Web engineering device independent web services. In Proceedings of the 23rd International Conference on Software Engineering (ICSE '01). IEEE Computer Society, USA, 795–796.

KORRA, S., BIKSHAM, V., RANJENDER, N. and REDDY, T.C. (2021) "Building Adaptive Software Reusable Components Using Domain Engineering," 2021 Sixth International Conference on Image Information Processing (ICIIP), 2021, pp. 229-234, doi: 10.1109/ICIIP53038.2021.9702594.

KRASNER, H. (2018) The Cost of Poor Quality Software in the US. Disponível em: https://www.it-cisq.org/the-cost-of-poor-quality-software-in-the-us-a-2018-report/The-Cost-of-Poor-Quality-Software-in-the-US-2018-Report.pdf. Acessado em: 22/fev/2022.

KRYSZYN, J., CYWONIUK, K., SMOLIK, W. T., WANTA, D., et al. (2022) Performance of an openEHR based hospital information system, International Journal of Medical Informatics, Volume 162, 2022, 104757, ISSN 1386-5056, https://doi.org/10.1016/j.ijmedinf.2022.104757.

LADAS, N., FRANZ, S., Haarbrandt, B., et al. (2022). openEHR-to-FHIR: Converting openEHR Compositions to Fast Healthcare Interoperability Resources (FHIR) for the German Corona Consensus Dataset (GECCO). Studies in health technology and informatics, 289, 485-486.

LADAVA, M. & NORRIS, T. (2022). Supporting openEHR with Azure Health Data Services. [ONLINE]. Disponível em: https://azure.microsoft.com/en-us/blog/supporting-openehr-with-azure-health-data-services. Acessado em: 25/06/2022.

LEE, E., SEO, Y.D. and KIM, Y.G. (2022) "Self-Adaptive Framework with Master-Slave Architecture for Internet of Things," in IEEE Internet of Things Journal, doi: 10.1109/JIOT.2022.3150598.

LEMAHIEU, W. (2001) Web service description, advertising and discovery: WSDL and beyond J. Vandenbulcke, M. Snoeck (Eds.), New Directions in Software Engineering, Leuven University Press.

LEMOS, R., GIESE, H., MULLER, A. & SHAW, M. (2013) Software Engineering for Self-Adaptive Systems II, 2013, Volume 7475, Springer, ISBN : 978-3-642-35812-8.

LEYMANN, F. (2001) Web Services Flow Language (WSFL 1.0), IBM Software Group 2001, pp. 118.

LI, M., CAI, H., ZHI, Y. et al. (2022a) A configurable method for clinical quality measurement through electronic health records based on openEHR and CQL. BMC Med Inform Decis Mak 22, 37 (2022). https://doi.org/10.1186/s12911-022-01763-3.

LI, M., Leslie, H., Qi, B., Nan, S., et al (2022b) Development of an openEHR Template for COVID-19 Based on Clinical Guidelines. J Med Internet Res 2020;22(6):e20239. DOI: 10.2196/20239.

LIASKOS, S., WANG, B. and ALIMOHAMMADI N. (2019) "Blockchain Networks as Adaptive Systems," 2019 IEEE/ACM 14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS), 2019, pp. 139-145, doi: 10.1109/SEAMS.2019.00025.

LIM, Y.J., HONG, G., SHIN, D. et al. (2016) "A runtime verification framework for dynamically adaptive multi-agent systems," 2016 International Conference on Big Data and Smart Computing (BigComp), 2016, pp. 509-512, doi: 10.1109/BIGCOMP.2016.7425981.

MALLYA, R. and KOTHARI, S. (2018) "Self-Adaptive Woman Health Monitoring System Using MAPE Components," 2018 3rd International Conference for Convergence in Technology (I2CT), 2018, pp. 1-7, doi: 10.1109/I2CT.2018.8529760.

MCKINLEY, P.K., Seyed Masoud Sadjadi, Eric P. Kasten, and Betty H. C. Cheng. (2004) “Composing Adaptive Software”. In: Computer 37.7 (2004), pp. 56–64.

MELLO, B.H., RIGO, S.J., COSTA, C.A. et al. (2022) Semantic interoperability in health records standards: a systematic literature review. Health Technol. (2022). https://doi.org/10.1007/s12553-022-00639-w.

MOGHADAM, M.H., SAADATMAND, M., BORG, M., BOHLIN, M. and LISPER, B. (2018) "Adaptive Runtime Response Time Control in PLC-based Real-Time Systems Using Reinforcement Learning," 2018 IEEE/ACM 13th International Symposium on

Software Engineering for Adaptive and Self-Managing Systems (SEAMS), 2018, pp. 217-223.

MOTTA GHMB, Araújo DAB, Lucena-Neto JR, Azevedo-Marques PM, Cordeiro SS, Araújo-Neto SA. Towards an Information Infrastructure for Medical Image Sharing. J Digit Imaging. 2020 Feb;33(1):88-98. doi: 10.1007/s10278-019-00243-x. PMID: 31197560; PMCID: PMC7064679.

MICROSOFT (2022). Azure. [Online]. Disponível em: https://azure.microsoft.com. Acessado em: 25/06/2022.

MUKHIYA, S.K., RABBI, F., PUN, K.I. and LAMO, Y. (2019) "An Architectural Design for Self-Reporting E-Health Systems," 2019 IEEE/ACM 1st International Workshop on Software Engineering for Healthcare (SEH), 2019, pp. 1-8, doi: 10.1109/SEH.2019.00008.

MUKHIYA, S.K., WAKE, J.D., INAL, Y. and LAMO, Y. (2020) "Adaptive Systems for Internet-Delivered Psychological Treatments," in IEEE Access, vol. 8, pp. 112220-112236, 2020, doi: 10.1109/ACCESS.2020.3002793.

MUSLIM, A., Puspitodjati, S., Mutiara, A.B., et al. (2017) "Web services of transformation data based on OpenEHR into Health Level Seven (HL7) standards," 2017 Second International Conference on Informatics and Computing (ICIC), Jayapura, 2017, pp. 1-4. Doi: 10.1109/IAC.2017.8280571.

MUSZYNSKI, M., LUGTIGHEID, S., CASTOR, F. and BRINKKEMPER, S. (2022) "A Study on the Software Architecture Documentation Practices and Maturity in Open-Source Software Development," 2022 IEEE 19th International Conference on Software Architecture (ICSA), 2022, pp. 47-57, doi: 10.1109/ICSA53651.2022.00013.

MUTANU, L. and KOTONYA, G. (2016) "Consumer-Centred Validation for Runtime Adaptation in Service-Oriented System," 2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA), 2016, pp. 16-23, doi: 10.1109/SOCA.2016.9.

MUTANU, L., and Kotonya, G. (2018). A self-learning approach for validation of runtime adaptation in service-oriented systems. Service Oriented Computing and Applications, 12(1), 11-24.

OLIVEIRA, D., Miranda, R., Leuschner, P. et al. (2021) OpenEHR modeling: improving clinical records during the COVID-19 pandemic. Health Technol. 11, 1109–1118 (2021). https://doi.org/10.1007/s12553-021-00556-4.

openEHR (2022) Clinical Knowledge Manage. Disponível em: https://ckm.openehr.org/ckm/. Acessado em: 24/01/2022.

OREIZY, P., GORLICK, M.M., TAYLOR, R.N. et al. (1999) An Architecture-based Approach to SelfAdaptive Software. IEEE Intelligent Systems. 14(3), p. 54-62, May-June, 1999.

OREIZY, P., MEDVIDOVIC, N., and TAYLOR, R. (2008) Runtime software adaptation: framework, approaches, and styles. In Companion of the 30th international conference on Software engineering (ICSE Companion '08). Association for Computing Machinery, New York, NY, USA, 899–910. DOI: https://doi.org/10.1145/1370175.1370181.

PALIWAL, G., BUNGLOWALA, A. and KANTHED, P. (2022) An architectural design study of electronic healthcare record systems with associated context parameters on MIMIC III. Health Technol. (2022). https://doi.org/10.1007/s12553-022-00638-x.

PATNAIK, S.K. and BABU C.N. (2021) A web information extraction framework with adaptive and failure prediction feature. J. Data and Information Quality Just Accepted (October 2021). DOI: https://doi.org/10.1145/3495008.

PATRIARCA-ALMEIDA, J. H., SANTOS, B. and CRUZ-CORREIA, R.J. (2013) "Using a clinical document importance estimator to optimize an agent-based clinical report retrieval system," Proceedings of the 26th IEEE International Symposium on Computer-Based Medical Systems, 2013, pp. 469-472, doi: 10.1109/CBMS.2013.6627843.

PEDRERA-JIMÉNEZ, M.; Spanish Expert Group on EHR standards; KALRA, D.; BEALE, T.; MIÑOZ-CARRERO, A.; SERRANO-BALAZOTE, P. (2022): Can OpenEHR, ISO 13606 and HL7 FHIR work together? An agnostic perspective for the selection and application of EHR standards from Spain. TechRxiv. Preprint. https://doi.org/10.36227/techrxiv.19746484.v1.

PEREIRA, J., SILVA, R., ANTUNES, N., SILVA, J.L.M., FRANÇA, B., MORAES, R. and VIEIRA, M. (2020) A platform to enable self-adaptive cloud applications using trustworthiness properties. Proceedings of the IEEE/ACM 15th International Symposium on Software Engineering for Adaptive and Self-Managing Systems. Association for Computing Machinery, New York, NY, USA, 71–77. DOI: https://doi.org/10.1145/3387939.3391608.

PEREZ-PALACIN, D., MIRANDOLA, R., MERSEGUER, J. (2014) On the relationships between QoS and software adaptability at the architectural level, Journal of Systems and Software, Volume 87, 2014, Pages 1-17, ISSN 0164-1212. Elsevier.

POHJONEN, H. (2022) Chapter 20 - Norway, Sweden, and Finland as forerunners in open ecosystems and openEHR, Editor(s): Evelyn Hovenga, Heather Grain, Roadmap to Successful Digital Health Ecosystems, Academic Press, 2022, Pages 457-471, ISBN 9780128234136, https://doi.org/10.1016/B978-0-12-823413-6.00011-2.

PRESSMAN, R., MAXIM, B. (2019) Software Engineering: A Practitioner's Approach. 9th. ed. McGraw-Hill Education, ISBN: 1259872971.

QUIN, F., WEYNS, D., GHEIBI, O. (2021) Decentralized Self-Adaptive Systems: A Mapping Study. 16th International Symposium on Software Engineering for Adaptive and Self-Managing Systems.

QURESHI, N. and PERINI, A. (2010) "Continuous adaptive requirements engineering: An architecture for self-adaptive service-based applications," in 2010 First International Workshop on Requirements at Run Time (RE@RunTime), Sydney, NSW, 2010 pp. 17-24. DOI:10.1109/RERUNTIME.2010.5628553.

RAJESWARI, S., KALAISELVI, R. (2017) "Survey of data and storage security in cloud computing ", IEEE International Conference on Circuits and Systems (ICCS), pp.76-81.

REEVES, J.J., et al., (2020) “Rapid response to COVID-19: health informatics support for outbreak management in an academic health system,” J. Am. Med. Informatics Assoc., vol. 27, no. 6, pp. 853–859, 2020, doi: 10.1093/jamia/ocaa037.

REINBACHER, T., ROZIER, K.Y., and SCHUMANN, J. (2014) Temporal-Logic Based Runtime Observer Pairs for System Health Management of Real-Time Systems. In: Ábrahám E., Havelund K. (eds) Tools and Algorithms for the Construction and Analysis of Systems. TACAS 2014. Lecture Notes in Computer Science, vol 8413. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-54862-8_24.

REIS, L.F., FERREIRA, D.G., MARANHÃO, P.A., CRUZ-CORREIA, R. and VIEIRA-MARQUES, P. (2018) "Integration through mapping — An OpenEHR based approach for research oriented integration of health information systems," 2018 13th Iberian Conference on Information Systems and Technologies (CISTI), 2018, pp. 1-5, doi: 10.23919/CISTI.2018.8399258.

RIAZ, A.R., RAUF, A., GILANI, S.M.M. and BASHIR, M.B. (2021) "Techniques and Methodologies of Self-Adaptive Software and its Architecture: A Survey," 2021 4th International Conference on Computing & Information Sciences (ICCIS), 2021, pp. 1-5, doi: 10.1109/ICCIS54243.2021.9676190.

RICHARDS, M. (2015) Software Architecture Patterns. O'Reilly Media, Inc. ISBN: 9781491924242.

RIPPLE, Foundation (2022) EtherCIS. [Online]. Disponível em: https://www.ripple.foundation/ethercis/. Acessado em 26/fev/2022.

ROBERTSON, P., SHROBE, H., LADDAGA, R. (2001) Self-Adaptive Software. First International Workshop, IWSAS 2000 Oxford, UK, April 17–19, Springer.

ROSA, N., CAVALCANTI, D., CAMPOS, G. et al. (2020) Adaptive middleware in go - a software architecture-based approach. J Internet Serv Appl 11, 3 (2020). https://doi.org/10.1186/s13174-020-00124-5.

RUST, P., PICARD, G., and RAMPARANY, F. (2021) Resilient Distributed Constraint Reasoning to Autonomously Configure and Adapt IoT Environments. ACM Trans. Internet Technol. Just Accepted (December 2021). DOI: https://doi.org/10.1145/3507907.

SABUHI, M., MAHMOUDI, N. and KHAZAEI, H. (2021) Optimizing the Performance of Containerized Cloud Software Systems Using Adaptive PID Controllers. ACM Trans. Auton. Adapt. Syst. 15, 3, Article 8 (September 2020), 27 pages. DOI: https://doi.org/10.1145/3465630.

SACHDEVA, S., YAGINUMA, D., CHU, W., BHALLA, S. (2011) Dynamic Generation of Archetype-Based User Interfaces for Queries on Electronic Health Record Databases. In: Kikuchi S., Madaan A., Sachdeva S., Bhalla S. (eds) Databases in Networked Information Systems. DNIS 2011. Lecture Notes in Computer Science, vol 7108. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-25731-5_10.

SADUOVA, A. and AL-MASRI, E. (2021) "A Self-Adaptive IoT-based Approach for Improving the Decision Making of Active Surgical Robots in Hospitals," 2021 IEEE 3rd Eurasia Conference on Biomedical Engineering, Healthcare and Sustainability (ECBIOS), 2021, pp. 270-273, doi: 10.1109/ECBIOS51820.2021.9510397.

SALEHIE, M. and TAHVILDARI, L. (2009) Self-adaptive software: Landscape and research challenges. ACM Trans. Auton. Adapt. Syst. 4, 2, Article 14 (May 2009), 42 pages. DOI: https://doi.org/10.1145/1516533.1516538.

SALESFORCE (2022). Heroku. Disponível em: https://www.heroku.com/. Acessado em: 30/01/2022.

SANTOS, B. RODIGRUES, P. and CRUZ-CORREIA, R. (2013) "An automatic clinical document importance estimator for an existing electronic patient record —

Architecture and implementation," Proceedings of the 26th IEEE International Symposium on Computer-Based Medical Systems, 2013, pp. 537-539, doi: 10.1109/CBMS.2013.6627866.

SAVARGIV, M., NAZEMI, E., and MEHRMOLAEI, S. (2017) “Improved Self-Management Architecture in Self-Adaptive System,” Artif. Intell. Robot., pp. 1–5, 2017, doi: 10.1109/RIOS.2017.7956435.

SCHLIEMANN, T.; DANIELSEN, C.; VIRTANEN, T.; VUOKKO, R. (2019) eHealth standardisation in the Nordic countries: Technical and partially semantics standardisation as a strategic means for realising national policies in eHealth. Copenhagen: Nordisk Ministerråd, 2019, p. 68. DOI: 10.6027/TN2019-537.

SCHREIER, G., and HAYN, D. (2018). Achieving interoperability between arden-syntaxbased clinical decision support and openehr-based data systems. In Health Informatics Meets EHealth: Biomedical Meets EHealth–From Sensors to Decisions. Proceedings of the 12th EHealth Conference (Vol. 248, p. 338). IOS Press.

SELMI, I., KABACHI, N., LAMINE, S. and ZGHAL, Z.H. (2020) Adaptive Agent-Based Architecture for Health Data Integration. In: Yangui S. et al. (eds) Service-Oriented Computing – ICSOC 2019 Workshops. ICSOC 2019. Lecture Notes in Computer Science, vol 12019. Springer, Cham. https://doi.org/10.1007/978-3-030-45989-5_18.

SHEVTSOV, S., BEREKMERI, M., WEYNS, D. and MAGGIO, M. (2018) "Control-Theoretical Software Adaptation: A Systematic Literature Review," in IEEE Transactions on Software Engineering, vol. 44, no. 8, pp. 784-810, 1 Aug. 2018, doi: 10.1109/TSE.2017.2704579.

SILVA, M.A.P., TIMES, V.C., ARAÚJO, A.M.C. and SILVA, P.C. (2019) "A Microservice-Based Approach for Increasing Software Reusability in Health Applications," 2019 IEEE/ACS 16th International Conference on Computer Systems and Applications (AICCSA), 2019, pp. 1-8, doi: 10.1109/AICCSA47632.2019.9035229.

SILVA, M.A.P, V. C. Times, A. M. C. Araújo, and P. C. Silva (2020a) “A Decoupled Health Software Architecture using Microservices and OpenEHR Archetypes,” Int. J. Comput. Appl., vol. 176, no. 28, pp. 21–29, 2020, doi: 10.5120/ijca2020920302.

SILVA, M.A.P, Times, V.C., Araújo, A.M.C., Silva, P.C. (2020b) A Visual Modelling Language to Build Healthcare Applications Based on a Reusable Software Architecture. Facit Business and Technology Journal. ISSN: 25264281.

SILVA, M.A.P., TIMES, V.C., ARAÚJO, A.M.C. and SILVA, P.C. (2022) "Toward an Adaptive Software Architecture for Archetype-Based Health-Care Applications," in IEEE Software, vol. 39, no. 2, pp. 89-96, March-April 2022, doi: 10.1109/MS.2021.3070418.

SILVA-FERREIRA, P.R., Vieira-Marques, P. M., Patriarca-Almeida, J. H. and Cruz-Correia, R. J. (2012) "Improving expressiveness of agents using openEHR to retrieve multi-institutional health data: Feeding local repositories through HL7 based providers," 7th Iberian Conference on Information Systems and Technologies (CISTI 2012), 2012, pp. 1-5.

SOBHY, D., BAHSOON, R., MINKU, L., and KAZMAN, R. (2021) Evaluation of Software Architectures under Uncertainty: A Systematic Literature Review. ACM Trans. Softw. Eng. Methodol. 30, 4, Article 51 (October 2021), 50 pages. https://doi.org/10.1145/3464305.

SRIVATSA, R.R. and GEETHA, K.S. (2020) "PRS Generic Data Store Service," 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT), 2020, pp. 1-6, doi: 10.1109/ICCCNT49239.2020.9225498.

SUBRAMANIAN, N. and CHUNG, L. (2001) Software architecture adaptability: an NFR approach. In Proceedings of the 4th International Workshop on Principles of Software Evolution (IWPSE '01). Association for Computing Machinery, New York, NY, USA, 52–61. DOI: https://doi.org/10.1145/602461.602470.

STELMACH, P. and FALAS, L. (2013) "Runtime data-driven adaptation of composite e-Health services," 2013 IEEE 15th International Conference on e-Health

Networking, Applications and Services (Healthcom 2013), 2013, pp. 85-89, doi: 10.1109/HealthCom.2013.6720644.

STOCKER, W. (2018) "From Agile to Continuous Development in the Healthcare Domain - Lessons Learned," 2018 IEEE/ACM 40th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP), 2018, pp. 211-212.

SUH, M. et al. (2012) "Dynamic self-adaptive remote health monitoring system for diabetics," 2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society, 2012, pp. 2223-2226, doi: 10.1109/EMBC.2012.6346404.

SUNDVAL, E., NYSTROM, M., KARLSSON, D. et al. (2013) Applying representational state transfer (REST) architecture to archetype-based electronic health record systems. BMC Med Inform Decis Mak 13, 57 (2013). https://doi.org/10.1186/1472-6947-13-57.

TAYLOR, R. N., MEDVIDOVIC, N. and OREIZY, P. (2009) "Architectural styles for runtime software adaptation," 2009 Joint Working IEEE/IFIP Conference on Software Architecture & European Conference on Software Architecture, 2009, pp. 171-180, doi: 10.1109/WICSA.2009.5290803.

TARENSKEEN, D., WETERING, R., BAKKER, R. and BRINKKEMPER, S. (2020) The Contribution of Conceptual Independence to IT Infrastructure Flexibility: The Case of openEHR, Health Policy and Technology, Volume 9, Issue 2, 2020, Pages 235-246, ISSN 2211-8837, https://doi.org/10.1016/j.hlpt.2020.04.001.

TIAN, F., LIANG, P., and BABAR, M.A. (2022) Relationships between software architecture and source code in practice: An exploratory survey and interview, Information and Software Technology, Volume 141, 2022, 106705, ISSN 0950-5849, https://doi.org/10.1016/j.infsof.2021.106705.

VAZHENIN, D. (2012) Cloud-based web-service for health 2.0. In Proceedings of the 2012 Joint International Conference on Human-Centered Computer Environments

(HCCE '12). Association for Computing Machinery, New York, NY, USA, 240–243. DOI: https://doi.org/10.1145/2160749.2160800.

VIEIRA-MARQUES, P., PATRIARCA-ALMEIDA, J., FRADE, S., BACELAR-SILVA, G., ROBLES, S. and CRUZ-CORREIA, R. (2014) "OpenEHR aware multi agent system for inter-institutional health data integration," 2014 9th Iberian Conference on Information Systems and Technologies (CISTI), 2014, pp. 1-6, doi: 10.1109/CISTI.2014.6876864.

VILLARREAL-VASQUEZ, M. et al. (2017) "An MTD-Based Self-Adaptive Resilience Approach for Cloud Systems," 2017 IEEE 10th International Conference on Cloud Computing (CLOUD), 2017, pp. 723-726, doi: 10.1109/CLOUD.2017.101.

VOGEL, T. and GIESE, H. (2010) Adaptation and abstract runtime models. In Proceedings of the 2010 ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems (SEAMS '10). Association for Computing Machinery, New York, NY, USA, 39–48. DOI: https://doi.org/10.1145/1808984.1808989.

VUKONIC, J., IVANKOVIC, D., HABL, C., & DIMNJAKOVIC, J. (2022). Enablers and barriers to the secondary use of health data in Europe: general data protection regulation perspective. Archives of Public Health, 80(1), 1-9.

WANG, J., Zhang, Y., Hong, M. et al. (2022) A self-adaptive level-based learning artificial bee colony algorithm for feature selection on high-dimensional classification. Soft Comput (2022). https://doi.org/10.1007/s00500-022-06826-1.

WEYNS, D. and IFTIKHAR, U. (2016). Model-based simulation at runtime for self-adaptive systems. Proceeding Models at Runtime, Würzburg 2016, 1-9.

WHO (2022), “World Health Organization,” [Online]. Disponível em: https://www.who.int. Acessado em 19/Jan/2021.

WU, Y., FU, W., SHAO, J. and TIANZHOU, C. (2013) "Design of a novel cloud terminal adaptation platform based on mobile web-middleware," 2013 First International Symposium on Future Information and Communication Technologies for

Ubiquitous HealthCare (Ubi-HealthTech), 2013, pp. 1-5, doi: 10.1109/Ubi-HealthTech.2013.6708057.

ULRIKSEN, GH., PEDERSEN, R. nad ELLINGSEN, G. (2017) Infrastructuring in Healthcare through the OpenEHR Architecture. Comput Supported Coop Work 26, 33–69 (2017). https://doi.org/10.1007/s10606-017-9269-x.

YOSHIURA, V., Oliveira, L., Miyoshi, N. and Barbosa, F. (2016). Avaliação de uma aplicação openEHR utilizando RESTful Web Services. Electronic Journal of Health Informatics. 8. 107-116.

YONBAWI, S. and CALINESCU, R. (2018) "Towards Self-Adaptive Systems with Hierarchical Decentralised Control," 2018 IEEE 3rd International Workshops on Foundations and Applications of Self* Systems (FAS*W), 2018, pp. 14-16, doi: 10.1109/FAS-W.2018.00017.

ZHANG, M. (2017) "Adaptive system construction of medical knowledge agent learning technology," 2017 IEEE 2nd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), 2017, pp. 2458-2462, doi: 10.1109/IAEAC.2017.8054465.

ZOGHI, P., SHTERN, M., LITOIU, M. and GHANBARI, H. (2016) Designing Adaptive Applications Deployed on Cloud Environments. ACM Trans. Auton. Adapt. Syst. 10, 4, Article 25 (February 2016), 26 pages. DOI: https://doi.org/10.1145/2822896.

ANEXO A – CÓDIGO DO PROVEDOR CRUD `BLOOD_PRESSURE.V2`

```
::::::::[Existing Graphical Interface component] - BloodPressure.html:
<!-- Developer must add these two tags in HTML file: -->
<meta charset="ISO-8859-1" />
<script src='https://code.angularjs.org/1.5.9/angular.min.js'></script>
<script src='./js/bloodPressure.js'></script>

::::::::[Connector component] - bloodPressure.js:
/* created by https://micro4ehr.herokuapp.com (2022-08-21 10:29:04.406), based on Javascript/Angular(1.5.9)
*/

var app = angular.module('app', []);

app.controller('bloodpressure-Controller', ['$http','$scope', function ($http, $scope) {

  $scope.bloodPressure = {} ;
  $scope.host = 'http://localhost:8080';

  $scope.save = function() {
    $scope.bloodPressure.systolic = $scope.systolic;
    $scope.bloodPressure.systolicUnit = $scope.systolicUnit;
    $scope.bloodPressure.dyastolic = $scope.dyastolic;
    $scope.bloodPressure.dyastolicUnit = $scope.dyastolicUnit;
    $scope.bloodPressure.meanArterialPressure = $scope.meanArterialPressure;
    $scope.bloodPressure.meanArterialPressureUnit = $scope.meanArterialPressureUnit;
    $scope.bloodPressure.sleepStatus = $scope.sleepStatus;
    $scope.bloodPressure.dyastolicEndpoint = $scope.dyastolicEndpoint;
    //var below is an unique name (id) generated by Microservice4EHR in order to trace it through Internet
    $scope.bloodPressure.application = 'app48640_20220821102904404';

    $http({
      method:'POST',
      url: $scope.host+'/bloodPressure',
      data: $scope.bloodPressure
    }).then(function success(response){
       $scope.msg = 'function save() has been done successfully';
       console.log (response);
    },function unsuccess(response){
       console.log('There is some error: '+response);
    });

  }
}]);

::::::::[Microservice component] - application.properties:
#created by https://micro4ehr.herokuapp.com (2022-08-21 10:29:04.406), based on SpringBoot 3.9.3
#this PROPERTIES belong to microservice48640_20220821102904404
spring.thymeleaf.mode=html
spring.thymeleaf.cache=false
# ================================
# = DATA SOURCE
# ================================
# Set here configurations for the database connection
spring.datasource.url=jdbc:postgresql://localhost:5432/<put your db name here>
spring.datasource.username=<put db username here>
spring.datasource.password=<put db password here>
spring.datasource.driver-class-name=org.postgresql.Driver
spring.jpa.hibernate.ddl-auto=create-drop
spring.jpa.properties.hibernate.temp.use_jdbc_metadata_defaults = false

::::::::[Microservice component] - pom.xml:
<?xml version="1.0" encoding="UTF-8"?>
<!-- created by https://micro4ehr.herokuapp.com (2022-08-21 10:29:04.406), based on SpringBoot 3.9.3 -->
<!-- this POM belong to microservice48640_20220821102904404 -->
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.0.1.RELEASE</version>
    <relativePath/> <!-- lookup parent from repository -->
  </parent>
  <groupId> com.example.demo.microservice48640_20220821102904404 </groupId>
```

```
        <artifactId>    <!--    https://micro4ehr.herokuapp.com    has    indentified    this    app    as:
microservice48640_20220821102904404 --> </artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name> <!-- developer must put here the name of project: microservice48640_20220821102904404--></name>
  <description>A REST API to support the form: BloodPressure in host: http://localhost:8080</description>
  <properties>
    <java.version>1.8</java.version>
  </properties>
  <build>
      <plugins>
        <plugin>
          <groupId>org.springframework.boot</groupId>
          <artifactId>spring-boot-maven-plugin</artifactId>
           <executions>
            <execution>
             <goals>
              <goal>repackage</goal>
             </goals>
            </execution>
           </executions>
        </plugin>
      </plugins>
  </build>
  <dependencies>
        <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
  <!-- https://mvnrepository.com/artifact/org.postgresql/postgresql -->
    <dependency>
      <groupId>org.postgresql</groupId>
      <artifactId>postgresql</artifactId>
      <!--<version>9.4-1206-jdbc42</version>$NO-MVN-MAN-VER$-->
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-thymeleaf</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-tomcat</artifactId>
      <scope>provided</scope>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-test</artifactId>
      <scope>test</scope>
    </dependency>
    </dependencies>
</project>

:::::::[Microservice component] - BloodPressureApplication.java:
/* created by https://micro4ehr.herokuapp.com (2022-08-21 10:29:04.406), based on SpringBoot 3.9.3


package com.example.demo.microservice48640_20220821102904404;


import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import com.example.demo.microservice48640_20220821102904404.Tracer;


@SpringBootApplication
public class BloodPressureApplication {


  private String microservice = microservice48640_20220821102904404;
  private Tracer trace = new Tracer();


  public static void main(String[] args) {
    SpringApplication.run(BloodPressureApplication.class, args);
    trace.execute("microservice48640_20220821102904404 - BloodPressureApplication.class", "run", "");
  }
}

:::::::[Microservice component] - BloodPressureModel.java:
/* created by https://micro4ehr.herokuapp.com (2022-08-21 10:29:04.406), based on SpringBoot 3.9.3
   this class belong to microservice48640_20220821102904404 */
```

```
package com.example.demo.microservice48640_20220821102904404;


import java.util.logging.Logger;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;


@Entity
public class BloodPressureModel{


        private String systolic;
        private String systolicUnit;
        private String dyastolic;
        private String dyastolicUnit;
        private String meanArterialPressure;
        private String meanArterialPressureUnit;
        private String sleepStatus;
        private String dyastolicEndpoint;
        @Id
        @GeneratedValue
        private String id;
        private String archetype = "openEHR-EHR-OBSERVATION.blood_pressure.v2.adl";

        //contructors
        public BloodPressureModel() { }
        public  BloodPressureModel(String  systolic,  String  systolicUnit,  String  dyastolic,  String
dyastolicUnit,  String  meanArterialPressure,  String  meanArterialPressureUnit,  String  sleepStatus,  String
dyastolicEndpoint, String id, String archetype){
                this.systolic = systolic;
                this.systolicUnit = systolicUnit;
                this.dyastolic = dyastolic;
                this.dyastolicUnit = dyastolicUnit;
                this.meanArterialPressure = meanArterialPressure;
                this.meanArterialPressureUnit = meanArterialPressureUnit;
                this.sleepStatus = sleepStatus;
                this.dyastolicEndpoint = dyastolicEndpoint;
                this.id = id;
                this.archetype = archetype;
        }
        //getters and setters
        public String getSystolic() {
                return systolic;
        }
        public void setSystolic(String systolic) {
                this.systolic = systolic;
        }
        public String getSystolicUnit() {
                return systolicUnit;
        }
        public void setSystolicUnit(String systolicUnit) {
                this.systolicUnit = systolicUnit;
        }
        public String getDyastolic() {
                return dyastolic;
        }
        public void setDyastolic(String dyastolic) {
                this.dyastolic = dyastolic;
        }
        public String getDyastolicUnit() {
                return dyastolicUnit;
        }
        public void setDyastolicUnit(String dyastolicUnit) {
                this.dyastolicUnit = dyastolicUnit;
        }
        public String getMeanArterialPressure() {
                return meanArterialPressure;
        }
        public void setMeanArterialPressure(String meanArterialPressure) {
                this.meanArterialPressure = meanArterialPressure;
        }
        public String getMeanArterialPressureUnit() {
                return meanArterialPressureUnit;
        }
        public void setMeanArterialPressureUnit(String meanArterialPressureUnit) {
                this.meanArterialPressureUnit = meanArterialPressureUnit;
        }
        public String getSleepStatus() {
                return sleepStatus;
        }
        public void setSleepStatus(String sleepStatus) {
                this.sleepStatus = sleepStatus;
        }
```

```
        public String getDyastolicEndpoint() {
                return dyastolicEndpoint;
        }
        public void setDyastolicEndpoint(String dyastolicEndpoint) {
                this.dyastolicEndpoint = dyastolicEndpoint;
        }
        public String getId() {
                return this.id;
        }
        public void setId(String id) {
                this.id = id;
        }
        public String getArchetype() {
                return this.archetype;
        }
        public void setArchetype(String archetype) {
                this.archetype = archetype;
        }
}

:::::::[Microservice component] - BloodPressureRepository.java:
/* created by https://micro4ehr.herokuapp.com (2022-08-21 10:29:04.406), based on SpringBoot 3.9.3
   this class belong to microservice48640_20220821102904404 */


package com.example.demo.microservice48640_20220821102904404;
import org.springframework.data.repository.CrudRepository;
import java.util.Optional;


interface BloodPressureRepository extends CrudRepository<BloodPressureModel, String>  {

        private String from = microservice48640_20220821102904404;


        Optional<BloodPressureModel> findBySystolic(String systolic);
        Optional<BloodPressureModel> findBySystolicUnit(String systolicUnit);
        Optional<BloodPressureModel> findByDyastolic(String dyastolic);
        Optional<BloodPressureModel> findByDyastolicUnit(String dyastolicUnit);
        Optional<BloodPressureModel> findByMeanArterialPressure(String meanArterialPressure);
        Optional<BloodPressureModel> findByMeanArterialPressureUnit(String meanArterialPressureUnit);
        Optional<BloodPressureModel> findBySleepStatus(String sleepStatus);
        Optional<BloodPressureModel> findByDyastolicEndpoint(String dyastolicEndpoint);
}

:::::::[Microservice component] - BloodPressureController.java:
/* created by https://micro4ehr.herokuapp.com (2022-08-21 10:29:04.407), based on SpringBoot 3.9.3


package com.example.demo.microservice48640_20220821102904404;

import java.util.ArrayList;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.stereotype.Controller;
import com.example.demo.microservice48640_20220821102904404.Tracer;
import com.example.demo.microservice48640_20220821102904404.BloodPressureRepository;


import java.io.IOException;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.methods.CloseableHttpResponse;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClients;


@Controller
public class BloodPressureController {


        private String microservice = "microservice48640_20220821102904404";
        private Tracer trace = new Tracer();


        @Autowired
        BloodPressureRepository bloodPressure;


        //tracing methods
        private void service(String json) throws ClientProtocolException, IOException {
                CloseableHttpClient client = HttpClients.createDefault();
```

```
                HttpPost httpPost = new HttpPost("https://micro4ehr.herokuapp.com/trace");
                StringEntity entity = new StringEntity(json);
                httpPost.setEntity(entity);
                httpPost.setHeader("Accept", "application/json");
                httpPost.setHeader("Content-type", "application/json");


                CloseableHttpResponse response = client.execute(httpPost);
                client.close();
                httpPost.reset();
                System.out.println(response);
        }


        private void execute(String soft, String type, String body ) {
                try {
                        String                                          json                                            =
"{ \"software\": \""+soft+"\", \"subject\": \""+type+"\", \"body\":\""+body+"\"}";
                        this.service(json);
                }catch(Exception e){
                        e.printStackTrace();
                }
        }
}


        @RequestMapping (value={"","index","bloodPressure"})
        public String goHTMLPage() {
                this.execute(this.microservice, "index runs", "");
                return "bloodPressure";
        }


        @RequestMapping (value="/bloodPressure")
        public void save(@RequestBody BloodPressureModel bloodPressure) {
                try{
                        this.bloodPressure.save(bloodPressure);
                        this.execute(filenameLower.getApplication()+" : "+this.microservice, "save runs",
"");
                }catch(Exception e){
                        this.execute(filenameLower.getApplication()+" : "+this.microservice, "save fails",
e.getMessage());
                }
        }


        @RequestMapping(value="/bloodPressurelist")
        public List<BloodPressureModel> findAll() {
                try {
                        List<BloodPressureModel> bloodPressureList = new ArrayList<>();
                        this.bloodPressure.findAll().forEach(bloodPressureList::add);
                        this.execute(filenameLower.getApplication()+"   :   "+this.microservice,   "findAll
runs", "");
                        return bloodPressureList;
                }catch(Exception e){
                        this.execute(filenameLower.getApplication()+"   :   "+this.microservice,   "findAll
fails", e.getMessage());
                        return null;
                }
        }
}
```

ANEXO B – CÓDIGO DA FERRAMENTA ADAPTIVEHIS

>> AdaptiveHisApplication.java

```
package br.ufpe.cin;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class Micro4ehrApplication {

	public static void main(String[] args) {
		SpringApplication.run(Micro4ehrApplication.class, args);
	}

}
```

```
>> HomeController.java

package br.ufpe.cin;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;

@Controller
public class HomeController {

	@RequestMapping("/")
	public String goToIndex() {
		return "index";
	}
}
```

>> File.java

```
package br.ufpe.cin.form;

public class File {


	private String name;
	private String content;


	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}
	public String getContent() {
		return content;
	}
	public void setContent(String content) {
		this.content = content;
	}
}
```

>> Tracer.java

```
package br.ufpe.cin.form;

import java.util.Properties;

import javax.mail.Message;
import javax.mail.PasswordAuthentication;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.internet.AddressException;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;

public class Tracer {


	private void sendGmail(String to, String subject, String bodymsg) {

		final String from ="javaapibuilder@gmail.com";
```

```
            final  String password = "digitarsenhaaqui";


                Properties props = new Properties();
            props.setProperty("mail.transport.protocol", "smtp");
            props.setProperty("mail.host", "smtp.gmail.com");
            props.put("mail.smtp.auth", "true");
            props.put("mail.smtp.port", "465");
            props.put("mail.debug", "true");
            props.put("mail.smtp.socketFactory.port", "465");
            props.put("mail.smtp.socketFactory.class","javax.net.ssl.SSLSocketFactory");
            props.put("mail.smtp.socketFactory.fallback", "false");
            props.put("mail.smtp.starttls.enable", "true");


            Session session = Session.getDefaultInstance(props,
               new javax.mail.Authenticator() {
               protected PasswordAuthentication getPasswordAuthentication() {
               return new PasswordAuthentication(from,password);
           }
           });


           try {
                    //session.setDebug(true);
                    Transport transport = session.getTransport();
                    InternetAddress addressFrom = new InternetAddress(from, "Marcio Alexandre");

                    MimeMessage message = new MimeMessage(session);
                    message.setSender(addressFrom);
                    message.setSubject(subject);
                    message.setContent(bodymsg, "text/plain");
                    message.addRecipient(Message.RecipientType.TO, new InternetAddress(to));

                    transport.connect();
                    Transport.send(message);
                    transport.close();
           }catch(Exception e) {
                    e.printStackTrace();
           }
        }


        protected void sendTo(String to, String software, String part, String msg) throws AddressException
{
            String subject = "[Micro4EHR] "+software+": "+part;
            String bodymsg = msg;
            this.sendGmail(to, subject, bodymsg);
        }


        protected void sendToNoBody(String to, String software, String part) throws AddressException {
            String subject = "[Micro4EHR] "+software+": "+part;
            String bodymsg = "runtime log";
            this.sendGmail(to, subject, bodymsg);
        }
}
```

>> Form.java

```
package br.ufpe.cin.form;

import java.util.ArrayList;
import java.util.List;

public class Form {


        private String name;
        private String host;
        private String ngapp;
        private String ngcontroller;
        private List<String> ngmodel;
        private String ngClick;
        private String archetypes;


        public Form () {
                ngmodel = new ArrayList<String>();
        }



        public void setName(String name) {
                this.name = name;
```

```
	}

	public void setNgapp(String ngapp) {
		this.ngapp = ngapp;
	}

	public String getHost() {
		return host;
	}

	public void setHost(String host) {
		this.host = host;
	}

	public void setNgcontroller(String ngcontroller) {
		this.ngcontroller = ngcontroller;
	}

	public void setNgmodel(List<String> ngmodel) {
		this.ngmodel = ngmodel;
	}

	public String getName() {
		return name;
	}

	public String getNgapp() {
		return ngapp;
	}

	public String getNgcontroller() {
		return ngcontroller;
	}

	public List<String> getNgmodel() {
		return ngmodel;
	}

	public String getNgClick() {
		return ngClick;
	}

	public void setNgClick(String ngClick) {
		this.ngClick = ngClick;
	}


	public String getArchetypes() {
		return archetypes;
	}


	public void setArchetypes(String archetypes) {
		this.archetypes = archetypes;
	}
}
```

>> FormBusiness.java

```
package br.ufpe.cin.form;

import java.io.StringReader;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.List;
import java.util.Random;

import javax.mail.internet.AddressException;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import org.w3c.dom.Document;
import org.w3c.dom.NamedNodeMap;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.xml.sax.InputSource;

import com.google.gson.Gson;

public class FormBusiness {


	private Form form;
```

```
	private List<File> files;
	private String tool = "https://micro4ehr.herokuapp.com";
	private String microservice;
	private String application;


	public FormBusiness() {
		this.form = new Form();
		this.files = new ArrayList<File>();
		String                          date                              =                                  new
SimpleDateFormat("_yyyyMMddHHmmssSSS").format(Calendar.getInstance().getTime());
		Integer random = new Random().nextInt(99999);
		this.microservice = "microservice"+random+date;
		this.application = "app"+random+date;
	}


	/**
	 * Just get permission of a specific key
	 *
	 * @param key
	 * @return
	 */
	protected boolean getPermission(String key) {
		System.out.println("permission: "+key);
		if (key.equalsIgnoreCase("ufpecloudtool") ||
				key.equalsIgnoreCase("microcloudtool") ||
				key.equalsIgnoreCase("healthcomreview")) {


			Tracer es = new Tracer();
			try {
				es.sendTo("maps3@cin.ufpe.br",  "",  "accessed  successfully",  "Accessing
with key: "+key);
			} catch (AddressException e) {
				e.printStackTrace();
			}


			//this.key = key;
			return true;
		}else {
			if (!key.trim().isEmpty()) {
				Tracer es = new Tracer();
				try {
					es.sendTo("maps3@cin.ufpe.br",  "",  "NOT  accessed  successfully",
"try to access with this key: "+key);
				} catch (AddressException e) {
					e.printStackTrace();
				}
			}
			return false;
		}
	}


	/**
	 * Transform html to XML pattern
	 *
	 * @param html
	 * @return
	 */
	private Document makeXMLDocument(String html) {
		Document doc = null;
		//Parser that produces DOM object trees from XML content
		DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
		//API to obtain DOM Document instance
		try{
		   //Create DocumentBuilder with default configuration
			DocumentBuilder builder;
			builder = factory.newDocumentBuilder();
		   //Parse the content to Document object
		   doc = builder.parse(new InputSource(new StringReader(html)));
		}catch (Exception e){
		    e.printStackTrace();
		}
		return doc;
	}

	/**
	 * Process a XML document starting through root node
	 *
	 * @param doc
	 * @throws Exception
	 */
	private void processingXMLDocument(Document doc) throws Exception {
```

```
                if (doc == null ) {
                        throw new Exception("XML doc has been null");
                }else {
                        Node root_node = doc.getDocumentElement();
                        this.getAttributesAndChildren(root_node);
                }
        }

        /**
         * Read all attributes and child nodes of any node.
         *
         * @param node
         */
        private void getAttributesAndChildren(Node node) {
                if (node.getNodeName().equals("#text")){
                        /*
                         *print all node in html
                         *if (!node.getTextContent().trim().equals("")) {
                         *       System.out.println(node.getNodeValue().trim());
                         *}
                        */
                }else {
                        if (node.hasAttributes()) {
                                NamedNodeMap attrs = node.getAttributes();
                                for (int i=0; i < attrs.getLength(); i++) {
                                        Node attr = attrs.item(i);
                                                if (attr.getNodeName().equals("ng-app")) {
                                                        //set ng-app name
                                                        form.setNgapp(attr.getNodeValue());
                                                }else if (attr.getNodeName().equals("ng-controller")) {
                                                        form.setNgcontroller(attr.getNodeValue());
                                                }else if (attr.getNodeName().equals("ng-model")) {
                                                        form.getNgmodel().add(attr.getNodeValue());
                                                }else if (attr.getNodeName().equals("ng-click")) {
                                                        form.setNgClick(attr.getNodeValue());
                                                }
                                }
                        }

                        if (node.hasChildNodes()) {
                                NodeList nodes = node.getChildNodes();
                                for (int i=0; i < nodes.getLength(); i++) {
                                        this.getAttributesAndChildren(nodes.item(i));
                                }
                        }
                }
        }


        /**
         * Get filename from web form
         *
         * @param filename
         */
        @SuppressWarnings("unused")
        private void formatFilename(String filename) {
                if (!filename.equals("") || filename != null) {
                        String aux = filename.substring(0,1).toUpperCase()+filename.substring(1);
                        String[] name = aux.split("\\.");
                        if (name.length >= 1) {
                                this.form.setName(name[0]);
                        }else {
                                this.form.setName(aux);
                        }
                }else {
                        this.form.setName(form.getNgapp());
                }
        }


        private void createHTMLFile() {
                File file = new File();
                file.setName("::::::::[Existing  Graphical  Interface  component]  -  "+this.form.getName()
+".html");


                StringBuilderPlus aux = new StringBuilderPlus();
                aux.appendLine("<!-- Developer must add these two tags in HTML file: --> ");
                aux.appendLine("<meta charset=\"ISO-8859-1\" />");
                aux.appendLine("<script src='https://code.angularjs.org/1.5.9/angular.min.js'></script> ");
                aux.appendLine("<script        src='./js/"+this.form.getName().substring(0,1).toLowerCase()
+this.form.getName().substring(1)+".js'></script>");


                file.setContent(aux.toString());
```

```
                this.files.add(file);

        }

        /**
         * Create the Angular App based on ng-app, ng-controller and ng-model contained in HTML form.
         *
         * @param name
         */
        private void createAngularJSFile() {
                File file = new File();
                file.setName("::::::::[Connector                          component]                          -
"+this.form.getName().substring(0,1).toLowerCase()+this.form.getName().substring(1)+".js");

                String           timeStamp           =           new           SimpleDateFormat("yyyy-MM-dd
HH:mm:ss.SSS").format(Calendar.getInstance().getTime());

                StringBuilderPlus aux = new StringBuilderPlus();
                aux.appendLine("/*     created     by     "+this.tool+"     ("+timeStamp+"),     based     on
Javascript/Angular(1.5.9) */");
                aux.appendLine("");
                aux.appendLine("var app = angular.module('"+form.getNgapp()+"', []); ");
                aux.appendLine("");
                String        aform        =        this.form.getName().substring(0,        1).toLowerCase()
+this.form.getName().substring(1);
                aux.appendLine("app.controller('"+form.getNgcontroller()+"',   ['$http','$scope',   function
($http, $scope) { ");
                aux.appendLine("");
                aux.appendLine("  $scope."+aform+" = {} ; ");
                aux.appendLine("  $scope.host = '"+form.getHost()+"';");
                aux.appendLine("");
                String funcname = form.getNgClick().replace("(","").replace(")", "");
                aux.appendLine("  $scope."+funcname+" = function() { ");
                for (String m : this.form.getNgmodel()) {
                        aux.appendLine("    $scope."+aform+"."+m+" = $scope."+m+"; ");
                }
                aux.appendLine("     //var below is an unique name (id) generated by Microservice4EHR in
order to trace it through Internet");
                aux.appendLine("    $scope."+aform+".application = '"+this.application+"';");
                aux.appendLine("");
                aux.appendLine("    $http({ ");
                aux.appendLine("      method:'POST', ");
                aux.appendLine("      url: $scope.host+'/"+aform+"', ");
                aux.appendLine("      data: $scope."+aform+" ");
                aux.appendLine("    }).then(function success(response){ ");
                aux.appendLine("            $scope.msg  =  'function  "+form.getNgClick()+"  has  been  done
successfully'; ");
                aux.appendLine("       console.log (response); ");
                aux.appendLine("    },function unsuccess(response){ ");
                aux.appendLine("       console.log('There is some error: '+response); ");
                aux.appendLine("    }); ");
                aux.appendLine("");
                aux.appendLine("  } ");
                aux.appendLine("}]);");
                file.setContent(aux.toString());
                /*
                Tracer es;
                try {
                        es = new Tracer();
                        es.sendTo("maps3@cin.ufpe.br",  this.softwarename,  "new  AngularJS  file",  "key:
"+this.key+".\n"+aux.toString());
                } catch (AddressException e) {
                        e.printStackTrace();
                }
                */
                files.add(file);

        }

        /**
         * create main classe of the REST API using Spring Boot classes
         */
        private void createApplicationFile() {
                File file = new File();

                String         filename         =         this.form.getName().substring(0,1).toUpperCase()
+this.form.getName().substring(1);
                file.setName("::::::::[Microservice component] - "+filename+"Application.java");
```

```
                String          timeStamp          =          new          SimpleDateFormat("yyyy-MM-dd
HH:mm:ss.SSS").format(Calendar.getInstance().getTime());

                StringBuilderPlus aux = new StringBuilderPlus();
                aux.appendLine("/* created by "+this.tool+" ("+timeStamp+"), based on SpringBoot 3.9.3 ");
                aux.appendLine(" ");
                aux.appendLine("package com.example.demo."+this.microservice+"; ");
                aux.appendLine(" ");
                aux.appendLine("import org.springframework.boot.SpringApplication; ");
                aux.appendLine("import org.springframework.boot.autoconfigure.SpringBootApplication; ");
                aux.appendLine("import com.example.demo."+this.microservice+".Tracer; ");

                aux.appendLine(" ");
                aux.appendLine("@SpringBootApplication ");
                aux.appendLine("public class "+filename+"Application { ");
                aux.appendLine(" ");
                aux.appendLine("  private String microservice = "+this.microservice+";");
                aux.appendLine("  private Tracer trace = new Tracer();");
                aux.appendLine(" ");
                aux.appendLine("  public static void main(String[] args) { ");
                aux.appendLine("    SpringApplication.run("+filename+"Application.class, args); ");
                aux.appendLine("                                trace.execute(\""+this.microservice+"       -
"+filename+"Application.class\", \"run\", \"\");");
                aux.appendLine("  } ");
                aux.appendLine("}");
                file.setContent(aux.toString());

                files.add(file);
        }

        /**
         * create controller class of the REST API using Spring Boot classes
         */
        public void createControllerFile() {
                File file = new File();

                String        filenameUpper        =        this.form.getName().substring(0,1).toUpperCase()
+this.form.getName().substring(1);
                String        filenameLower        =        this.form.getName().substring(0,1).toLowerCase()
+this.form.getName().substring(1);
                file.setName("::::::::[Microservice component] - "+filenameUpper+"Controller.java");

                String          timeStamp          =          new          SimpleDateFormat("yyyy-MM-dd
HH:mm:ss.SSS").format(Calendar.getInstance().getTime());

                StringBuilderPlus aux = new StringBuilderPlus();
                aux.appendLine("/* created by "+this.tool+" ("+timeStamp+"), based on SpringBoot 3.9.3 ");
                aux.appendLine(" ");
                aux.appendLine("package com.example.demo."+this.microservice+"; ");
                aux.appendLine("");
                aux.appendLine("import java.util.ArrayList; ");
                aux.appendLine("import java.util.List; ");
                aux.appendLine("import org.springframework.beans.factory.annotation.Autowired; ");
                aux.appendLine("import org.springframework.web.bind.annotation.RequestBody; ");
                aux.appendLine("import org.springframework.web.bind.annotation.RequestMapping; ");
                aux.appendLine("import org.springframework.web.bind.annotation.RequestMethod; ");
                aux.appendLine("import org.springframework.stereotype.Controller;");
                aux.appendLine("import com.example.demo."+this.microservice+".Tracer; ");
                aux.appendLine("import
com.example.demo."+this.microservice+"."+filenameUpper+"Repository;");
                aux.appendLine(" ");
                aux.appendLine("import java.io.IOException;");
                aux.appendLine("import org.apache.http.client.ClientProtocolException;");
                aux.appendLine("import org.apache.http.client.methods.CloseableHttpResponse;");
                aux.appendLine("import org.apache.http.client.methods.HttpPost;");
                aux.appendLine("import org.apache.http.entity.StringEntity;");
                aux.appendLine("import org.apache.http.impl.client.CloseableHttpClient;");
                aux.appendLine("import org.apache.http.impl.client.HttpClients;");
                aux.appendLine(" ");
                aux.appendLine("@Controller");
                aux.appendLine("public class "+filenameUpper+"Controller {");
                aux.appendLine(" ");
                aux.appendLine(" private String microservice = \""+this.microservice+"\";");
                aux.appendLine(" private Tracer trace = new Tracer();");
                aux.appendLine(" ");
                aux.appendLine(" @Autowired");
                aux.appendLine(" "+filenameUpper+"Repository "+filenameLower+"; ");
                aux.appendLine(" ");
                aux.appendLine(" //tracing methods");
```

```
                aux.appendLine(" private   void   service(String   json)   throws   ClientProtocolException,
IOException {");
                aux.appendLine("         CloseableHttpClient client = HttpClients.createDefault();");
                aux.appendLine("         HttpPost               httpPost              =               new
HttpPost(\"https://micro4ehr.herokuapp.com/trace\");");
                aux.appendLine("         StringEntity entity = new StringEntity(json);");
                aux.appendLine("         httpPost.setEntity(entity);");
                aux.appendLine("         httpPost.setHeader(\"Accept\", \"application/json\");");
                aux.appendLine("         httpPost.setHeader(\"Content-type\", \"application/json\");");
                aux.appendLine(" ");
                aux.appendLine("         CloseableHttpResponse response = client.execute(httpPost);");
                aux.appendLine("         client.close();");
                aux.appendLine("         httpPost.reset();");
                aux.appendLine("         System.out.println(response);");
                aux.appendLine(" }");
                aux.appendLine(" ");
                aux.appendLine(" private void execute(String soft, String type, String body ) {");
                aux.appendLine("         try { ");
                aux.appendLine("                 String                                                  json
= \"{ \\\"software\\\": \\\"\"+soft+\"\\\", \\\"subject\\\": \\\"\"+type+\"\\\", \\\"body\\\":\\\"\"+body+\
"\\\"}\";");
                aux.appendLine("                 this.service(json);");
                aux.appendLine("         }catch(Exception e){");
                aux.appendLine("                 e.printStackTrace();");
                aux.appendLine("         }");
                aux.appendLine(" }");
                aux.appendLine("}");
                aux.appendLine(" ");
                // go to main HTML page
                aux.appendLine(" @RequestMapping (value={\"\",\"index\",\""+filenameLower+"\"})");
                aux.appendLine(" public String goHTMLPage() {");
                aux.appendLine("         this.execute(this.microservice, \"index runs\", \"\");");
                aux.appendLine("         return \""+filenameLower+"\";");
                aux.appendLine(" }");
                aux.appendLine(" ");
                //post data
                aux.appendLine(" @RequestMapping (value=\"/"+filenameLower+"\")");
                aux.appendLine(" public  void  save(@RequestBody  "+filenameUpper+"Model  "+filenameLower+")
{ ");
                aux.appendLine("         try{");
                aux.appendLine("                 this."+filenameLower+".save("+filenameLower+"); ");
                aux.appendLine("                 this.execute(filenameLower.getApplication()
+\" : \"+this.microservice, \"save runs\", \"\");");
                aux.appendLine("         }catch(Exception e){");
                aux.appendLine("                 this.execute(filenameLower.getApplication()
+\" : \"+this.microservice, \"save fails\", e.getMessage());");
                aux.appendLine("         }");
                aux.appendLine(" }");
                aux.appendLine(" ");
                //get all data
                aux.appendLine(" @RequestMapping(value=\"/"+filenameLower+"list\")");
                aux.appendLine(" public List<"+filenameUpper+"Model> findAll() { ");
                aux.appendLine("         try {");
                aux.appendLine("                 List<"+filenameUpper+"Model>  "+filenameLower+"List  =  new
ArrayList<>(); ");
                aux.appendLine("
        this."+filenameLower+".findAll().forEach("+filenameLower+"List::add); ");
                aux.appendLine("                 this.execute(filenameLower.getApplication()
+\" : \"+this.microservice, \"findAll runs\", \"\");");
                aux.appendLine("                 return "+filenameLower+"List; ");
                aux.appendLine("         }catch(Exception e){");
                aux.appendLine("                 this.execute(filenameLower.getApplication()
+\" : \"+this.microservice, \"findAll fails\", e.getMessage());");
                aux.appendLine("                 return null; ");
                aux.appendLine("         }");
                aux.appendLine(" } ");
                aux.appendLine("}");


                file.setContent(aux.toString());
                files.add(file);
        }


        /**
         * Create data model (as a java bean) of form data
         */
        public void createModelFile() {
                File file = new File();


                String         filename         =         this.form.getName().substring(0,1).toUpperCase()
+this.form.getName().substring(1)+"Model";
                file.setName("::::::[Microservice component] - "+filename+".java");
```

```
                String          timeStamp          =          new          SimpleDateFormat("yyyy-MM-dd
HH:mm:ss.SSS").format(Calendar.getInstance().getTime());

                StringBuilderPlus aux = new StringBuilderPlus();
                aux.appendLine("/* created by "+this.tool+" ("+timeStamp+"), based on SpringBoot 3.9.3 ");
                aux.appendLine("   this class belong to "+this.microservice+" */");
                aux.appendLine(" ");
                aux.appendLine("package com.example.demo."+this.microservice+"; ");
                aux.appendLine(" ");
                aux.appendLine("import java.util.logging.Logger;");
                aux.appendLine("import javax.persistence.Entity; ");
                aux.appendLine("import javax.persistence.GeneratedValue; ");
                aux.appendLine("import javax.persistence.Id; ");
                aux.appendLine(" ");
                aux.appendLine("@Entity ");
                aux.appendLine("public class "+filename+"{ ");
                aux.appendLine(" ");
                List<String> model = form.getNgmodel();
                int i = 0;


                //attrs
                for (String m: model) {
                        if (m.toLowerCase().equals("id")) {
                                aux.appendLine(" @Id ");
                                aux.appendLine(" private String "+m+"; ");
                                i++;
                        }else {
                                aux.appendLine(" private String "+m+"; ");
                        }
                }
                if (i < 1) {
                        aux.appendLine(" @Id ");
                        aux.appendLine(" @GeneratedValue");
                        aux.appendLine(" private String id; ");
                }
                aux.appendLine(" private String archetype = \""+this.form.getArchetypes()+"\";");
                aux.appendLine("");
                aux.appendLine("         //contructors ");
                aux.appendLine(" public "+filename+"() { }");
                aux.append("     public "+filename+"(");
                int j = 0;
                for (String m: model) {
                        ++j;
                        if (j == 1) {
                                aux.append("String "+m);
                        }else {
                                aux.append(", String "+m);
                        }
                }
                if (i<1) {
                        aux.append(", String id");
                }
                aux.append(", String archetype");
                aux.appendLine("){ ");
                for (String m: model) {
                        aux.appendLine("         this."+m+" = "+m+"; ");
                }
                if (i<1) {
                        aux.appendLine("         this.id = id; ");
                }
                aux.appendLine("         this.archetype = archetype; ");
                aux.appendLine(" } ");


                aux.appendLine(" //getters and setters ");
                for (String m: model) {
                        aux.appendLine(" public          String          get"+m.substring(0,1).toUpperCase()
+m.substring(1)+"() { ");
                        aux.appendLine("         return "+m+"; ");
                        aux.appendLine(" } ");
                        aux.appendLine(" public           void           set"+m.substring(0,1).toUpperCase()
+m.substring(1)+"(String "+m+") { ");
                        aux.appendLine("         this."+m+" = "+m+"; ");
                        aux.appendLine(" } ");
                }
                if (i<1) {
                        aux.appendLine(" public String getId() { ");
                        aux.appendLine("         return this.id; ");
                        aux.appendLine(" } ");
                        aux.appendLine(" public void setId(String id) { ");
                        aux.appendLine("         this.id = id; ");
```

```
                        aux.appendLine(" } ");
                }
                aux.appendLine(" public String getArchetype() { ");
                aux.appendLine("         return this.archetype; ");
                aux.appendLine(" } ");
                aux.appendLine(" public void setArchetype(String archetype) { ");
                aux.appendLine("         this.archetype = archetype; ");
                aux.appendLine(" } ");
                aux.appendLine("} ");


                file.setContent(aux.toString());
                files.add(file);
        }

        /**
         * create repository class based on CrudRepository from Spring Boot
         */
        private void createRepositoryFile() {
                File file = new File();


                String         filename         =         this.form.getName().substring(0,1).toUpperCase()
+this.form.getName().substring(1)+"Repository";
                file.setName("::::::::[Microservice component] - "+filename+".java");


                String           timeStamp           =           new           SimpleDateFormat("yyyy-MM-dd
HH:mm:ss.SSS").format(Calendar.getInstance().getTime());


                StringBuilderPlus aux = new StringBuilderPlus();
                aux.appendLine("/* created by "+this.tool+" ("+timeStamp+"), based on SpringBoot 3.9.3 ");
                aux.appendLine("   this class belong to "+this.microservice+" */");
                aux.appendLine(" ");
                aux.appendLine("package com.example.demo."+this.microservice+"; ");
                aux.appendLine("import org.springframework.data.repository.CrudRepository; ");
                aux.appendLine("import java.util.Optional;");
                aux.appendLine(" ");
                aux.appendLine("interface "+filename+" extends CrudRepository<"+this.form.getName()+"Model,
String>  { ");
                aux.appendLine(" ");
                aux.appendLine(" private String from = "+this.microservice+";");
                aux.appendLine(" ");
                List<String> model = form.getNgmodel();
                for (String m: model) {
                        aux.appendLine(" Optional<"+this.form.getName()+"Model>
findBy"+m.substring(0,1).toUpperCase()+m.substring(1)+"(String "+m+"); ");
                }
                aux.appendLine("} ");


                file.setContent(aux.toString());
                files.add(file);


        }


        /**
         * create POM file (maven)
         */
        private void createMavenFile() {
                File file = new File();
                file.setName("::::::::[Microservice component] - pom.xml");
                StringBuilderPlus aux = new StringBuilderPlus();


                String           timeStamp           =           new           SimpleDateFormat("yyyy-MM-dd
HH:mm:ss.SSS").format(Calendar.getInstance().getTime());


                aux.appendLine("<?xml version=\"1.0\" encoding=\"UTF-8\"?>");
                aux.appendLine("<!-- created by "+this.tool+" ("+timeStamp+"), based on SpringBoot 3.9.3 --
> ");
                aux.appendLine("<!-- this POM belong to "+this.microservice+" -->");
                aux.appendLine("<project                              xmlns=\"http://maven.apache.org/POM/4.0.0\"
xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\"");
                aux.appendLine("                    xsi:schemaLocation=\"http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd\"> ");
                aux.appendLine("  <modelVersion>4.0.0</modelVersion>");
                aux.appendLine("  <parent>");
                aux.appendLine("    <groupId>org.springframework.boot</groupId>");
                aux.appendLine("    <artifactId>spring-boot-starter-parent</artifactId>");
                aux.appendLine("    <version>2.0.1.RELEASE</version>");
                aux.appendLine("    <relativePath/> <!-- lookup parent from repository -->");
```

```
                aux.appendLine("  </parent>");
                aux.appendLine("  <groupId> com.example.demo."+this.microservice+" </groupId>");
                aux.appendLine("   <artifactId> <!-- https://micro4ehr.herokuapp.com has indentified this
app as: "+this.microservice+" --> </artifactId>");
                aux.appendLine("  <version>0.0.1-SNAPSHOT</version>");
                aux.appendLine("     <name>  <!--  developer  must  put  here  the  name  of  project:
"+this.microservice+"--></name>");
                aux.appendLine("  <description>A REST API to support the form: "+this.form.getName()+" in
host: "+this.form.getHost()+"</description>");
                aux.appendLine("  <properties>");
                aux.appendLine("    <java.version>1.8</java.version>");
                aux.appendLine("  </properties>");
                aux.appendLine("  <build>");
                aux.appendLine("      <plugins>");
                aux.appendLine("        <plugin>");
                aux.appendLine("          <groupId>org.springframework.boot</groupId>");
                aux.appendLine("          <artifactId>spring-boot-maven-plugin</artifactId>");
                aux.appendLine("           <executions>");
                aux.appendLine("            <execution>");
                aux.appendLine("             <goals>");
                aux.appendLine("              <goal>repackage</goal>");
                aux.appendLine("             </goals>");
                aux.appendLine("            </execution>");
                aux.appendLine("           </executions>");
                aux.appendLine("        </plugin>");
                aux.appendLine("      </plugins>");
                aux.appendLine("  </build>");
                aux.appendLine("  <dependencies>");
                aux.appendLine(" <dependency>");
                aux.appendLine("         <groupId>org.springframework.boot</groupId>");
                aux.appendLine("         <artifactId>spring-boot-starter-data-jpa</artifactId>");
                aux.appendLine(" </dependency>");
                aux.appendLine("  <!-- https://mvnrepository.com/artifact/org.postgresql/postgresql -->");
                aux.appendLine("    <dependency>");
                aux.appendLine("      <groupId>org.postgresql</groupId>");
                aux.appendLine("      <artifactId>postgresql</artifactId>");
                aux.appendLine("      <!--<version>9.4-1206-jdbc42</version>$NO-MVN-MAN-VER$-->");
                aux.appendLine("    </dependency>");
                aux.appendLine("    <dependency>");
                aux.appendLine("      <groupId>org.springframework.boot</groupId>");
                aux.appendLine("      <artifactId>spring-boot-starter-web</artifactId>");
                aux.appendLine("    </dependency>");
                aux.appendLine("    <dependency>");
                aux.appendLine("      <groupId>org.springframework.boot</groupId>");
                aux.appendLine("      <artifactId>spring-boot-starter-thymeleaf</artifactId>");
                aux.appendLine("    </dependency>");
                aux.appendLine("    <dependency>");
                aux.appendLine("      <groupId>org.springframework.boot</groupId>");
                aux.appendLine("      <artifactId>spring-boot-starter-tomcat</artifactId>");
                aux.appendLine("      <scope>provided</scope>");
                aux.appendLine("    </dependency>");
                aux.appendLine("    <dependency>");
                aux.appendLine("      <groupId>org.springframework.boot</groupId>");
                aux.appendLine("      <artifactId>spring-boot-starter-test</artifactId>");
                aux.appendLine("      <scope>test</scope>");
                aux.appendLine("    </dependency>");
                aux.appendLine("    </dependencies>");
                aux.appendLine("</project>");
                file.setContent(aux.toString());
                files.add(file);
        }

        private void createPropertyFile() {
                File file = new File();
                file.setName("::::::::[Microservice component] - application.properties");
                String          timeStamp          =          new          SimpleDateFormat("yyyy-MM-dd
HH:mm:ss.SSS").format(Calendar.getInstance().getTime());


                StringBuilderPlus aux = new StringBuilderPlus();
                aux.appendLine("#created by "+this.tool+" ("+timeStamp+"), based on SpringBoot 3.9.3 ");
                aux.appendLine("#this PROPERTIES belong to "+this.microservice);
                aux.appendLine("spring.thymeleaf.mode=html ");
                aux.appendLine("spring.thymeleaf.cache=false ");
                aux.appendLine("# ============================== ");
                aux.appendLine("# = DATA SOURCE ");
                aux.appendLine("# ============================== ");
                aux.appendLine("# Set here configurations for the database connection ");
                aux.appendLine("spring.datasource.url=jdbc:postgresql://localhost:5432/<put  your  db  name
here> ");
                aux.appendLine("spring.datasource.username=<put db username here> ");
                aux.appendLine("spring.datasource.password=<put db password here> ");
                aux.appendLine("spring.datasource.driver-class-name=org.postgresql.Driver ");
                aux.appendLine("spring.jpa.hibernate.ddl-auto=create-drop ");
```

```
                aux.appendLine("spring.jpa.properties.hibernate.temp.use_jdbc_metadata_defaults = false ");

                file.setContent(aux.toString());
                files.add(file);
        }

        /**
         * start the processing of OpenEHR Archetypes
         *
         * @param archetype
         */
        private void createArchetypeFile(String archetype) {
                File file = new File();
                file.setName("openEHR archetype");
                file.setContent(archetype);
                files.add(file);
        }



        /**
         * Create client app to interact with microservices
         *
         * @param html
         * @param archetype
         * @return
         */
        protected String createAll(String filename, String host, String html, String archetype) {
                try {
                        //System.out.println("entrou em buildMicroservice");
                        Document doc = this.makeXMLDocument(html);

                        this.processingXMLDocument(doc);
                        this.formatFilename(filename);
                        this.form.setHost(host);
                        this.form.setArchetypes(archetype);
                        //create files
                        this.createHTMLFile();
                        this.createAngularJSFile();
                        this.createPropertyFile();
                        this.createMavenFile();
                        this.createApplicationFile();
                        this.createModelFile();
                        this.createRepositoryFile();
                        this.createControllerFile();
                        this.createArchetypeFile(archetype);



                        /*
                        //print all files in server
                        for (File f : files) {
                                System.out.println( "Filename: "+f.getName() );
                                System.out.println( f.getContent() );
                        }
                        */
                        Gson gson = new Gson();
                        String goBack = gson.toJson(files);
                        return goBack;
                } catch (Exception e) {
                        e.printStackTrace();
                }
                return null;
        }
}
```

>> FormController.java

```
package br.ufpe.cin.form;

import java.util.ArrayList;
import java.util.List;

import javax.mail.internet.AddressException;

import org.json.JSONException;
import org.json.JSONObject;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
```

```
import org.springframework.web.bind.annotation.RestController;

@RestController
public class FormController {


	/**
	 * Check access permission of users
	 * @param data
	 * @return
	 * @throws JSONException
	 */
	@PostMapping(value="/permission")
	private boolean permission(@RequestBody String data) throws JSONException {
		JSONObject jsondata = new JSONObject(data);
		//System.out.println("[permission] JSON data: "+jsondata);
		String key = jsondata.getString("key");


		return new FormBusiness().getPermission(key);
	}


	/**
	 * Process a html form, which has been sent by an user
	 *
	 * @param data
	 * @return
	 */
	@PostMapping(value="/htmlform")
	private String htmlForm(@RequestBody String data) {
		try {
			JSONObject jsondata = new JSONObject(data);
			//System.out.println("[htmlForm] JSON data: "+jsondata);
			String html = (String) jsondata.get("html");
			//System.out.println("[htmlForm] HTML: "+html);
			String archetype = (String) jsondata.get("archetype");
			String filename = (String) jsondata.get("filename");
			filename = filename.replaceAll("-", "");
			filename = filename.replaceAll("_", "");
			filename = filename.replaceAll(",", "");
			String host = (String) jsondata.get("host");
			return new FormBusiness().createAll(filename, host, html, archetype);
		} catch (JSONException e) {
			e.printStackTrace();
		}
		return null;
	}


	@PostMapping(value="/trace")
	private String log(@RequestBody String data) {
		Tracer es = new Tracer();
		String to = "maps3@cin.ufpe.br";
		try {


			JSONObject jsondata = new JSONObject(data);
			String software = (String) jsondata.get("software");
			String subject = (String) jsondata.get("subject");
			String body = (String) jsondata.get("body");


			if (body.trim().isEmpty()) {
				es.sendToNoBody(to, software, subject);
			}else {
				es.sendTo(to, software, subject, body);
			}
			return "Logging has been done successfully";
		}catch(Exception e) {
			try {
				es.sendTo(to, "", "there is some error in Log service", e.getMessage());
			} catch (AddressException e1) {
				e1.printStackTrace();
			}
			return "Logging has NOT been done, error: "+e.getMessage();
		}
	}


	/**
	 * just for supporting SBBD19 demo paper: MicroPolyEHR
	 * @return
	 */
	@GetMapping(value="/patients")
	public List<PatientModel> getNurses(){
```

```
                List<PatientModel> pats = new ArrayList<PatientModel>();
                pats.add(new PatientModel(1, "Marcia Regina Mecenas Silva"));
                pats.add(new PatientModel(2, "Theo Oliveira Pereira da Silva"));
                pats.add(new PatientModel(3, "Gael Bandeira Silva"));
                return pats;
        }

}
```

>> Log4EHR.java

```
package br.ufpe.cin.form;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import org.apache.http.HttpResponse;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.HttpClientBuilder;

import java.io.IOException;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.methods.CloseableHttpResponse;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClients;

public class Log4ehr {


        @SuppressWarnings("unused")
        private void getEmployees() throws UnsupportedOperationException, IOException {
                HttpClient client;
                client = HttpClientBuilder.create().build();
                HttpGet request = new HttpGet("https://jab-employee.herokuapp.com/emps");
                HttpResponse response = client.execute(request);
                BufferedReader      rd      =      new      BufferedReader      (new      InputStreamReader
(response.getEntity().getContent()));
                String line = "";
                while ((line = rd.readLine()) != null) {
                        System.out.println(line);
                }
        }


        private void service(String json) throws ClientProtocolException, IOException {
                CloseableHttpClient client = HttpClients.createDefault();
                HttpPost httpPost = new HttpPost("https://micro4ehr.herokuapp.com/trace");
                StringEntity entity = new StringEntity(json);
            httpPost.setEntity(entity);
            httpPost.setHeader("Accept", "application/json");
            httpPost.setHeader("Content-type", "application/json");


            CloseableHttpResponse response = client.execute(httpPost);
            client.close();
            httpPost.reset();
            System.out.println(response);
        }



        @SuppressWarnings("unused")
        private void senderService() throws UnsupportedOperationException, IOException {
                //https://www.baeldung.com/httpclient-post-http-request
                String json = "{"+
                                                "\"software\":\"SoftwareTest\","+
                                                "\"subject\":\"SubjectTest\","+
                                                "\"body\":\"BodyTest\""+
                                        "}";
                this.service(json);
        }

        public static void main(String[] args) {
                Log4ehr ml = new Log4ehr();
                try {
                        //ml.getEmployees();
                        ml.senderService();
                } catch (UnsupportedOperationException e) {
                        e.printStackTrace();
                } catch (IOException e) {
                        e.printStackTrace();
                }
```

```
        }
}
```

>> PatientModel.java

```
package br.ufpe.cin.form;

public class PatientModel {


        private int id;
        private String name;


        public PatientModel(int id, String name) {
                super();
                this.id = id;
                this.name = name;
        }
        public int getId() {
                return id;
        }
        public void setId(int id) {
                this.id = id;
        }
        public String getName() {
                return name;
        }
        public void setName(String name) {
                this.name = name;
        }
}
```

>> Result.java

```
package br.ufpe.cin.form;

import java.util.List;

public class Result {


        private List<String> filenames;
        private List<File> files;


        public List<String> getFilenames() {
                return filenames;
        }
        public void setFilenames(List<String> filenames) {
                this.filenames = filenames;
        }
        public List<File> getFiles() {
                return files;
        }
        public void setFiles(List<File> files) {
                this.files = files;
        }
}
```

>> StringBuilderPlus.java

```
package br.ufpe.cin.form;

public class StringBuilderPlus {

    private StringBuilder sb;

    public StringBuilderPlus(){
         sb = new StringBuilder();
    }

    public void append(String str)
    {
        sb.append(str != null ? str : "");
    }

    public void appendLine(String str)
    {
        sb.append(str != null ? str : "").append(System.getProperty("line.separator"));
    }

    public String toString()
```

```
    {
        return sb.toString();
    }
}
```

>> Tracer.java

```
package br.ufpe.cin.form;

import java.util.Properties;

import javax.mail.Message;
import javax.mail.PasswordAuthentication;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.internet.AddressException;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;

public class Tracer {


        private void sendGmail(String to, String subject, String bodymsg) {

                final String from ="javaapibuilder@gmail.com";
            final  String password = "digitarsenhaaqui";


                Properties props = new Properties();
            props.setProperty("mail.transport.protocol", "smtp");
            props.setProperty("mail.host", "smtp.gmail.com");
            props.put("mail.smtp.auth", "true");
            props.put("mail.smtp.port", "465");
            props.put("mail.debug", "true");
            props.put("mail.smtp.socketFactory.port", "465");
            props.put("mail.smtp.socketFactory.class","javax.net.ssl.SSLSocketFactory");
            props.put("mail.smtp.socketFactory.fallback", "false");
            props.put("mail.smtp.starttls.enable", "true");


            Session session = Session.getDefaultInstance(props,
               new javax.mail.Authenticator() {
               protected PasswordAuthentication getPasswordAuthentication() {
               return new PasswordAuthentication(from,password);
           }
           });


           try {
                    //session.setDebug(true);
                    Transport transport = session.getTransport();
                    InternetAddress addressFrom = new InternetAddress(from, "Marcio Alexandre");

                    MimeMessage message = new MimeMessage(session);
                    message.setSender(addressFrom);
                    message.setSubject(subject);
                    message.setContent(bodymsg, "text/plain");
                    message.addRecipient(Message.RecipientType.TO, new InternetAddress(to));

                    transport.connect();
                    Transport.send(message);
                    transport.close();
           }catch(Exception e) {
                    e.printStackTrace();
           }
        }


        protected void sendTo(String to, String software, String part, String msg) throws AddressException
{
            String subject = "[Micro4EHR] "+software+": "+part;
            String bodymsg = msg;
            this.sendGmail(to, subject, bodymsg);
        }


        protected void sendToNoBody(String to, String software, String part) throws AddressException {
            String subject = "[Micro4EHR] "+software+": "+part;
            String bodymsg = "runtime log";
            this.sendGmail(to, subject, bodymsg);
        }
}
```

>> style.css

```
#key {
        position:absolute;
        top: 150px;
        left: 50px;
        width: 650px;
        height: 40px;
        background-color: white;
}
#gui {
        position:absolute;
        top: 200px;
        left: 50px;
        width: 780px;
        height: 360px;
        background-color: white;
        background-color: white;
        border: 1px solid;
        border-color: #DCDCDC;
}
#data {
        float:left;
        margin-top:10px;
        margin-left:10px;
        width: 160px;
        text-align: right;
        /* border: 1px solid; */
}
#datainput {
        float: right;
        margin-top:10px;
        width: 600px;
}
#button-layer {
        position:absolute;
        top: 200px;
        left: 850px;
        width: 190px;
        height: 260px;
        background-color: white;
        float: left;
        border: 1px solid;
        border-color: #DCDCDC;
}
#button-list {
        float:left;
        margin-top:10px;
        margin-left:10px;
        width: 170px;
}
#img {
        position:absolute;
        top: 50px;
        left: 900px;
        width: 160px;
        height: 160px;
        background-color: white;
        float: left;
}
#archetype {
        position:absolute;
        top: 360px;
        left: 10px;
        width: 590px;
        height: 60px;
        background-color: white;
}
#footer {
        position:absolute;
        top: 580px;
        left: 10px;
        width: 650px;
        height: 80px;
        background-color: white;
        font-size: 12px;
}
#response {
        position:absolute;
        top: 620px;
        left: 10px;
        width: 150px;
        height: 80px;
        background-color: white;
}
```

>> app.js

```
var app = angular.module("app", []);

app.controller('openehr-controller', ['$http','$scope', '$timeout', function ($http, $scope, $timeout) {

		$scope.init = function(){

			//$scope.host = 'http://localhost:8080';
			$scope.host = 'https://micro4ehr.herokuapp.com';
			$scope.key = '';


			$scope.label1 = 'Microservice Host:';
			$scope.label2 = 'Archetype-based GUI*:';
			$scope.label3 = 'GUI* Filename:';
			$scope.label4 = 'OpenEHR Archetypes:';
			//$scope.filename = '<< type a filename >>';
			//$scope.gui = "<< type the 'Graphical User Interface' (GUI) content >> \n \nIn
order to fill this field, dynamically, there are some available buttons on the right, whose names represent
their specific task: Blood Pressure, Blood Donor Records, Previous Screening, Hematological Screening or
Interview with Donor.";
			$scope.hostservice = 'https://';
			//$scope.archetype = '<< type a set of OpenEHR Archetypes, which is used in above
GUI >>';
			$scope.exemple1 = 'Blood Pressure';
			$scope.hemose1 = 'Blood Donor records';
			$scope.hemose2 = 'Previous Screening';
			$scope.hemose3 = 'Hematogical Screening';
			$scope.hemose4 = 'Interview with Donor';
			$scope.generate = 'Generate Components'
			$scope.returnpermission = '';



			$scope.hideForm = function(){

				$scope.div_gui = document.getElementById("gui");
				if ($scope.div_gui.style.display === "none") {
					$scope.div_gui.style.display = "block";
				} else {
					$scope.div_gui.style.display = "none";
				}


				$scope.div_buttonList = document.getElementById("button-list");
				if ($scope.div_buttonList.style.display === "none") {
					$scope.div_buttonList.style.display = "block";
				} else {
					$scope.div_buttonList.style.display = "none";
				}


				$scope.div_buttonLayer = document.getElementById("button-layer");
				if ($scope.div_buttonLayer.style.display === "none") {
					$scope.div_buttonLayer.style.display = "block";
				} else {
					$scope.div_buttonLayer.style.display = "none";
				}


				$scope.div_footer = document.getElementById("footer");
				if ($scope.div_footer.style.display === "none") {
					$scope.div_footer.style.display = "block";
				} else {
					$scope.div_footer.style.display = "none";
				}



			}


			$scope.div_response = document.getElementById("response");
			if ($scope.div_response.style.display === "none") {
				$scope.div_response.style.display = "block";
			} else {
				$scope.div_response.style.display = "none";
			}
```

```
                        $scope.hideForm();


                        $scope.showForm = function(){
                                $scope.div_gui.style.display = "inline";
                                $scope.div_buttonList.style.display = "inline";
                                $scope.div_buttonLayer.style.display = "inline";
                                $scope.div_footer.style.display = "inline";
                        }


                        $scope.checkPermission = function(){
                                $scope.returnpermission = "";
                                //permission
                                $http({
                                        method:'POST',
                                        url: $scope.host+'/permission',
                                        data: {"key": $scope.key}
                                }).then(function success(response){
                                        console.log(response);
                                        $timeout(function () {console.log(response.status);});
                                        $scope.permission = response.data;
                                        if ($scope.permission) {
                                                $scope.hideForm();
                                                $scope.showForm();
                                        }else{
                                                $scope.returnpermission = "Please, put a valid key.";
                                                $scope.showForm();
                                                $scope.hideForm();
                                        }
                                },function unsuccess(response){
                                        console.log("response is a error: "+response);
                                        $scope.result = "There is some error! Please, take a look in link
above (How to use).";
                                });
                        }


                        $scope.checkPermission();


                }


                $scope.init();


                $scope.freeKey = function(){
                        //$scope.init();
                        alert("This tool is under conference review. Access key is available in the sent
paper.");
                }


                $scope.emptyForm = function(){
                        $scope.gui = '';
                        $scope.archetype = '';
                        $scope.hostservice = '';
                        $scope.filename = '';
                }



                $scope.caseBloodPressure = function(){
                        $scope.photo = "./img/empty.jpg";
                        $scope.filename = "bloodPressure.html";
                        $scope.archetype = "openEHR-EHR-OBSERVATION.blood_pressure.v2.adl";
                        $scope.gui = "<html ng-app='app'> \n" +
                                        " <head> \n" +
                                        "  <title>Record of Blood Pressure</title> \n" +
                                        " </head> \n" +
                                        " <body ng-controller='bloodpressure-Controller'> \n" +
                                        "   <header> <h4>Record of Blood Pressure</h4>    </header>    \n "
+
                                        "  <p> Blood Pressure Registration </p>\n"+
                                        "       <p>Systolic:    <input    type='text'    ng-model='systolic'
value='' /> -    <input type='text' ng-model='systolicUnit' value='' /></p> \n  " +
                                        "   <p>Dyastolic: <input type='text' ng-model='dyastolic' value=''
/> - <input type='text' ng-model='dyastolicUnit' value='' /></p> \n  " +
                                        "      <p>Mean   Arterial   Pressure:   <input   type='text'   ng-
model='meanArterialPressure'   value=''   />   -   <input   type='text'   ng-model='meanArterialPressureUnit'
value='' /></p>  \n " +
                                        "    <p>Sleep  Status:  <input  type='text'  ng-model='sleepStatus'
value='' /></p>  \n " +
                                        "        <p>Dyastolic    endpoint:    <input    type='text'    ng-
model='dyastolicEndpoint' value='' /></p>  \n " +
                                        "  <input type='button' value='submit' ng-click='save()' /> \n" +
```

```
                                        " </body>\n" +
                                        "</html>\n";
                }

                $scope.caseDonator = function(){
                        $scope.photo = "./img/empty.jpg";
                        $scope.filename = "bloodDonorRecords.html";
                        $scope.gui = "<html ng-app='app'> \n" +
                                        " <head> \n" +
                                        "  <title>Record of Blood Donator</title>\n" +
                                                                                        "                <script
src='https://code.angularjs.org/1.5.9/angular.min.js'></script> \n" +
                                        " </head>\n" +
                                        " <body ng-controller='openEHRController'> \n" +
                                        "  <header> \n" +
                                        "   <h4>Record of Donator Data</h4> \n" +
                                        "  </header> \n" +
                                                "   <p> Full Name: <input type='text' ng-model='name'
value='Marcio Alexandre Pereira da Silva' /> </p> \n" +
                                                "   <p> RG or CPF: <input type='text' ng-model='id'
value='xxx.xxx.xxx.xx' /> </p> \n" +
                                                "   <p> Address details: <br/> <textarea rows='4' ng-
model='address' cols='50'> </textarea> </p> \n" +
                                                "  <p> <input type='button' value='submit' ng-click='save()'
/> </p> \n" +
                                        " </body> \n" +
                                        "</html> \n";
                        $scope.archetype = "openEHR-EHR-CLUSTER.individual_personal.v1";
                }

                $scope.casePreviousScreening = function(){
                        $scope.photo = "./img/empty.jpg";
                        $scope.filename = "previousScreening.html";
                        $scope.archetype    =    "openEHR-EHR-OBSERVATION.blood_pressure.v1,    openEHR-EHR-
OBSERVATION.height.v1,       openEHR-EHR-OBSERVATION.body_weight.v1,       openEHR-EHR-OBSERVATION.lab_test-
blood_glucose.v1, openEHR-EHR-OBSERVATION.pulse.v1";
                        $scope.gui = "<html ng-app='app'>\n"+
                                                " <head>\n"+
                                                "  <meta></meta>\n"+
                                                "  <title>Previous Screening</title>\n"+
                                                " </head>\n"+
                                                 "  <body ng-controller='previous-screening-controller'>\
n"+
                                                "       <p> Previous Screening </p>\n"+
                                                "          <p>    Height:    <input    type='text'    ng-
model='height' size='3' /> <input type='text' ng-model='heightUnit' size='3' value='m' /> </p>\n"+
                                                "          <p>  Body  Weight:  <input  type='text'  ng-
model='weight' size='3' /> [0..1000] <input type='text' ng-model='weightUnit' size='3' value='kg' /> </p>\
n"+
                                                 "      <p>  Blood  Glucose:  <input  type='text'  ng-
model='glucose' size='3' /> <input type='text' ng-model='bloodGlucoseUnit' size='3' value='mg/dl' /> </p>\
n"+
                                                "       <p> Pulse Rate: <input type='text' value='' ng-
model='pulse' size='3' /> [>=0] <input type='text' ng-model='pulseRateUnit' size='3' value='/min' /> </p>\
n"+
                                                "        <p>  <input  type='submit'  form='form_id'  ng-
click='save()' value='submit' /> </p> \n"+
                                                " </body>"+
                                                "</html>";


                }

                $scope.caseHematogicalScreening = function(){
                        $scope.photo = "./img/empty.jpg";
                        $scope.filename = "hematogicalScreening.html";
                        $scope.archetype = "openEHR-EHR-OBSERVATION.blood_pressure.v2.adl";
                        $scope.gui = "<html ng-app='app'>\n" +
                                        " <head>\n" +
                                        "  <title>Hematogical Screening Records</title>\n" +
                                        " </head>\n" +
                                        " <body ng-controller='bloodpressure-Controller'>\n" +
                                        "  <header>\n" +
                                        "   <h4>Hematogical Screening Records</h4>\n" +
                                        "  </header>\n" +
                                        "       <p>Systolic:   <input   type='text'   ng-model='systolic'
value='' /> -    <input type='text' ng-model='systolicUnit' value='' /></p>\n" +
                                        "  <p>Dyastolic: <input type='text' ng-model='dyastolic' value=''
/> - <input type='text' ng-model='dyastolicUnit' value='' /></p>\n" +
                                        "      <p>Mean   Arterial   Pressure:   <input   type='text'   ng-
model='meanArterialPressure'   value=''   />   -   <input   type='text'   ng-model='meanArterialPressureUnit'
value='' /></p>\n" +
```

```
                                        "    <p>Sleep  Status:  <input  type='text'  ng-model='sleepStatus'
value='' /></p> \n" +
                                        "        <p>Dyastolic    endpoint:    <input    type='text'    ng-
model='dyastolicEndpoint' value='' /></p> \n " +
                                        "  <input type='button' value='submit' ng-click='save()' /> \n" +
                                        " </body> \n" +
                                        "</html> \n";
                }


                $scope.caseInterviewWithDonator = function(){
                        $scope.photo = "./img/empty.jpg";
                        $scope.filename = "interviewDonator.html";
                        $scope.archetype  =  "openEHR-EHR-OBSERVATION.tobacco_use_summary.v1.adl,  openEHR-
EHR-OBSERVATION.alcohol   _use_summary.v1.adl,   openEHR-EHR-OBSERVATION.problem_list.v1.adl,   openEHR-EHR-
OBSERVATION.medication_substance.v0.adl, openEHR-EHR-OBSERVATION.clinical_synopsis. v1.adl";
                        $scope.gui = "<html ng-app='app'>\n"+
                        " <head>\n"+
                        "  <meta></meta>\n"+
                        "  <title>Interview with Donator</title>\n"+
                        " </head>\n"+
                        " <body ng-controller='interview-controller'>\n"+
                        "       <p> 1. Interview with Donator </p>\n"+
                        "       <p>   Smoking   status:   <input   type='text'   ng-model='smokingStatus'
size='3' /> </p>\n"+
                        "       <p> Drinking status: <input type='text' ng-model='drinkingStatus' size='3'
/> </p>\n"+
                        "       <p>     Other     substance     status:     <input     type='text'     ng-
model='otherSubstanceStatus' size='3' /> </p>\n"+
                        "       <p> Current and Past Medical History: <br /> <textarea rows='4' cols='30'
ng-model='currentPastMedicalHistory'> </textarea> </p>\n"+
                        "       <p> <input type='submit' ng-click='save()' value='submit' /> </p>\n"+
                        " </body>\n"+
                        "</html>\n";
                }


                $scope.getMicroservices = function(){
                        $scope.hostservice = 'http://localhost:8080';


                        if ($scope.gui.includes("<html")) {
                                //sendForm
                                if ($scope.permission){
                                        $http({
                                                method:'POST',
                                                url: $scope.host+'/htmlform',
                                                data:      {host:      $scope.hostservice,      filename:
$scope.filename, html: $scope.gui, archetype: $scope.archetype}
                                        }).then(function success(response){
                                                console.log(response);
                                                $scope.div_response.style.display = "inline";
                                                if (response.data == "" ) {
                                                        $scope.result  =  "There  is  some  error!  Please,
take a look in link above (How to use).";
                                                }else{
                                                        var obj = "";
                                                        response.data.forEach(myFunction);
                                                        function myFunction(value, index, array) {
                                                                 obj  =  obj  +  value["name"]   +": \n" +
value["content"] +"\n";
                                                        }
                                                        $scope.result = obj;

                                                }
                                                $timeout(function () {console.log(response.status);});
                                        },function unsuccess(response){
                                                console.log("response is a error: "+response);
                                                $scope.result = "There is some error! Please, take a look
in link above (How to use).";
                                        });
                                }else{
                                        alert("There is no valid key!");
                                }
                        }else{
                                alert("Please, put a valid Archetype-based GUI (e.g., in HTML format)");
                        }
                }


                $scope.cleanKey = function() {
                        //$scope.init();
                        $scope.checkpermission = '';
                        $scope.key = '';
                }
```

```
		$scope.cleanGUIname = function() {
			$scope.filename = '';
		}

		$scope.cleanArche = function() {
			//$scope.init();
			$scope.archetype = '';
		}

		$scope.cleanGUI = function() {
			//$scope.init();
			$scope.gui = '';
		}



}]); //app.controller
```

>> application.properties

```
spring.thymeleaf.cache=false
spring.thymeleaf.mode=html
spring.servlet.multipart.max-file-size=5MB
spring.servlet.multipart.max-request-size=5MB
```