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

ANDRE MAGNO COSTA DE ARAÚJO

UM FRAMEWORK PARA CRIAÇÃO DE ESQUEMAS DE DADOS, GERAÇÃO DE INTERFACES GRÁFICAS DE USUÁRIO E PERSITÊNCIA POLIGLOTA DO RES UTILIZANDO ARQUÉTIPOS

Recife
2018

André Magno Costa de Araújo

Um Framework para Criação de Esquemas de Dados, Geração de Interfaces Gráficas de Usuário e Persistência Poliglota do RES Utilizando Arquétipos

Tese submetida ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Doutor em Ciência da Computação.

ORIENTADOR (A): **Prof.ª Valéria Cesário Times**

Recife
2018

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

A663f Araújo, André Magno Costa de
Um framework para criação de esquemas de dados, geração de interfaces gráficas de usuário e persistência poliglota do RES utilizando arquétipos / André Magno Costa de Araújo. – 2018.
139 f.: il., fig., tab.

Orientadora: Valéria Cesário Times.
Tese (Doutorado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2018.
Inclui referências.

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

025.04 CDD (23. ed.) UFPE- MEI 2018-094

André Magno Costa de Araújo

“Um Framework para Criação de Esquemas de Dados, Geração de Interfaces Gráficas de Usuário e Persistência Poliglota do RES Utilizando Arquétipos”

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.

Aprovado em: 04/06/2018.

_____

Orientadora: Profa. Valéria Cesário Times

BANCA EXAMINADORA

_____

Prof. Vinícius Cardoso Garcia

Centro de Informática /UFPE

_____

Prof. Fernando José Castor de Lima Filho

Centro de Informática / UFPE

_____

Prof. Luciano de Andrade Barbosa

Centro de Informática / UFPE

_____

Prof. João Eduardo Ferreira

Instituto de Matemática e Estatística / USP

_____

Profa. Vanessa Braganholo Murta

Instituto de Computação / UFF

Dedico o resultado deste esforço aos meus pais, Luís e Mirtes, à minha amada esposa, Glauciane, e às nossas trigêmeas, Alice, Heloísa e Isabela, que estão chegando para alegrar ainda mais as nossas vidas.

AGRADECIMENTOS

Agradeço primeiramente ao autor da vida, por me proporcionar todas as condições necessárias para a realização deste tão sonhado e árduo doutorado. Deus, obrigado por me proteger nas inúmeras viagens realizadas, por fortalecer a minha fé e renovar minha dedicação a cada dificuldade encontrada. Obrigado pelo teu amor e pelo o dom da vida.

Aos meus pais, avós, irmãos e irmãs, agradeço imensamente pelo amor, carinho e pelas orações que me mantiveram firme durante esta longa jornada. À Glauciane, minha amada esposa, obrigado pelo amor, companheirismo e apoio incondicional nos momentos felizes e difíceis. À Jória e sua família, obrigado por me acolher com tanto carinho em sua casa. Ao amigo Marcus Urbano, companheiro de inúmeros artigos e projetos, obrigado pela amizade e ajuda ao longo desta jornada. À Danielle, meus sinceros agradecimentos pela pronta disposição em me ajudar e incentivar nesses longos anos de estudo. Aos amigos, Carlos Andrew e Weberson Ribeiro, registro minha gratidão por estarem sempre dispostos a me ajudar. Aos voluntários que participaram dos estudos realizados nesta tese, meus sinceros agradecimentos.

Aos demais familiares e amigos, obrigado pelo apoio, incentivo e pela convivência harmoniosa ao longo destes tantos anos. Não serei capaz de citá-los individualmente, fica registrado o meu reconhecimento e agradecimento por fazerem parte da minha vida.

À Profª. Valéria, minha eterna gratidão por ter sido além de orientadora, minha amiga e conselheira. Expresso aqui meus sinceros agradecimentos pela oportunidade, por acreditar em mim, por me fazer um ser humano melhor e por me acolher com tanto amor e carinho. Levarei comigo todos os ensinamentos recebidos, e lembrarei sempre com muito carinho desse período da minha vida de muito trabalho, porém de muita admiração, respeito e bom humor.

A Fundação de Amparo à Ciência e Tecnologia do Estado de Pernambuco – FACEPE, agradeço pelo incentivo financeiro proporcionado.

A todos que contribuíram direta ou indiretamente para mais essa vitória, meus sinceros agradecimentos.

RESUMO

O desenvolvimento de sistemas de informação em Saúde (SIS) baseado em arquétipos e templates cria mecanismos de interoperabilidade para o registro eletrônico de saúde (RES), além de melhorar a flexibilidade das aplicações de saúde. Um arquétipo pode ser definido como uma expressão computacional representada por restrições de domínio, que modelam os atributos de dados e dão significado semântico ao RES, enquanto templates representam interfaces gráficas do usuário criadas a partir das especificações definidas nos arquétipos. Arquétipos e templates foram utilizados no setor de saúde 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 de SIS. No entanto, relata-se no estado da arte a falta de ferramentas que construam esquemas de dados para o armazenamento do RES em diferentes sistemas de bancos de dados utilizando arquétipos. Além disso, a construção dinâmica de interfaces gráficas de usuário com recurso de persistência poliglota do RES é relatada pela comunidade científica como um importante mecanismo para melhorar a flexibilidade e a extensibilidade de SIS. Este trabalho propõe um framework chamado de Template4EHR, o qual tem o objetivo de construir esquemas de dados para o armazenamento do RES em bancos de dados relacionais e NoSQL, como também gerar interfaces gráficas de usuário a partir dos atributos de dados, das terminologias e das restrições dos arquétipos. ParaqQ fornecer uma visão conceitual de como construir esquemas de dados utilizando arquétipos, este trabalho especifica um metamodelo em UML que exibe os conceitos e relacionamentos da arquitetura dual para modelar o RES. Um algoritmo especifica como os atributos de dados, as terminologias e as restrições são extraídas dos arquétipos e, um conjunto de regras de mapeamento descrevem como as interfaces gráficas de usuário são geradas. Para validar o framework proposto, testes experimentais foram realizados com profissionais de computação e saúde, e os resultados indicam que template4EHR reduziu em 62% o esforço de codificação de uma aplicação de saúde. Um conjunto de métricas de software foi utilizado para verificar conformidade de Template4EHR com as características de manutenibilidade, flexibilidade e reusabilidade. Além disso, Template4EHR otimizou a criação de esquema de dados e o desenvolvimento de interfaces gráficas com recurso de persistência de dados.

Palavras-chave: Framework. Sistemas de Informação em Saúde. Esquemas de Dados. Persistência Poliglota. Registro Eletrônico de Saúde. Arquétipos.

ABSTRACT

Health Information Systems (HIS) development based on archetypes and templates establishes interoperability mechanisms for the Electronic Health Record (EHR) and improves the flexibility of health applications. An archetype consists of a computational expression represented by domain constraints that model data attributes and give a semantic meaning to the EHR data while Templates are graphical user interfaces built at runtime from archetype specifications. Archetypes and templates were used in the healthcare sector to represent clinical data in legacy system, model the EHR in database systems and specify data requirements and terminologies in health applications. However, there is a lack of tools in the state-of-the-art to build data schemas to store the EHR in different database systems. In addition, the dynamic construction of graphical user interfaces with polyglot persistence is pointed out by the scientific community as an important contribution to improve the flexibility and extensibility of HIS. This work specifies a framework called Template4EHR, which aims to build data schemas to store the EHR in relational and NoSQL databases, as well as generate graphical user interfaces using archetypes. To provide a conceptual view on how to build data schemas using archetypes, this work specifies a UML metamodel that exemplify the concepts and relationships of the openEHR architecture to model the EHR. An algorithm specifies how data attributes, terminologies and constraints are extracted while a set of mapping rules describe how the graphical user interfaces are generated. To validate the proposed framework, experimental studies were performed with computer and health professionals. The results showed that Template4EHR reduced the coding effort for a health application by 62%. A set of software metrics confirmed the compliance of Template4EHR with framework maintainability, flexibility and reusability characteristics. In addition, it optimized data schemas and graphical user interface development.

Keywords: Framework. Health Information Systems. Data Schemas. Polyglot Persistence. Electronic Health Record. Archetypes.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

Esta seção contextualiza e discute a importância do desenvolvimento de sistemas de informação em saúde baseado no conceito de arquétipos, motiva o desenvolvimento do framework descrito neste documento, lista os objetivos gerais e específicos almejados e descreve como esta tese está organizada.

1.1 CONTEXTUALIZAÇÃO

Ao longo dos últimos anos, a indústria de software, as instituições governamentais e a academia têm debatido o uso de padrões em saúde no desenvolvimento de soluções que melhorem a qualidade dos serviços prestados, aumentem a produtividade dos profissionais de saúde e popularizem o acesso às informações do registro eletrônico de saúde (RES) [1-5]. Essa iniciativa visa melhorar a flexibilidade e extensibilidade dos sistemas de informação em saúde (SIS), e permitir que os profissionais envolvidos nessa área contribuam efetivamente no desenvolvimento de SIS.

Conforme determinam as boas práticas de órgãos internacionais [6], os SIS devem armazenar o RES, preservando o histórico e a evolução dos dados clínicos, podendo o SIS ser reutilizado e compartilhado por outros domínios da área da saúde. Como qualquer outra categoria de software, os SIS enfrentam problemas ocasionados pelo alto custo de desenvolvimento e manutenção, pela falta de padrões que modelem o RES e pela rigidez de esquema de dados ocasionada pelo uso de um único paradigma de sistema de banco de dados para armazenar a heterogeneidade dos requisitos de dados do RES, por exemplo, um exame laboratorial com diversos níveis hierárquicos de dados, a evolução clínica de um paciente com exames de imagem ou a estrutura tabular de um exame de anatomia patológica.

Dentro do ciclo de vida de um SIS, a criação de esquemas de dados para o armazenamento do RES é um tema relevante e tem sido objeto de estudo de diversos trabalhos de pesquisa [7,8]. Segundo a norma ISO/TS 18308 [9], o esquema de dados do RES deve armazenar qualquer evento clínico relevante para os cuidados do paciente, incluindo o registro de descrições textuais, valores numéricos, valores lógicos e

expressões de data e hora. Além disso, o RES deve conter dados em tabelas de modo que as relações entre as linhas e colunas sejam preservadas. Por fim, o RES deve permitir o armazenamento de dados na forma de pares de atributos e em estruturas de dados hierarquizadas de diversos níveis. No entanto, como relatado em [10], pouca atenção tem sido dada na investigação de como representar a heterogeneidade dos dados do RES utilizando diferentes sistemas de bancos de dados.

Como característica natural, um sistema de software muda ao logo do tempo e, além disso, deve adaptar-se às novas exigências do seu domínio de aplicação, mesmo estando em execução. Os SIS lidam com um grande número de conceitos que mudam continuamente ou são especializados após um curto período de tempo. Consequentemente, os SIS construídos por meio de metodologias de desenvolvimento de software tradicionais são caros de manter. Geralmente, as mudanças requeridas em uma aplicação de saúde demandam esforço e dependência em uma equipe de programação. Por outro lado, percebe-se que os SIS não são projetados para permitir mudanças dinâmicas, ou seja, não se adaptam ao contexto do domínio de problema, nem permitem que usuários finais criem novas instâncias de uma aplicação para outros domínios, ou desenvolvam novas funcionalidades.

Nesse contexto, os padrões *Health Level Seven* (HL7) e openEHR representam importantes iniciativas que visam melhorar o desenvolvimento e a troca de dados entre aplicações de saúde [11]. O padrão HL7 fornece um conjunto de especificações que padronizam a troca e a transferência de dados entre SIS, enquanto a arquitetura dual da openEHR, objetiva melhorar a flexibilidade, manutenibilidade e extensibilidade dos SIS. A arquitetura dual realiza a separação das características genéricas que modelam a estrutura do RES, também conhecido como modelo de referência, das restrições e padrões associados aos dados clínicos, denominado de modelo de conhecimento [12,13].

A especificação da arquitetura dual da openEHR é realizada por meio de arquétipos e templates. Um arquétipo pode ser definido como uma expressão computacional representada por restrições de domínio, que modelam os atributos de dados e dão significado semântico ao RES, enquanto templates representam interfaces gráficas de interação do usuário criadas em tempo de execução a partir das especificações definidas nos arquétipos [14,15].

A criação de um framework que construa dinamicamente esquemas de dados e interfaces gráficas de usuário a partir da leitura de arquétipos, permitirá que SIS sejam desenvolvidos a partir de um padrão em saúde que uniformiza os atributos de dados, as terminologias e as restrições do RES. Além disso, o uso da persistência poliglota minimiza os problemas de representação da heterogeneidade dos dados do RES e a falta de flexibilidade para a alteração de esquemas de dados. O termo persistência poliglota adotado neste trabalho consiste na inserção, atualização, exclusão e consulta dos dados do RES em diferentes sistemas de bancos de dados (e.g., Relacional e NoSQL).

1.2 MOTIVAÇÃO

Embora a divulgação da arquitetura proposta pela openEHR venha crescendo consideravelmente nos últimos anos [16-49], nota-se no estado da arte a necessidade de se investigar a construção de esquemas de dados para armazenamento do RES utilizando arquétipos [50-51]. A aplicação de arquétipos no ciclo de vida inicial de um sistema de software ou na integração com SIS legados evidencia diversos tipos de dificuldades que necessitam ser exploradas [52], por exemplo, a falta de ferramentas que permitam a geração e a customização de interfaces gráficas de usuário com funcionalidades de persistência de dados para SIS, a construção de esquemas de dados utilizando os atributos de dados, as terminologias e as restrições de arquétipos e a avaliação da arquitetura de software de aplicações de saúde criadas a partir das especificações da openEHR. Além disso, recentes manifestos da comunidade científica ressaltam a importância da especificação de frameworks para apoiar a indústria de software no desenvolvimento de SIS utilizando arquétipos [53].

Trabalhos de pesquisa baseados nas especificações openEHR voltados para o setor de saúde incluem a modelagem do RES utilizando e especializando arquétipos existentes [54], a representação de arquétipos a partir de ontologias para permitir o processamento de informações semânticas [55], a avaliação do desempenho de SGDB no processamento de arquétipos [56] e um estudo sobre padrões de desenvolvimento para sistemas computacionais em saúde [57]. Além disso, arquétipos da openEHR já foram utilizados para criar protótipos de interfaces gráficas de usuário que gerenciam os cuidados clínicos de pacientes [58], representar os conceitos clínicos de sistemas legados [59] e modelar o

RES em um sistema de banco de dados proprietário [60]. Contudo, a criação de esquemas de dados em diferentes sistemas de bancos de dados e a geração de interfaces gráficas de usuário com recurso de persistência poliglota do RES são questões que necessitam ser investigadas.

Existem duas vantagens principais no uso de diferentes sistemas de bancos de dados para armazenar os dados do RES. Primeiro, esquemas de dados NoSQL são criados e estendidos em tempo de execução para representar os requisitos de dados que sofrem constantes mudanças (e.g., exames laboratoriais, prescrições médicas, evolução clínica do paciente) em um domínio da saúde. Segundo, os requisitos de dados (e.g., dados do paciente, dados do médico, dados financeiros) que sofrem poucas alterações e que necessitam manter a integridade referencial dos dados são armazenados em um banco de dados relacional.

A principal motivação para o desenvolvimento do trabalho aqui proposto consiste em especificar um framework para construir SIS dotado das características de manutenibilidade, reusabilidade e extensibilidade. A ideia é permitir que os profissionais de saúde construam, a partir da especificação do RES definida nos arquétipos, aplicações de saúde minimizando a dependência de uma equipe de desenvolvimento de software. Além disso, o framework deve contar com os seguintes recursos para o desenvolvimento de SIS: i) gerar dinamicamente interfaces gráficas de usuário a partir dos atributos de dados, das terminologias e das restrições dos arquétipos; ii) adicionar e remover componentes da interface gráfica gerada, mesmo ela estando em execução; iii) definir os componentes de preenchimento obrigatório da interface gráfica; e por fim, iv) persistir dinamicamente os dados inseridos na interface gráfica em diferentes sistemas de banco de dados.

1.3 PROBLEMA DE PESQUISA E HIPÓTESE

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

- *Questão 1: Como construir esquemas de dados independentes de tecnologias em aplicações de saúde utilizando os atributos de dados, as terminologias e as restrições de arquétipos?*

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

- *Hipótese 1: A utilização de um framework de aplicação baseado nas especificações openEHR reduz o esforço de programação, melhora a flexibilidade e a extensibilidade de aplicações de saúde, além de permitir que os usuários finais criem suas próprias funcionalidades utilizando arquétipos.*

Conhecidas as questões centrais de investigação, descreve-se na seção seguinte os objetivos gerais e específicos desta tese.

1.4 OBJETIVOS

Este trabalho propõe um framework para o desenvolvimento de aplicações de saúde utilizando arquétipos, chamado de Template4EHR, o qual tem como objetivo geral construir esquemas de dados para o armazenamento do RES nos paradigmas relacional e NoSQL, e gerar interfaces gráficas de usuário com recursos de persistência poliglota do RES.

Para fornecer uma visão conceitual de como construir esquemas de dados utilizando arquétipos, este trabalho objetiva ainda a especificação de um metamodelo em *unified modeling language* (UML) que represente os conceitos e relacionamentos da arquitetura dual da openEHR. Para garantir que os esquemas de dados gerados por Template4EHR sejam independentes de uma tecnologia de Sistema de Gerenciamento de Banco de Dados (SGBD), utilizou-se um padrão arquitetural que permite que as regras de negócios da aplicação acessem a camada de persistência de dados sem que a mesma tenha conhecimento do sistema de banco de dados escolhido. Um algoritmo exemplificará como os atributos de dados, as terminologias e as restrições são extraídas dos arquétipos e um conjunto de regras de mapeamento descreverão como as interfaces gráficas de usuário são geradas dinamicamente. Além disso, será ilustrada a criação de uma instância de aplicação de saúde utilizando arquétipos do repositório da openEHR e será investigado se o uso de Template4EHR reduz o esforço de programação durante o desenvolvimento de uma aplicação de saúde. Por fim, será avaliada a arquitetura de software por meio de métricas de software, a usabilidade das interfaces gráficas geradas e a acurácia dos esquemas de dados relacionais e NoSQL.

Para o desenvolvimento de Template4EHR, os objetivos específicos listados abaixo representam as atividades a sere desenvolvidas para a conclusão da pesquisa descrita neste documento. São elas:

- Criar um metamodelo em UML para prover uma visão conceitual de como construir esquemas de dados utilizando arquétipos.
- Especificar um algoritmo capaz de extrair de um arquétipo, os atributos de dados, as terminologias e as restrições, independente da quantidade de níveis hierárquicos em que os atributos estejam organizados.
- Descrever um conjunto de regras de mapeamento para geração das interfaces gráficas de usuário a partir dos atributos de dados, das terminologias e das restrições extraídas dos arquétipos.
- Projetar a arquitetura de Template4EHR.
- Garantir a independência de tecnologia dos esquemas de dados criados por Template4EHR, e a conformidade dos mesmos com o modelo de referência da openEHR.
- Desenvolver um serviço em nuvem para criação de esquemas de dados relacionais e NoSQL a partir dos elementos extraídos dos arquétipos.
- Desenvolver um serviço em nuvem para a geração de interfaces gráficas de usuário.
- Implementar uma REST API para persistir os dados manipulados por meio das interfaces gráficas em bancos de dados relacionais e NoSQL.
- Desenvolver uma aplicação móvel para reutilizar e executar as aplicações de saúde criadas por Template4EHR.
- Mensurar a redução de esforço proporcionada pelo framework durante o desenvolvimento de uma aplicação de saúde.
- Avaliar a arquitetura de software por meio de métricas de software.
- Realizar estudos experimentais para validar a acurácia dos esquemas de dados gerados e a usabilidade das interfaces gráficas de usuário.

1.5 METODOLOGIA DE PESQUISA

Para a elaboração desta tese, utilizou-se como metodologias de pesquisa a revisão bibliográfica e a pesquisa documental. A primeira identificou os principais trabalhos correlatos no estado da arte, enquanto que a segunda foi utilizada para extrair dos modelos computacionais fornecidos pela openEHR (i.e., *Demographic Information Model, Data Structures Information Model e openEHR Architecture*), os componentes necessários para a especificação do framework proposto neste trabalho.

1.6 PUBLICAÇÕES RELACIONADAS A TESE

Esta seção lista as publicações alcançadas e indica como elas se relacionam com os objetivos deste trabalho.

- Especificação do framework incluindo a arquitetura de software, a persistência poliglota do RES, os algoritmos para extração dos elementos de arquétipos, a geração de esquemas de dados relacionais e NoSQL e os estudos experimentais que avaliaram a acurácia dos esquemas de dados gerados e a usabilidade das interfaces gráficas de usuário.

[118] A. M. C. Araújo, V. C Times, M. U. Silva, " **A Tool for Generating Health Applications using Archetypes,** IEEE Software Journal, DOI 10.1109/MS.2018.110162508, pp. 1-7, 2018.

[124] A. M. C. Araújo, V. C. Times, M. U. Silva, "**Template4EHR: Building Dynamically GUIs for the Electronic Health Records Using Archetypes**". 16th IEEE International Conference on Computer and Information Technology, pp. 26-33, 2016. DOI: 10.1109 / CIT.2016.43.

[119] A. M. C. Araújo, V. C. Times, M. U. Silva, "**PolyEHR: A Framework for Polyglot Persistence of the Electronic Health Record**", The 17th International Conference on Internet Computing and Internet of Things. USA: CSREA Press, p. 71-77, 2016.

[120] A. M. C. de Araújo, Valéria C. Times, Marcus U. Silva, Sistema de persistência poliglota para registro de dados de saúde. 2016, Brasil. **Patente**: Privilégio de Inovação. Número do registro: **BR10201601858**, título: "**Sistema de persistência poliglota para registro de dados de saúde**" , Instituição de registro: INPI - Instituto Nacional da Propriedade Industrial. Depósito: 13/08/2016.

[122] A. M. C. Araújo, V. C. Times, M. U. Silva, "**Um Serviço para Geração de Esquemas de Dados Utilizando Arquétipos**", XV Congresso Brasileiro de Informática em Saúde, Goiânia, pp. 385-394, 2016.

[123] A. M. C. Araújo, V. C. Times, M. U. Silva, "**Dynamically Generating NoSQL Data Schemas using Archetypes**", 16th International Conference on WWW/Internet, pp. 31-38, 2017.

[125] D. S. Alves, V. C. Times, A. M. C. Araújo, M. U. Silva, A. S. C. Filho, M. A. Novaes, "**Usability Analysis of Archetyped Interfaces for the Electronic Health Record: a Comparative Study**". The Tenth International Conference on Advances in Computer-Human Interactions - ACHI 2017, pp. 169-175, 2017.

- Criação do metamodelo em UML que ilustra como construir esquemas de dados a partir das especificações da openEHR e o desenvolvimento de um serviço em nuvem que gera esquemas de dados e interfaces gráficas de usuário utilizando arquétipos.

[117] A. M. C. Araújo, V. C. Times, M. U. Silva, "**A Cloud Service for Graphical User Interfaces Generation and Electronic Health Record Storage**", Information Technology - New Generations. Advances in Intelligent Systems and Computing, vol 558. Springer, Cham. pp. 257-263, 2017. https://doi.org/10.1007/978-3-319-54978-1_36

[92] A. M. C. Araújo, V. C. Times, S. S. C. Branco, "**A Conceptual Data Model for Health Information Systems**", The 14th International Conference on Software Engineering Research and Practice. USA: CSREA Press, p. 236-242, 2016.

[91] A. M C. Araújo, V. C. Times, M. U. Silva, Carlos A.C. Bezerra, "**A CASE Tool for Modeling Healthcare Applications with Archetypes and Analysis Patterns**".The

Eleventh International Conference on Software Engineering Advances. Rome,Italy: IARIA, p. 206-211, 2016.

- Avaliação da arquitetura de software e das principais funcionalidades de Template4EHR.

[121] A. M. C. Araújo, V. C. Times, M. U. Silva, C. A. C. Bezerra, "**Software Architecture Modeling for Legacy Health Information System Using Polyglot Persistence and Archetypes**", The Twelfth International Conferece on Software Engineering Advances, ICSEA 2017, pp. 122-127, 2017.

- Desenvolvimento de uma aplicação móvel que reutiliza as interfaces gráficas de usuário e os esquemas de dados de Template4EHR e um estudo sobre padrões em saúde para criação de esquema de dados e intercâmbio do RES entre organizações de saúde.

[126] A. M. C. Araújo, V. C. Times, M. U. Silva, "**Building Health Mobile Applications Using Archetypes**". The Tenth International Conference on Advances in Computer-Human Interactions - ACHI 2017, pp. 47-52, 2017.

[12] C. A. C. Bezerra, A. M. C Araújo, "**Middleware For Heterogeneous Healthcare Data Exchange: A Survey**. The Tenth International Conference on Software Engineering Advances". Barcelona, Spain: pp. 409-414, 2015,.

1.7 ORGANIZAÇÃO DO TRABALHO

As seções restantes desta tese encontram-se estruturadas da seguinte forma.

2 Fundamentação Teórica

Esta seção descreve a fundamentação teórica e os conceitos básicos utilizados para a elaboração do trabalho detalhado neste documento.

3 Trabalhos Relacionados

Esta seção lista os principais trabalhos realizados pela comunidade cientifica na área de desenvolvimento de aplicações de saúde baseadas na arquitetura da openEHR, incluindo os trabalhos correlatos acerca do mapeamento de arquétipos em SIS, mecanismos de

persistência de arquétipos em banco de dados e frameworks voltados para o desenvolvimento de SIS

4 O Framework Template4EHR

Esta seção apresenta o framework desenvolvido para geração de esquemas de dados e interfaces gráficas de usuário utilizando arquétipos. Inicialmente, um metamodelo em UML ilustrará como criar esquemas de dados a partir dos conceitos da modelagem dual e arquétipos. Em seguida, a arquitetura de software especificada para o framework proposto e os principais componentes desenvolvidos são discutidos, e um conjunto de regras de mapeamento para extração dos elementos dos arquétipos e geração dos artefatos de software serão expostos. Um aplicativo móvel que reutiliza as interfaces gráficas de usuário e esquemas de dados gerados por Template4EHR será apresentado, e a criação de uma instância de aplicação de saúde utilizando arquétipos será ilustrada.

5 Avaliação da Arquitetura de Template4EHR

Esta seção apresenta as avaliações da arquitetura de Template4EHR. Inicialmente, investiga-se o quanto de esforço o framework pode reduzir no desenvolvimento de uma aplicação de saúde. Por meio de um conjunto de métricas de software, avalia-se a conformidade de Template4EHR com as características de manutenibilidade, reusabilidade e extensibilidade. Por fim, testa-se a geração de esquemas de dados em diferentes sistemas de bancos de dados, apresenta-se um experimento realizado com profissionais de computação e saúde para avaliar a geração e a customização de interfaces gráficas e compara-se o framework com outros trabalhos relacionados.

6 Conclusão

Esta seção apresenta as considerações finais sobre os principais tópicos abordados nesta tese, incluindo as contribuições alcançadas e as indicações de trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Esta seção contém a fundamentação teórica relativa ao trabalho descrito neste documento, iniciando com a definição dos principais conceitos sobre framework e arquétipos (Seção 2.1), e finalizando com os fundamentos básicos sobre a experimentação de produtos e ferramentas na engenharia de software (Seção 2.2.3).

2.1 CONCEITOS BÁSICOS

Esta seção está organizada como segue. A Seção 2.1.1 descreve os conceitos básicos de um framework, enquanto que a Seção 2.1.2 conceitua a modelagem dual proposta pela openEHR para a especificação do RES. A Seção 2.1.3 discute o uso da persistência poliglota para o armazenamento do RES. Por fim, a Seção 2.1.4 apresenta os conceitos de computação em nuvem utilizados no trabalho detalhado neste documento.

2.1.1 Definição de Framework

Ao longo dos últimos anos, pesquisas na área da engenharia de software têm mostrado que o uso de frameworks [61] é uma abordagem promissora para minimizar problemas relacionados a evolução de software, manutenibilidade e melhoria na qualidade do software produzido. O conceito de framework tem atraído a atenção da indústria de software e da academia principalmente em virtude da generalidade, reuso e redução do tempo de disponibilização de uma aplicação no mercado. Segundo Johnson (1998) [62], um framework pode ser definido como um projeto reutilizável de todo (ou parte de) um sistema de software, representado por um conjunto de classes abstratas que interagem entre si. Já para *Pree* (2005) [63], um framework modela o comportamento de um conjunto de aplicações.

Metodologias de desenvolvimento de sistemas baseadas em frameworks [64,65] preconizam o desenvolvimento de uma arquitetura comum para todas as instâncias de software dentro de um dado domínio. Comumente, dois tipos de frameworks são descritos na literatura. Framework conceitual, que corresponde a uma coleção de classes abstratas dedicadas a modelar um domínio de problema por meio de uma arquitetura flexível e extensível [64]; e framework de aplicação, que compreende um conjunto de classes

implementadas em uma linguagem de programação, usadas para auxiliar o desenvolvimento de software [65]. Dentre as principais características almejadas no desenvolvimento de um framework, destacam-se:

- **Reuso**: O propósito principal de um framework é o de ser reusável. Para isso, deve ser bem documentado e fácil de usar.
- **Flexibilidade**: Refere-se à capacidade de alterar funcionalidades existentes, prevendo impactos no funcionamento do framework.
- **Extensibilidade**: Refere-se à capacidade de permitir extensões de funcionalidades genéricas implementadas no framework.

A Figura 1 exemplifica como ocorre o desenvolvimento de uma instância de aplicação de software utilizando um framework. Nela é possível observar que as classes representadas dentro do retângulo simbolizam os componentes reutilizáveis do framework, enquanto as classes que estão fora do retângulo representam o código desenvolvido por um programador para atender as necessidades de uma aplicação específica.

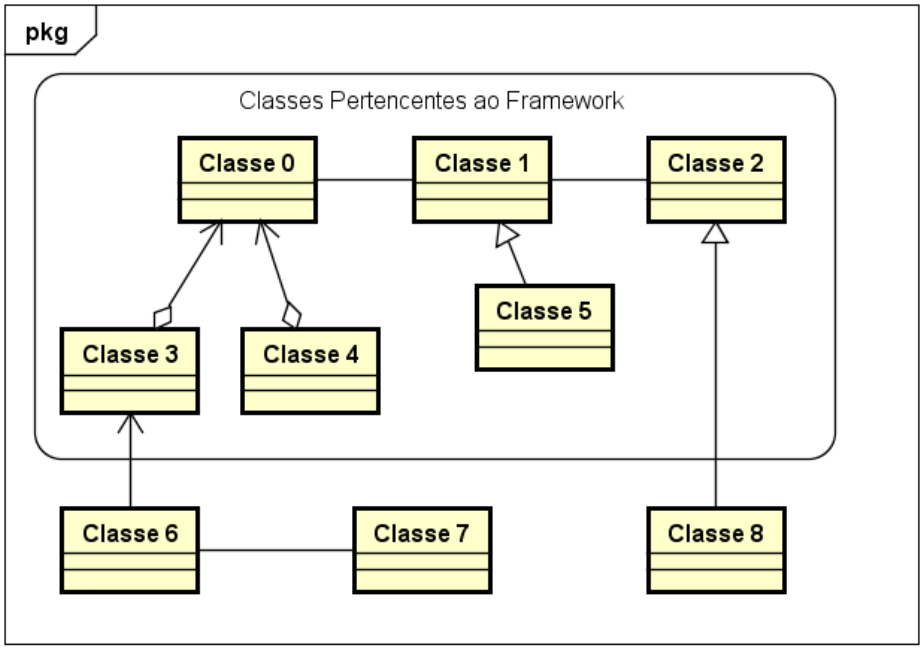


Figura 1 - Exemplo de Classes de um Framework

Para se construir um framework capaz de se adaptar a um conjunto de aplicações diferentes, é fundamental que se realize a modelagem de um conjunto significativo de aplicações de domínio. Este conjunto pode se referir a aplicações previamente desenvolvidas, ou aplicações que se deseja construir a partir do framework. Em termos práticos, dotar um framework de generalidade, flexibilidade e extensibilidade requer uma cuidadosa identificação de pontos de flexibilização (i.e., *hot spots*). O conceito de pontos de flexibilização corresponde a partes das estruturas de classes do framework que devem ser mantidas flexíveis, para possibilitar a adaptação da aplicação desenvolvida a diferentes domínios [66].

Como uma aplicação genérica, um framework deve encapsular tanto as características comuns, quanto os pontos que permitem que o framework seja customizado para aplicações com requisitos específicos [66]. Quando uma aplicação é criada a partir de um framework, liga-se um ponto de flexibilização com uma funcionalidade específica implementada por um programador.

O framework proposto neste trabalho é caracterizado como sendo de aplicação e contém as características de reuso, flexibilidade e extensibilidade. O reuso é aplicado quando uma determinada interface gráfica de usuário é utilizada em diferentes instâncias de aplicações geradas por Template4EHR. A flexibilidade implementada permite escolher os atributos de dados, as terminologias e as restrições que serão utilizadas para gerar as interfaces gráficas, enquanto a extensibilidade das classes permite a inserção de novos métodos sem afetar o comportamento do framework.

A análise comparativa entre as principais contribuições desta tese e os frameworks correlatos identificados no estado da arte, é relatada na Seção 3.3 deste documento.

2.1.2 Modelagem Dual e Arquétipos

A fundação openEHR é responsável por manter e validar todas as especificações da modelagem dual para SIS. As especificações para a criação do RES publicadas pela openEHR são definidas usando a notação da UML e estão estruturadas da seguinte forma: modelo de referência (MR), modelo de serviço (MS) e modelo de arquétipos (MA) [67]. O modelo de referência define a informação presente no RES, enquanto o modelo de serviço contém a visão computacional utilizada para a implementação do RES em

plataformas de software. O modelo de arquétipos (também conhecido como modelo de conhecimento) representa a ligação entre o modelo de referência, as terminologias e os padrões em saúde. Todos os modelos especificados pela openEHR incluem classes, funções, restrições, pré e pós-condições [68].

A Figura 2 ilustra o relacionamento entre MR, MS e MA e a plataforma computacional da openEHR. Pode-se observar que cada modelo se relaciona com um ou mais conceitos da plataforma computacional. Este relacionamento indica que, para cada conceito da plataforma computacional, o respectivo modelo serve de guia de implementação. As definições de cada modelo estão especificadas por meio de diagramas UML e visam auxiliar o projetista no desenvolvimento de SIS utilizando as especificações da openEHR.

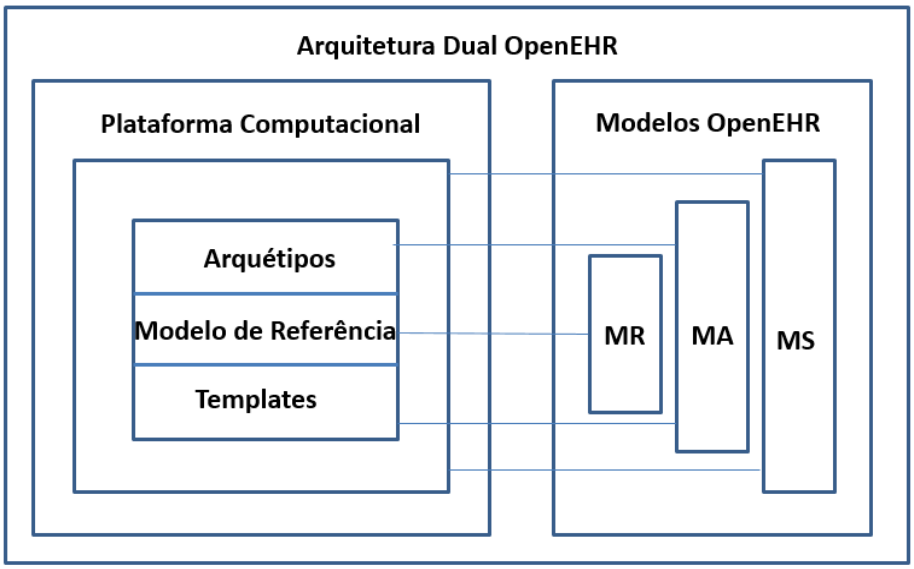


Figura 2 - Plataforma computacional da openEHR (adaptada de [27]).

A modelagem dual da fundação openEHR objetiva a criação de um RES único, preservando o histórico e a evolução dos dados clínicos do paciente, podendo o RES ser compartilhado e reutilizado por outros domínios da saúde [69]. Conceitua-se como modelagem dual, a separação entre a informação que compõe o RES e o conhecimento associado aos conceitos e às terminologias da área da saúde. O primeiro nível da modelagem dual contempla os componentes de linguagem de programação, linguagem de troca de informações (XML) e os outros componentes relacionados ao

desenvolvimento de softwares, tais como: tecnologia de banco de dados e artefatos de software (e.g., diagramas da UML, modelo conceitual e lógico de banco de dados).

A Figura 3 ilustra como ocorre a separação entre os níveis de informação e conhecimento da modelagem dual. Para realizar a separação entre os conceitos de informação e conhecimento, a openEHR utiliza as ontologias de informação e conhecimento. É importante observar que a ontologia da informação classifica o modelo de referência e o modelo de serviço como modelos de informação estável, e que são implementados em plataformas de software (i.e., primeiro nível). Pode-se também notar que o nível de conhecimento, representado pelas terminologias, está separado do modelo de referência e é representado por arquétipos e templates.

Cada arquétipo possui um conjunto de regras definidas no modelo de referência, sendo instanciado em tempo de execução por meio dos templates, que controlam todo o processamento das informações. Templates são interfaces gráficas (e.g., formulário e relatório) que permitem que os arquétipos sejam agrupados e disponibilizados para os usuários finais.

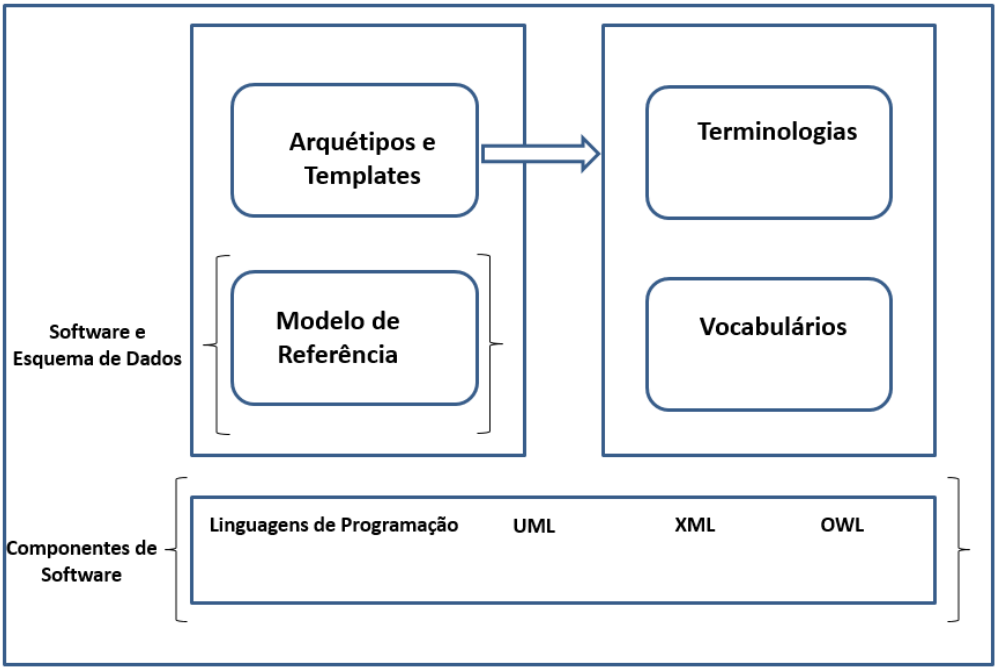


Figura 3 - Modelagem Dual da OpenEHR (adaptada de [28]).

Em termos práticos, a utilização da modelagem dual reduz a dependência de desenvolvedores de software e da tecnologia de desenvolvimento escolhida, tornando a

manutenção e extensão dos aplicativos mais ágil e flexível. Isso ocorre em decorrência da implementação ocorrer apenas no primeiro nível, uma vez que todas as alterações e a inclusão de novas funcionalidades ocorrem no segundo nível, com a utilização de arquétipos e templates, minimizando a necessidade de alterações do modelo de referência, também chamado de modelo computacional.

Em um arquétipo, a especificação dos atributos é realizada por meio de construtores de entrada de dados chamados de estruturas de dados genéricas. As estruturas de dados genéricas permitem representar a heterogeneidade dos dados do RES por meio dos tipos, *ITEM_SINGLE, ITEM_LIST, ITEM_TREE e ITEM_TABLE*.

O *ITEM_SINGLE* modela um único atributo de dado. Por exemplo, pode ser usado para especificar o peso, a altura e a idade do paciente. O *ITEM_LIST* agrupa um conjunto de atributos e é representado na forma de uma lista. O endereço de um paciente com os itens, rua, número e CEP, exemplifica o uso de um *ITEM_LIST*. O *ITEM_TREE* especifica uma estrutura hierárquica de dados que é logicamente representada na forma de uma árvore. Ele pode ser usado, por exemplo, para modelar a avaliação física e neurológica de um paciente. Por fim, o ITEM_TABLE modela os elementos de dados por meio de linhas e colunas, onde as linhas representam a definição do elemento, e as colunas o valor da informação. O resultado de um exame de análises clínicas contendo os campos dispostos em linhas, e os valores de referências em colunas, pode ser indicado como exemplo do ITEM_TABLE.

A Figura 4 mostra o exemplo de um arquétipo (i.e., *family history)* que modela os antecedentes familiares de um paciente utilizando um ITEM_TREE. Neste arquétipo, os atributos de dados estão organizados em uma estrutura hierárquica de vários níveis. Além disso, é possível observar na Figura 4, que o atributo de dado *Problem/diagnosis* possui uma terminologia em saúde (i.e., *ICD10CM)* que padroniza o diagnóstico do paciente e uma restrição (i.e., *ocurrences*) que indica que pelo menos um diagnóstico deve ser informado.

Cada atributo de uma estrutura de dados é caracterizado por um tipo de dado e pode ter ainda um conjunto de restrições de domínio e terminologias associadas. As terminologias dão significado semântico aos dados clínicos e podem ser representadas por meio de uma padronização de termos em saúde ou uma informação textual definida por um especialista do domínio.

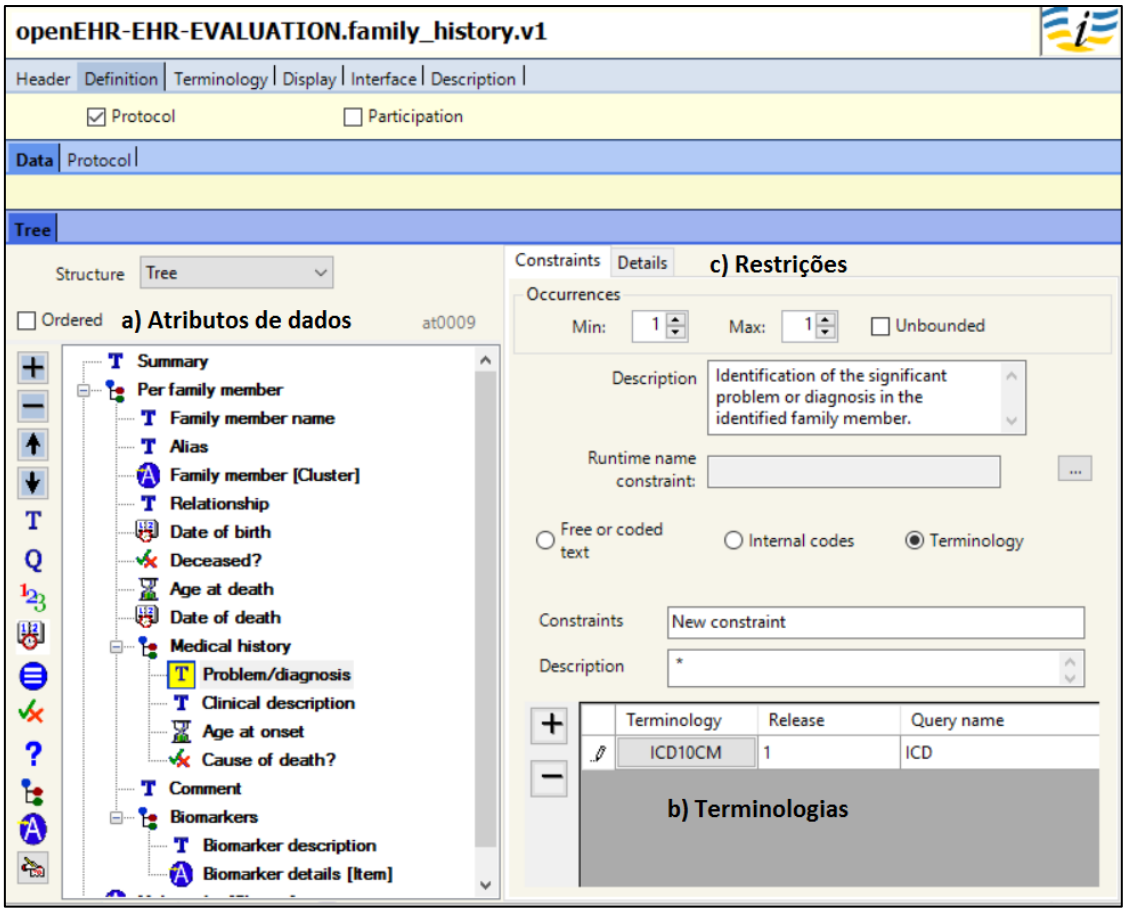


Figura 4 - Arquétipo Family History

A especificação do RES por meio das estruturas de dados genéricas representa uma alternativa para se alcançar a interoperabilidade entre transmissores e receptores de informação. Como as estruturas de dados genéricas atuam no primeiro nível do modelo, que é implementado nos sistemas computacionais, é possível que os desenvolvedores de software criem ferramentas ou serviços para ler e interpretar tais estruturas de dados genéricas (e.g., XML e *webservices*), permitindo que a interoperabilidade seja garantida entre as partes envolvidas na troca de informações.

Arquétipos podem ser descritos e especificados por meio de uma linguagem formal denominada *Archetype Definition Language* (ADL) [70]. Segundo Beale e Heard (2007) [70], a linguagem ADL descreve restrições baseadas em modelos de conteúdo de domínio. A linguagem ADL possui três elementos básicos na sua sintaxe: (1) dADL (*data ADL*) usada para definição de dados, (2) cADL (*constraint ADL*) possibilita a definição de restrições e (3) FOPL (*First-Order Predicate Logic*) que se destina à construção de expressões lógicas baseadas em predicados de primeira ordem.

A dADL define instâncias de dados baseados em um modelo de informação e sua estrutura é legível tanto para um leitor humano, quanto por uma estrutura automatizada de recuperação de arquétipos. As seções de um arquétipo que utilizam a sintaxe dADL são: idioma, tradução, descrição, terminologias, definição de restrições, termos obrigatórios e restrições obrigatórias. Um documento dADL pode conter um ou mais referências de arquétipos e todos os seus identificadores são oriundos de um modelo de informação.

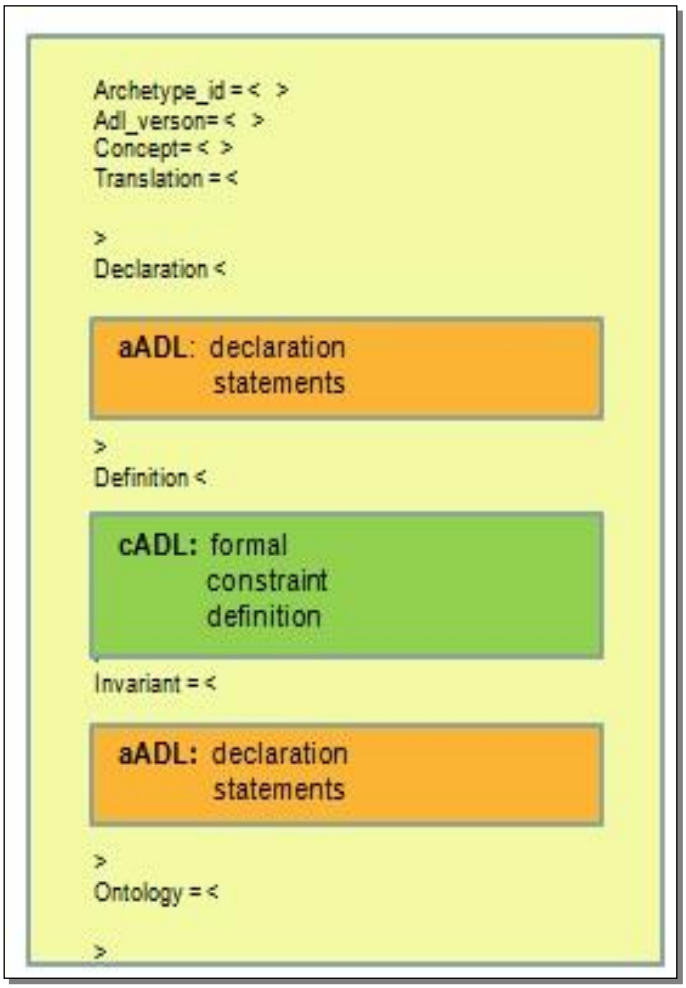


Figura 5 - Estrutura de um Arquétipo em Linguagem ADL (adaptada de [70]).

cADL é uma sintaxe que aplica restrições sobre dados definidos por modelos de informação orientados a objetos. Ela pode ser utilizada em tempo de desenvolvimento, por programadores e ferramentas e também em tempo de execução por algoritmos computacionais para validação de instâncias de dados. Dentre suas palavras reservadas destacam-se: *conjunto* - que define uma relação de pertinência; *ocorrência* - que indica o número de instâncias possíveis de um atributo; *existência* - que indica a opcionalidade de

um atributo; *cardinalidade* - que define se o atributo possui restrições e as palavras *ordenado, não ordenado* e *unicidade* que definem o tipo de agrupamento lógico usado no arquétipo.

A FOPL é uma linguagem de lógica de predicados de primeira ordem que vem inserida nas seções: definição e invariante. Nesta parte da estrutura ADL, utiliza-se principalmente os operadores de comparação e de igualdade.

Em síntese, a organização de um arquétipo descrito em ADL apresenta as seguintes seções básicas [70]:

Cabeçalho – Contém identificação do arquétipo e propriedades que definem a linguagem original, além de traduções, dados de autoria, informações que descrevem o arquétipo e que podem ser utilizadas para recuperação do arquétipo mantido em um repositório.

Definição - Especifica restrições para os atributos de dados no arquétipo, sendo as restrições descritas em cADL.

Invariante – Define expressões lógicas de primeira ordem para validar restrições que contêm fórmulas matemáticas ou lógicas.

Ontologia – Representa a ligação entre os atributos de dados e as terminologias.

A Figura 5 ilustra a estrutura de um arquétipo definido em ADL, cujo cabeçalho inclui a definição dos metadados dos arquétipos e a parte central contém a definição das restrições dos arquétipos. Além da especificação no formato *ADL*, os arquétipos podem ser especificados no padrão *eXtensible Markup Language* (XML), o que facilita a integração e a utilização dos arquétipos pela maioria das plataformas de desenvolvimento e armazenamento de dados.

2.1.3 Persistência Poliglota

Persistência poliglota consiste no uso de diferentes tecnologias de armazenamento de dados para lidar com diferentes necessidades de armazenamento [71]. A ideia central no uso de persistência poliglota consiste em armazenar os requisitos de dados estruturados em um banco de dados relacional, enquanto os dados de natureza semiestruturada e/ou não estruturada são mantidos em banco de dados NoSQL.

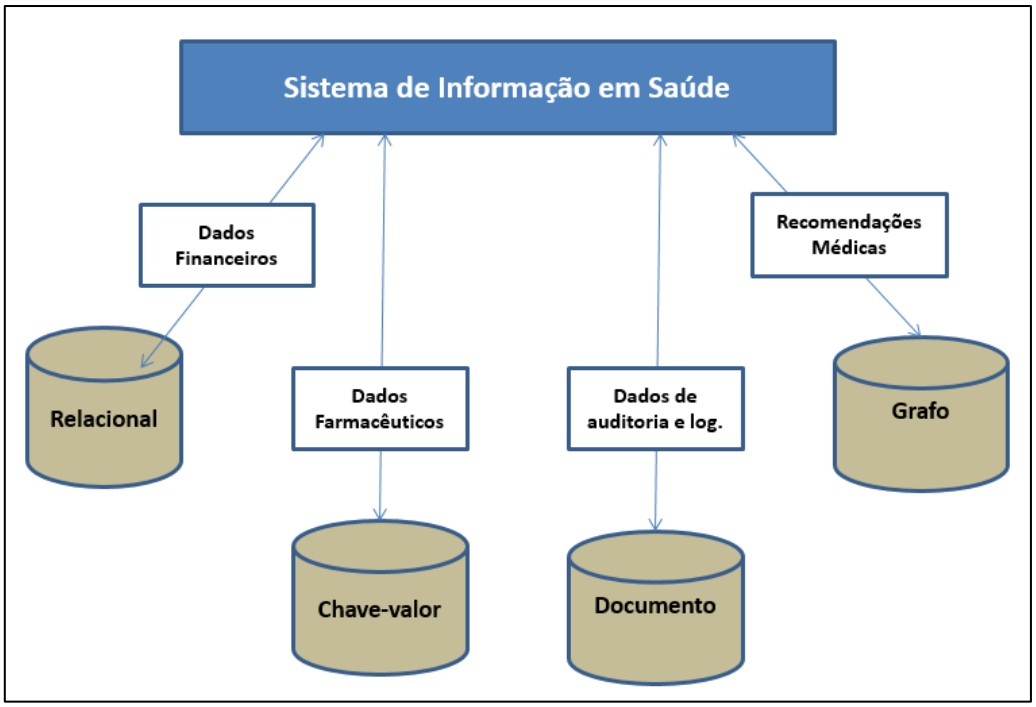


Figura 6 - Armazenamento em Diferentes Sistemas de Banco de Dados

A Figura 6 mostra como os diferentes tipos requisitos de dados de um domínio da saúde podem ser armazenados utilizando o conceito de persistência poliglota. Por exemplo, os dados financeiros de uma organização em saúde podem ser armazenados em um SGBD relacional, enquanto as recomendações médicas de um paciente podem ser armazenadas em um sistema de banco de dados orientado a grafo.

Como mostra a Figura 6, a abordagem NoSQL oferece diferentes sistemas de bancos de dados. Dentre os mais comuns, destacam-se: chave-valor, documento, grafo e família de colunas. A definição de cada sistema de banco de dados NoSQL é dada a seguir.

- **Chave-valor**: Armazena as informações na forma de uma tabela com duas colunas: chave e valor. O campo chave representa a descrição da informação, enquanto que o valor representa a instância do campo. A vantagem desta abordagem é que se novos requisitos de dados da aplicação surgirem, não há necessidade de se alterar o esquema de dados com a criação de novas colunas, haja vista que estes requisitos são registrados como uma linha nos campos, chave e valor.
- **Documento**: Gerencia o armazenamento de dados por meio de coleções de documentos. Um documento é composto por um conjunto de campos que podem

ser representados na forma de um único elemento, uma lista ou conjunto de campos aninhados. Uma das principais características desse modelo é que ele não depende de um esquema de dados previamente definido, ou seja, é possível criar e alterar a estrutura do documento em tempo de execução.

- **Grafo**: Utiliza nós, relacionamentos e propriedades para armazenar os dados de um domínio de problema. Os nós representam os vértices do grafo, os relacionamentos modelam as arestas, enquanto as propriedades representam os atributos.
- **Família de Colunas**: Gerencia o armazenamento de dados por meio dos conceitos de *keyspace*, família de colunas e colunas. O *Keyspace* representa conjuntos de famílias de colunas, enquanto a família de colunas incorpora e organiza um conjunto de colunas, semelhante a uma tabela relacional. Por fim, as colunas armazenam os dados por meio dos campos: nome, *timestamp* e valor.

Sistemas de bancos de dados NoSQL possuem características que os diferenciam dos demais sistemas de armazenamento de dados (e.g., relacional), como por exemplo, estrutura de dados para representação de informações não estruturadas e flexibilidade na criação e alterações de esquema de dados. Por esses motivos, a combinação de diferentes sistemas de bancos de bancos (i.e., NoSQL e relacional) vem sendo recomendada em diversos domínios de problemas, inclusive no setor de saúde [72].

2.1.4 Computação em Nuvem

Desde que as organizações em saúde minimizaram o uso de papel para registrar o histórico clínico dos pacientes, o gerenciamento dos dados eletrônicos tornou-se um importante desafio no ciclo de vida de um SIS. Um simples atendimento ambulatorial criado para registrar anotações clínicas, tratamentos e prescrições médicas de um único paciente, pode gerar milhares de tuplas em um banco de dados. Estima-se que até o ano de 2020, os dados processados pelos SIS em todo o mundo alcancem a impressionante marca de 2.500 *Pbytes* [73]. O processamento de grandes volumes de dados aliado aos desafios de gerenciar uma infraestrutura local para prover aplicações e serviços tem feito com que as organizações de saúde introduzam a computação em nuvem como alternativa para melhorar o gerenciamento dos ativos de TI.

Entende-se como computação em nuvem, todo ambiente computacional formado por diversos servidores, físicos ou virtuais, com capacidade de processamento e gerenciamento de aplicações, plataformas e serviços disponibilizados na internet [74,75]. Como principal característica, a computação em nuvem possibilita acesso de modo conveniente e sob demanda a um conjunto de recursos computacionais configuráveis que podem ser rapidamente adquiridos e liberados com o mínimo esforço de configuração ou interação com o provedor de serviços [76]. 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 [77].

Outra característica importante é que a infraestrutura necessária para o processamento, a conectividade e o armazenamento dos dados, ficam hospedados em provedores (e.g., *Microsoft Azure e Amazon*) especializados nesse tipo de serviço. Por outro lado, em um ambiente tradicional da área da saúde, para se construir ou gerir um SIS, os profissionais de 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.

Conforme ilustra a Figura 7, 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). A definição de cada serviço é dada a seguir.

- **Software como Serviço:** O software é oferecido como serviço ou sob demanda. 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.

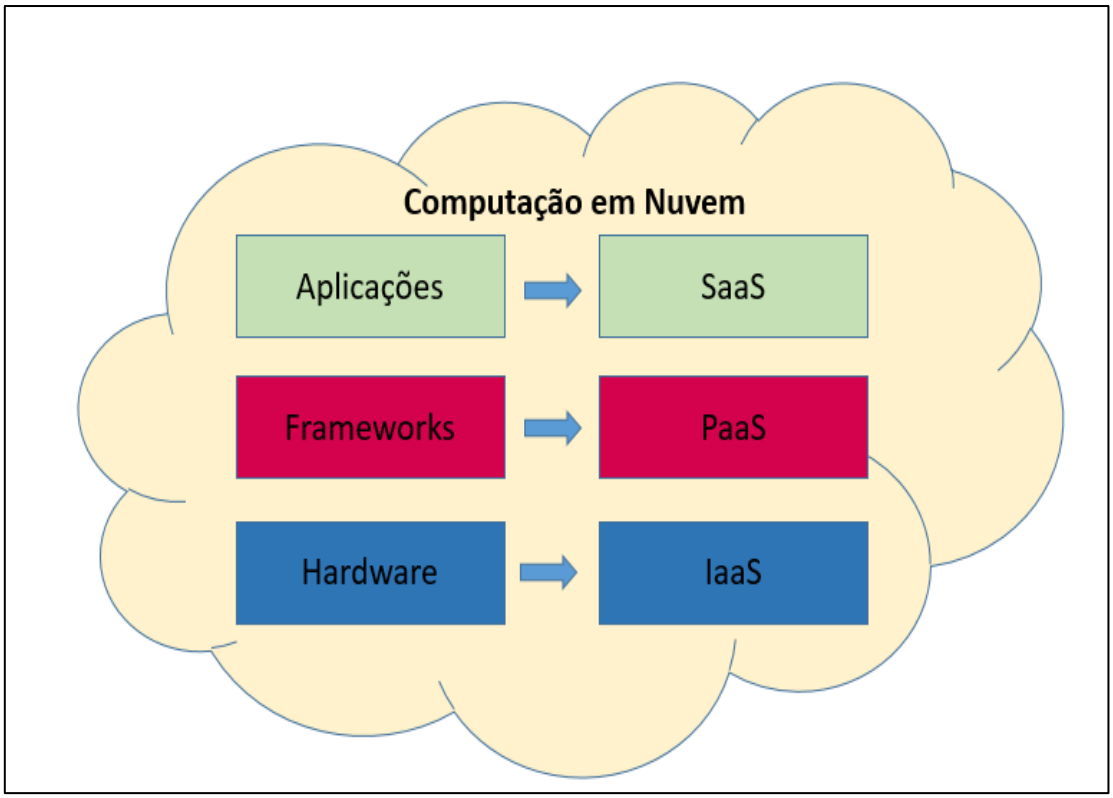


Figura 7 - Serviços da Computação em Nuvem

Os trabalhos correlatos identificados no estado da arte que utilizam computação em nuvem em SIS e a especificação do serviço em nuvem destinado a geração de esquemas de dados e interfaces gráficas de usuário são descritos nas Seções, 3.4 e 4.4, respectivamente, deste documento.

2.2 METODOLOGIA DE AVALIAÇÃO DE FRAMEWORKS

Esta seção descreve as metodologias utilizadas para avaliar o framework descrito neste documento. A Seção 2.2.1 apresenta um modelo de avaliação que investiga a redução de esforço de programação proporcionada por um framework, enquanto que a Seção 2.2.2 descreve o uso de métricas de software como forma de avaliar a arquitetura de um sistema. Por fim, a Seção 2.2.3 aborda a utilização da experimentação na avaliação de ferramentas de software.

2.2.1 Avaliando o Benefício de Uso de Um Framework

´Um dos principais desafios na avaliação de um framework de aplicação consiste em mensurar o quão rápido um projeto de software pode ser construído com baixo custo e, quanto esforço de desenvolvimento pode ser reduzido na fase de codificação[78,79].

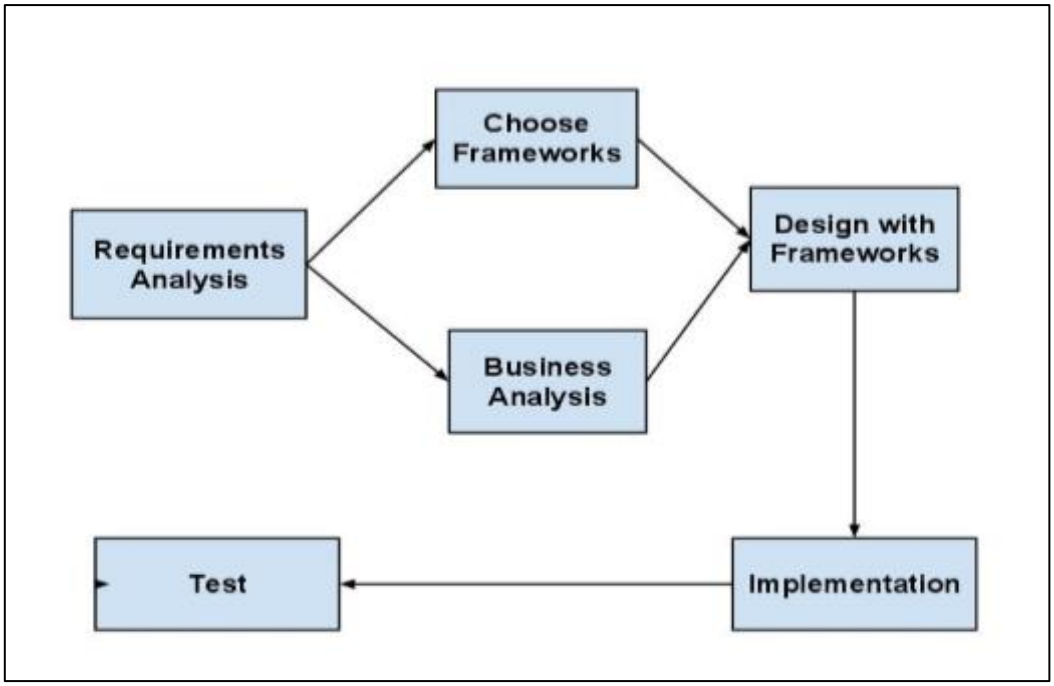


Figura 8 - Uso de Frameworks no Ciclo de Desenvolvimento de Software [80]

Nesse sentido, a partir do entendimento de como os frameworks atuam no ciclo de vida de desenvolvimento de software (Figura 8), Lee (2012) [80] especificou um modelo de avaliação baseado em cinco categorias de custos de um projeto. São elas: i) custo do sistema; ii) custo do projeto; iii) custo de aprendizagem; iv) custo de implementação, e v) custo de testes. A definição de cada categoria é dada a seguir.

Custo do Sistema: Esta métrica avalia o custo do licenciamento dos softwares e o esforço gasto na configuração e instalação das tecnologias a serem utilizadas antes do desenvolvimento. Por exemplo, para que uma aplicação seja executada em um navegador na internet, um servidor web precisa ser instalado e um SGDB configurado para armazenar os dados. Nesse caso, cada tecnologia possui características e configurações próprias que demandam tempo, esforço e valor.

Custo do Projeto: Após a escolha de um framework, projeta-se a arquitetura de software com base nos requisitos de dados do domínio da aplicação. Nesse sentido, é preciso investigar o quão alinhado está a arquitetura da aplicação com os recursos fornecidos pelo framework escolhido. A arquitetura do software determina como os requisitos funcionais, bem como os atributos de qualidade do projeto, como compatibilidade, extensibilidade, modularidade, reutilização, robustez e usabilidade, são atendidas. Sendo assim, esta métrica avalia as classes, interfaces, padrões de projeto e esquemas de dados.

Custo de aprendizagem: O custo de aprendizagem investiga o tempo necessário para entender a sintaxe da linguagem de programação e os recursos oferecidos pelo framework.

Custo de Implementação: Esta métrica avalia o quanto de esforço o framework reduz na implementação dos requisitos funcionais e não funcionais de uma aplicação. Por exemplo, o framework pode dispor de templates e controles que criam dinamicamente funcionalidades de autenticação e recuperação de senha de usuários.

Custo de Testes: Esta métrica mensura se durante o ciclo de vida de software os desenvolvedores possuem ferramentas para testar, analisar e depurar o código fonte no framework.

O Quadro 1 sumariza as categorias e as métricas que devem ser consideradas durante o processo de avaliação de um framework.

Quadro 1 - Categoria de Custos para Avaliação de Frameworks

Custos				
Sistema	**Projeto**	**Aprendizagem**	**Implementação**	**Testes**
Instalação	Classes	Usabilidade	Requisitos Funcionais	Testabilidade
Ferramentas	Interfaces	Suportabilidade	Requisitos não funcionais	
	Banco de dados	Manejabilidade		
		Compreensibilidade		
		Adoção		

Para calcular os custos de um projeto de software e os benefícios de se desenvolver uma aplicação com o auxílio de um framework, foi especificado um questionário baseado nas cinco categorias de custos de um projeto de software (Quadro 2) e que deve ser respondido pelos participantes da avaliação.

Quadro 2 - Questionário de Avaliação de Um Framework de Aplicação

Custo do Sistema (15)
Quão difícil é a configuração do ambiente de desenvolvimento? (0 – 5)
Quantos programas de software você precisa instalar? (0 – 5)
Quanto custa os programas de software? (0 – 5)
Custos do Projeto (20)
Você precisa criar classes? (0 – 5)
Você precisa projetar interfaces? (0 – 5)
Você precisa aplicar um padrão de projeto? (0 – 5)
Você precisa projetar um banco de dados? (0 – 5)
Custo de Aprendizagem (20)
Usabilidade: quão fácil é a usabilidade do framework? (0 – 5)
Adoção: quão popular é o framework? (0 – 5)
Suportabilidade: o quão bem organizado é o recurso de ajuda? (0 – 5)
Gerenciamento: com que facilidade pode ser adicionado novos recursos ao framework? (0 – 5)
Custo de Implementação (25)
Segurança: quão estável é o framework? (0 – 5)
Extensibilidade: o framework pode trocar dados e se comunicar com sistemas implementados por partes externas? (0 – 5)
Usabilidade: quão responsivo é a aplicação gerada pelo framework? (0 – 5)
Acessibilidade: é possível acessar a aplicação desenvolvida por meio de diversos dispositivos? (0 – 5)
Requisitos não funcionais
Quantos requisitos funcionais o framework não cobre? (0 – 5)
Custo de Testes (15)
Testabilidade: o framework possui recursos de testes? (0 – 5)
Testabilidade: o framework possui ferramentas de depuração? (0 – 5)
Testabilidade: o framework possui recursos para avaliação de métricas? (0 – 5)

O objetivo da avaliação é calcular o custo total máximo do desenvolvimento de uma aplicação e o benefício de se utilizar um framework. Para cada pergunta do questionário, a resposta deve conter um valor de 0 a 5. O custo total máximo é calculado multiplicando-

se o total de perguntas por 5, isto é, pelo valor máximo que uma pergunta pode receber. O custo do framework é calculado a partir do somatório do valor atribuído a cada resposta (i.e., 0 a 5), enquanto o benefício é calculado subtraindo-se o custo total máximo, do custo do framework que está sendo avaliado. Por fim, para mensurar a taxa de redução de custos, divide-se o benefício do framework pelo custo total. Por exemplo, imagine que o custo total máximo é 95 (i.e., multiplicação da quantidade de perguntas do questionário por 5), e 35 é o custo do framework (i.e., somatório de todas as respostas dadas). Assim, o benefício do framework é de 60, e a taxa de redução de custos é de 66% (i.e., 60/95= 0,63). Nesse caso, 63% de esforços e custos podem ser reduzidos com o uso do framework avaliado.

Como descrito em [80], recomenda-se que a avaliação do framework em questão ocorra em um domínio real de aplicação, e que de dois a cinco participantes do experimento tenham experiência superior a dois anos em desenvolvimento de software.

2.2.2 Medição de Software

A garantida da qualidade de software, é um tema atual, relevante e que tem atraído a atenção da indústria de software e da academia na busca por soluções que melhorem a qualidade dos atributos de software, tais como, a funcionalidade, a confiabilidade, a manutenibilidade e a eficiência [81]. Como descrito em [82], existem diversas maneiras de medir a qualidade dos atributos de um sistema de software. Dentre elas, a medição de software ajuda a entender o comportamento e o funcionamento de produtos de software, além de ajudar a identificar padrões, metas e critérios de aceitação de um produto.

Métricas de software servem para apresentar medidas, preferencialmente quantitativas e que reflitam as características específicas dos processos e do software em análise. Do ponto de vista de medição, as métricas de software podem ser classificadas como diretas e indiretas [83]. Define-se como medidas diretas o custo e o esforço aplicados no desenvolvimento e na manutenção de um software, enquanto medidas indiretas avaliam aspectos que não são medidos quantitativamente, por exemplo, a complexidade, a eficiência e a confiabilidade.

Três conceitos básicos são necessários para se realizar a medição de um software. São eles [83]:

- **Medida:** um número que representa a quantidade, dimensão, capacidade ou tamanho de um sistema de software.
- **Medição:** ato de calcular uma medida.
- **Indicador:** é uma métrica ou a combinação delas, que fornece compreensão do processo ou do produto de software.

Nesta tese, utilizou-se o conceito de métricas de software para se investigar a conformidade de Template4EHR com as com as características de manutenibilidade, reusabilidade e extensibilidade. Para isso, utilizou-se o benchmark da Microsoft [84] como ferramenta de medição das seguintes métricas (medidas):

- **Índice de manutenibilidade** – refere-se à facilidade, precisão e economia na execução de ações de manutenção no software avaliado.
- **Complexidade ciclomática** - mede a quantidade de caminhos lógicos independentes a partir do código fonte de um programa.
- **Profundidade de herança** - indica a posição de uma classe na árvore de herança de um software.
- **Acoplamento das classes** - define o grau de dependência entre os componentes de uma arquitetura de software.
- **Linhas de código implementadas** – mede a quantidade linhas de código implementadas.

A partir dos indicadores calculados para as métricas descritas acima, pode-se avaliar e conhecer alguns dos atributos de qualidade de Template4EHR. A Seção 5.2 deste documento apresenta os resultados obtidos na avaliação da arquitetura de software realizada por meio do uso de métricas de software.

2.2.3 Experimentação na Engenharia de Software

A utilização de estudos empíricos e a experimentação na engenharia de software vem sendo amplamente debatida pela comunidade científica [85]. Estudos empíricos são importantes para a avaliação de processos e atividades realizada por seres humanos, pois fornecem uma maneira sistemática, disciplinada, quantificável e controlada de avaliar produtos ou ferramentas de software [86].

A experimentação na engenharia de software é uma investigação empírica que manipula um fator ou variável do cenário estudado [87]. Com base na aleatorização, diferentes tratamentos são aplicados a diferentes sujeitos, mantendo outras variáveis constantes e medindo os efeitos sobre os resultados produzidos. As avaliações são feitas em um ambiente controlado (e.g., laboratório) onde os participantes realizam diferentes tratamentos aleatoriamente. Nesse caso, tem-se o objetivo de manipular uma ou mais variáveis e controlar todas as outras variáveis em níveis fixos. O efeito da manipulação é medido e, com base nisso, uma análise estatística pode ser realizada [87].

O delineamento experimental é o plano utilizado para realizar o experimento. Esse plano determina a maneira como os diferentes tratamentos deverão ser distribuídos nos tratamentos experimentais e como serão analisados os dados a serem obtidos. Como exemplo de delineamento, pode-se citar o quadrado latino que é comumente utilizado em experimentos na engenharia de software.

Os dois meios mais comuns de coleta de dados são questionários e entrevistas [88]. Os questionários podem ser fornecidos em formato de papel ou por meio de um formulário eletrônico. O questionário deve ser entregue aos participantes juntamente com as instruções sobre como preenchê-lo. Para confeccionar um questionário, deve-se adotar uma linguagem clara, objetiva e com escalas de respostas que representem o objeto de estudo do experimento [89,90].

Na seção 5 desta tese, descreve-se o experimento controlado realizado para comparar Template4EHR com uma ferramenta correlata de larga utilização no mercado internacional. Além disso, apresenta-se os estudos que avaliaram as funcionalidades de geração dinâmica de esquemas de dados e interfaces gráficas de usuário.

2.3 CONSIDERAÇÕES FINAIS

Esta seção apresentou os conceitos básicos utilizados para o desenvolvimento da tese descrita neste documento. Para tanto, definiu-se os fundamentos sobre frameworks, arquitetura dual da openEHR, persistência poliglota e computação em nuvem. Além disso, discutiu-se o uso de métricas de software e estudos empíricos na avaliação de produtos e ferramentas de software.

Na seção 3, apresenta-se os trabalhos relacionados identificados no estado da arte na área de desenvolvimento de aplicações de saúde baseadas na arquitetura da openEHR.

3 TRABALHOS RELACIONADOS

Esta seção lista os principais trabalhos realizados pela comunidade cientifica na área de desenvolvimento de aplicações de saúde baseadas na arquitetura da openEHR, incluindo os trabalhos correlatos acerca do mapeamento de arquétipos em SIS, mecanismos de persistência de arquétipos em banco de dados e frameworks voltados para o desenvolvimento de SIS.

3.1 ARQUÉTIPOS EM APLICAÇÕES DE SAÚDE

Diversos projetos de pesquisa e várias aplicações têm sido desenvolvidas a partir das especificações da arquitetura openEHR e do conceito de arquétipos [91-93]. Um dos trabalhos pioneiros foi desenvolvido em [58] para modelar um protótipo de RES para pacientes do departamento de neonatologia. O objetivo do trabalho consistiu em apoiar as tarefas de cuidados clínicos de bebês prematuros e recém-nascidos. Segundo os autores, o conceito de arquétipo reduz a necessidade de criação de novas documentações para os dados clínicos, sendo seu uso vital para a troca de informações e para interoperabilidade semântica em um hospital. No entanto, a falta de ferramentas de mapeamento de arquétipos em banco de dados foi identificada como uma das principais dificuldades relatadas.

De acordo com Lezcano *et. al* (2008) [55], o RES tem informações clínicas registradas ao longo de toda a vida do paciente. Esta informação é compartilhada por diversos sistemas heterogêneos, tornando-se, na maioria das vezes, sintaticamente ou semanticamente incompatível entre as aplicações de saúde. Existem diferentes padrões para a representação e o intercâmbio de informações do RES entre os diferentes sistemas, as quais comumente são representadas por meio de arquétipos e usando a linguagem ADL. No entanto, a especificação de arquétipos por meio da linguagem ADL possui desvantagens quando informações semânticas são processadas em ambiente web. Por isso, a especificação de arquétipos para a web por meio de uma ontologia definida na linguagem OWL (*Ontology Web Language*) foi proposta e uma solução que mescla tecnologias de web semântica e um modelo de conversão foi desenvolvida para converter arquétipos codificados em linguagem ADL para OWL. Além das dificuldades com a linguagem ADL, identificou-se nos resultados apresentados que os atributos de dados

representados por meio de estruturas hierárquicas de vários níveis (i.e., ITEM_TREE) não foram consideradas.

Em [57], investigou-se a viabilidade de se representar os conceitos clínicos de um sistema regional por meio de arquétipos e foi proposta a conversão de arquétipos para um formato proprietário. Um mapeamento semântico entre os modelos de referência especificados pela openEHR e o sistema de saúde regional foi realizado e um protótipo de software foi desenvolvido para realizar a conversão de arquétipos. Nota-se que as especificações openEHR são expressivas para representar conceitos clínicos e que os arquétipos disponíveis em âmbito internacional podem ser convertidos para um modelo de RES legado. Contudo, o trabalho fez uso apenas das terminologias especificadas nos arquétipos e não abordou a criação de esquemas de dados.

Embora o desenvolvimento de SIS utilizando as especificações da openEHR seja uma área de pesquisa multidisciplinar [94-102], percebe-se que na prática as organizações em saúde têm dificuldade em utilizar arquétipos e templates no desenvolvimento de suas aplicações. Isso ocorre principalmente pela falta de uma metodologia que expresse como os atributos de dados, as terminologias e as restrições dos arquétipos devem ser identificadas e extraídas e como, a partir destes elementos, os esquemas de dados e as interfaces gráficas de usuário para SIS podem ser criadas.

3.2 GERAÇÃO DE ESQUEMAS DE DADOS E INTERFACES GRÁFICAS

Conforme indicado em [3], os requisitos de dados do RES são dinâmicos, heterogêneos e mudam constantemente em virtude da evolução natural do ciclo de vida do RES. A mudança contínua dos requisitos de dados ocasiona a dependência de uma equipe de desenvolvimento de software para realizar as alterações no esquema de dados e, consequentemente, nas regras de negócios e nas interfaces da aplicação. Além disso, outro problema que afeta o armazenamento de dados em saúde é a falta de uniformidade para modelar e permitir a troca de informações do RES entre SIS[3].

Para minimizar alguns desses problemas, a solução proposta em [103] utilizou o sistema de banco de dados *Entity Attribute Value* (EAV) para gerenciar o armazenamento dos dados do RES. A abordagem EAV objetiva a representação dos requisitos de cuidados clínicos por meio de pares de atributos. Assim, uma tabela contendo as colunas do tipo

atributo e valor armazena as informações do RES. A vantagem desta abordagem é que se novos atributos de dados da aplicação surgirem, não há necessidade de se alterar o esquema de dados com a criação de novas colunas porque as instâncias dos dados são registradas como uma linha nos campos atributo e valor. No entanto, a recuperação de dados em um sistema de banco de dados EAV exige a criação de instruções complexas.

Um banco de dados clínicos baseados em XML provê armazenamento flexível e facilidade de representação de dados não estruturados como dados hierárquicos de diversos níveis [8]. No entanto, percebe-se que o uso de XML vem sendo comumente adotado como um mediador na troca de informações em saúde e que, na maioria das vezes, as organizações adotam uma solução mista, i.e., utiliza-se um banco de dados relacional como solução central e XML para facilitar a troca de informações entre as aplicações. Essa prática pode ser observada em [8], onde um framework facilita a inserção de novos requisitos de dados em formato XML em uma base de dados legada, e também, extrai dados de um banco relacional e os mapeia para um arquivo XML para trocar dados dos pacientes entre organizações de saúde.

Outros trabalhos criaram bancos de dados para o RES beneficiando-se de padrões já consolidados na área da saúde. Em [52], utilizou-se a especificação fornecida pela openEHR para modelar e reestruturar o RES em um sistema legado. Inicialmente, o esquema do banco de dados proprietário foi reestruturado para que cada campo das tabelas do banco de dados estivesse vinculado a um elemento do arquétipo. Contudo, a solução proposta não cria esquemas de dados, tampouco utiliza as terminologias e restrições dos arquétipos no banco de dados proprietário.

De forma similar, Georg, Judith and Christoph (2015) [59] propõem uma abordagem para mapear arquétipos em uma base de dados legada e gerar dinamicamente interfaces gráficas para uma aplicação em saúde chamada de *ArchiMed*. Um conjunto de regras de mapeamento foram implementadas em um algoritmo de modo a vincular cada elemento de dado do arquétipo com um campo de uma tabela no banco de dados. Os resultados apresentados indicam que a solução proposta não conseguiu mapear os arquétipos que continham estruturas de dados hierárquicas (i.e., ITEM_TREE).

Outra alternativa para criar um banco de dados utilizando arquétipos consiste em mapear os elementos de arquétipos definidos em ADL para um formato de dados reconhecido por um SGBD. Nesse sentido, a fundação openEHR disponibiliza uma

solução chamada de Node+Path [104] que utiliza o princípio da abordagem EAV para mapear os atributos de dados de um arquétipo em uma tabela de duas colunas. Assim, armazena-se a referência do atributo na primeira coluna, enquanto o valor é armazenado na segunda coluna. No entanto, apesar da simplicidade de armazenar hierarquias de dados, a solução não persiste os elementos dos arquétipos em diferentes sistemas de banco de dados.

Em [105], uma ferramenta é proposta para apoiar o mapeamento de arquétipos em banco de dados relacional. A solução consiste em um conjunto de regras de mapeamento que extraem os atributos de dados dos arquétipos e os armazenam em tabelas e colunas de dados. Além disso, estende-se a linguagem *Archetype Query Language* (AQL), adicionando os comandos de *insert, update* e *delete* para prover a manipulação de dados. Observando o esquema de dados gerado pela ferramenta, percebe-se que as restrições e terminologias especificadas nos arquétipos não são consideradas, não há definição das restrições de integridade referencial entre as tabelas e, trata-se de uma solução voltada para um sistema legado.

Freire et al. (2016) [106] avaliaram o desempenho de sistemas de bancos de dados NoSQL com relação à recuperação de dados clínicos gerados a partir das especificações da openEHR. Dois aspectos principais foram avaliados. Primeiro, o tamanho do espaço alocado para o armazenamento dos dados. Segundo, o tempo de processamento de um conjunto de consultas. Para o estudo proposto, quatro bancos de dados relacionais legados foram utilizados para gerar dados clínicos nos formatos XML e JSON. Os resultados apresentados mostraram que o espaço de armazenamento de JSON é menor que o espaço correspondente em XML. Além disso, foi verificado o baixo desempenho de documentos XML em relação a JSON, quando usados no processamento de um conjunto de consultas para recuperação dos dados.

Outros trabalhos de pesquisa têm investigado o armazenamento de dados clínicos em bancos de dados NoSQL[107-111]. No entanto, nenhum deles propõe a geração dinâmica de esquemas de dados utilizando arquétipos.

3.3 FRAMEWORKS PARA CONSTRUÇÃO DE APLICAÇÕES DE SAÚDE

EhrScape[1] é um framework baseado na arquitetura da openEHR que fornece um conjunto de recursos para construir serviços em saúde [112]. 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.

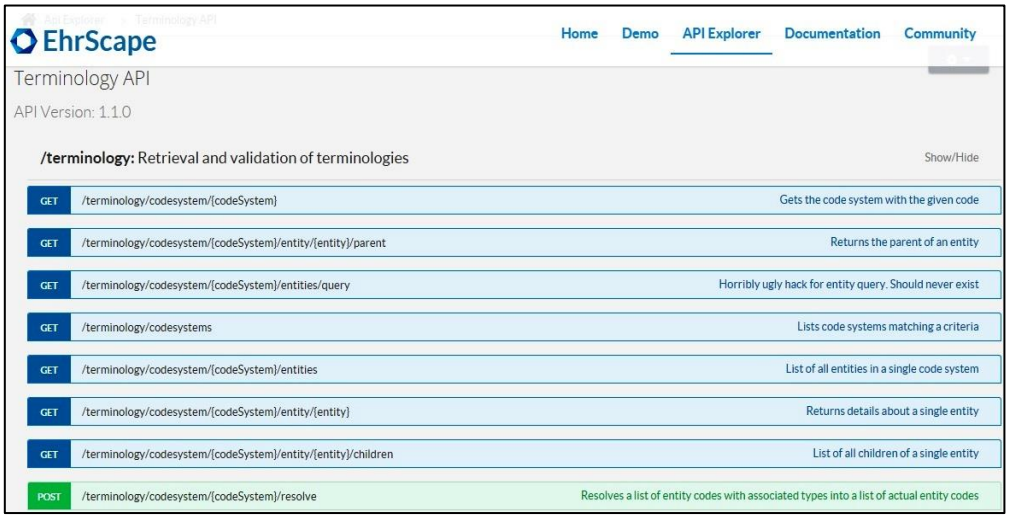


Figura 9 - Framework EhrScape [112]

Figura 10 - Framework EHRGen [113]

[1] https://www.ehrscape.com/

Como mostra a Figura 9, por meio de requisições GET e POST das API disponibilizadas pelo framework EhRScape é possível recuperar os elementos de um arquétipo em formato JSON e criar soluções que integrem aplicações legadas com as especificações openEHR. Contudo, o framework não cria esquemas de dados e nem permite que que os dados sejam persistidos em um sistema de banco de dados.

EHRGen[2] é um framework voltado para o desenvolvimento de aplicações que gerenciam o histórico clinico dos pacientes [113]. Suas principais características são: cria templates utilizando arquétipos e armazena os dados em uma solução proprietária. No entanto, *EHRGen* não permite a customização dos templates gerados e utiliza um único banco de dados para armazenar a heterogeneidade do RES. A Figura 10 mostra o ambiente de EHRGen para a criação de fichas clínicas a partir de arquétipos.

3.4 SOLUÇÕES DE PERSITÊNCIA POLIGLOTA

Persistência poliglota vem sendo aplicada nos mais diferentes domínios de problema. *Prasad* e *Avinash* (2014) [114] melhoraram um sistema de gestão de energia que lida com o armazenamento de diferentes tipos informações (e.g., dados transacionais, informações de sessões do usuário, séries temporais). O uso de um único sistema de banco de dados não atendia às necessidades do domínio de problema estudado, e assim, os autores desenvolveram um protótipo para que que os dados da aplicação fossem armazenados em um banco de dados relacional e NoSQL. Como resultado, obteve-se melhoria no desempenho da aplicação e na manutenção dos esquemas de dados.

Nos dias atuais, muitas aplicações necessitam interagir com diferentes sistemas de bancos de dados, por exemplo, relacionais, documentos, chave-valor, grafos, entre outros. O uso de diferentes sistemas de banco de dados aumenta a complexidade e impõe novos desafios aos programadores na busca por soluções que gerenciem a manipulação de dados (i.e., inserção, atualização, exclusão e consulta). *Sellami*, *Bhiri* e *Defude* (2014) [73] especificaram um conjunto integrado de modelos de dados, algoritmos e ferramentas destinadas a desenvolver, distribuir e migrar aplicações para utilizarem múltiplos sistemas de bancos de dados em nuvem. A solução proposta foi materializada em um

[2] https://code.google.com/p/open-ehr-gen-framework/

protótipo que permite a inserção, exclusão, atualização e consultas sobre os diferentes SGBD utilizados. Além disso, o protótipo desenvolvido foi utilizado para implementar funcionalidades do projeto openPaaS[3].

Outros projetos de pesquisa utilizaram o conceito de persistência poliglota para: i) especificar uma arquitetura PaaS de alta disponibilidade da *International Business Machines* (IBM) [71], ii) construir uma ferramenta para identificação automática de esquemas de dados a partir do código fonte da aplicação [72], e iii) automatizar as atividades de engenharia reversa de bases de dados legadas em diferentes sistemas de banco de dados [73]. Percebe-se que, em virtude do grande volume de dados processados pelos sistemas de informação, e pela necessidade de se criar soluções que armazenem os mais diferentes tipos de dados, o conceito de persistência poliglota vem sendo utilizado nas mais diferentes áreas [73].

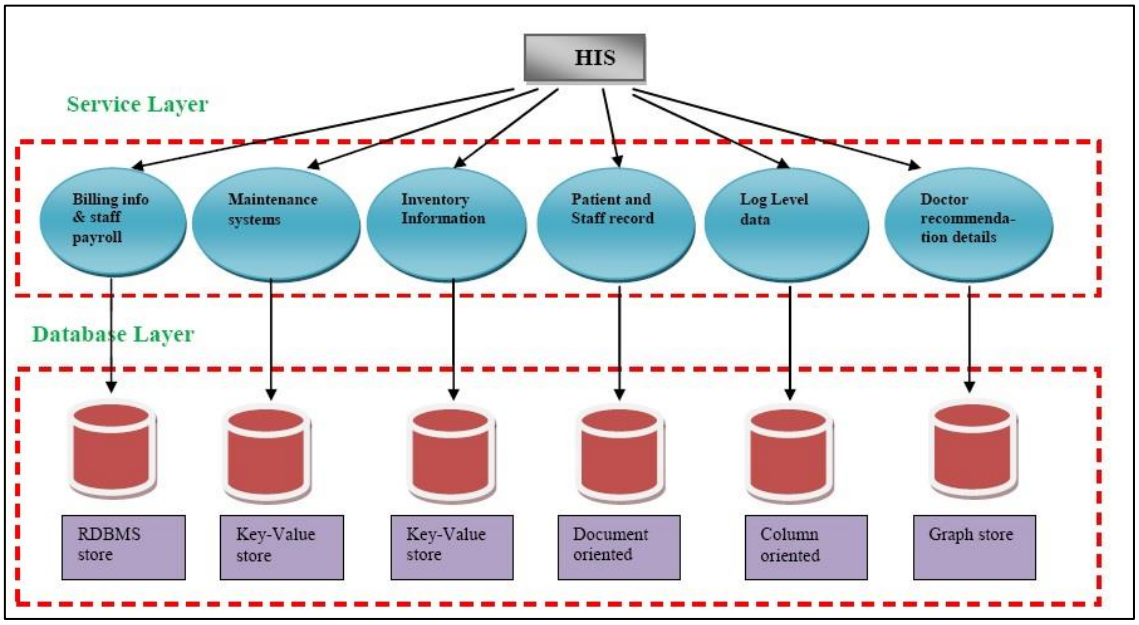


Figura 11 - Arquitetura de SIS Utilizando Persistência Poliglota [75]

Na área da saúde, a persistência poliglota representa uma alternativa para se armazenar a heterogeneidade dos dados representados nos diversos subsistemas (e.g., diagnóstico e terapia, clínica e assistencial, suprimentos, faturamento e gestão estratégica) que compõem um SIS. Para minimizar a rigidez causada pelos esquemas de dados relacionais e dar suporte às constantes mudanças de requisitos de dados ocorridas em um domínio da

[3] https://research.linagora.com/display/openpaas/Open+PAAS+Overview

saúde, Ellison e Calinescu (2014) [75] especificaram uma arquitetura de software (Figura 11) que remodela um SIS para usar o conceito de persistência poliglota.

Na Figura 11, cada área do domínio de problema estudado foi remodelado de modo que cada subsistema tenha um sistema de banco de dados adequado para armazenar dados estruturados e não estruturados. Um protótipo baseado na arquitetura proposta foi desenvolvido e o uso da persistência poliglota melhorou o gerenciamento dos dados em saúde.

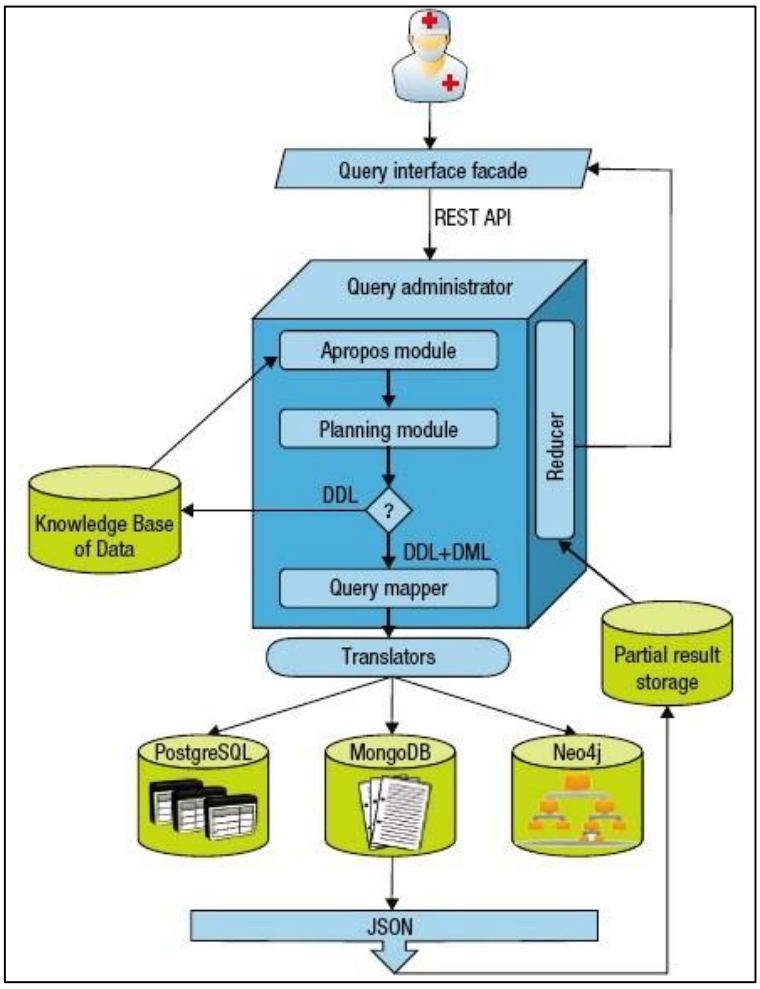


Figura 12 - Arquitetura de Persistência Poliglota[72]

Alguns projetos de pesquisa têm se dedicado à especificação de novos modelos de arquitetura para SIS, de modo que eles usem a persistência poliglota para armazenamento dos dados. Kaur e Rani (2015) [72] debatem a importância da adoção de diferentes sistemas de banco de dados em aplicações de saúde e exemplificam por meio da especificação de uma arquitetura de software como um SIS pode utilizar diferentes

SGBD. A arquitetura proposta em [72] (Figura 12) consiste no armazenamento de dados estruturados em um sistema de banco de dados relacional (i.e., *PostegreSQL*), enquanto que dois sistemas de banco de dados NoSQL (i.e., *MongoDB* e *Neo4j*) armazenam dados semiestruturados ou não estruturados, por exemplo, exames laboratoriais, prescrições médicas, entre outros. Contudo, as soluções de persistência poliglota identificadas no estado da arte não utilizam arquétipos para padronizar os atributos e as terminologias do RES, nem propõem a criação automática de esquemas de dados.

A especificação de um serviço para criar esquemas de dados em diferentes sistemas de banco de dados utilizando arquétipos e o uso da persistência poliglota para armazenar os dados do RES por meio das interfaces gráficas de usuário geradas por este serviço são temas abordados nesta tese.

3.5 FRAMEWORKS DE PERSITÊNCIA DE DADOS

Além da investigação sobre soluções de persistência poliglota aplicadas à diversas áreas de conhecimento, buscou-se também por frameworks de persistência de dados disponíveis na indústria de software. O intuito principal consistiu em verificar a possibilidade de extensão de um framework para gerar esquemas de dados em diferentes sistemas de bancos de dados utilizando arquétipos. Nesse sentido, analisou-se a arquitetura dos seguintes frameworks: *Entity Framework, Hibernate, Torque*, *OJB* e *Simple ORM* [115]. Como característica comum, os frameworks aqui citados dispõem de recursos que permitem a persistência de dados utilizando uma linguagem comum que independe da tecnologia de um SGBD e que inclui a associação, a herança, o polimorfismo, a composição e a estrutura de coleções de dados. Além disso, fornecem classes e interfaces que realizam a ponte entre os objetos das classes da camada de negócio e os SGBD que armazenam os dados. Contudo, optou-se por especificar um framework de geração de esquemas de dados em diferentes sistemas de banco de bancos em virtude do custo do desenvolvimento associado a extensão de um framework do mercado e, em razão da necessidade de se projetar uma arquitetura no qual os componentes de software baseiam-se em serviços na nuvem.

3.6 DISCUSSÃO SOBRE OS TRABALHOS INVESTIGADOS

A partir dos principais trabalhos investigados no estado da arte, apresenta-se nesta seção uma análise sobre as principais características de cada estudo e discute-se as principais contribuições desta tese.

Para facilitar o entendimento, o Quadro 3 apresenta os seguintes critérios de comparação: i) mecanismo de geração automática de esquemas de dados utilizando arquétipos; ii) geração de interfaces gráfica de usuário utilizando arquétipos; iii) persistência poliglota do RES; iv) arquitetura de software com representação do armazenamento em diferentes sistemas de banco de dados; v) geração de esquema de dados independente da tecnologia de um SGBD.

Como mostra o Quadro 3, somente EHRGen, Template4EHR e o trabalho proposto por Wang et al. [105] especificam um mecanismo para criação de esquemas de dados utilizando arquétipos. Contudo, esses trabalhos utilizam um único sistema de banco de dados para armazenamento do RES, não fazem uso das terminologias e restrições dos arquétipos, não estendem dinamicamente o esquema de dados a partir de novos arquétipos e são orientados a uma tecnologia de SGBD.

Quadro 3 - Análise comparativa dos Trabalhos Correlatos

Trabalho/Critério	Geração de Esquema de Dados	Geração de Interfaces Gráficas	Persistência Poliglota do RES	Arquitetura de Software	Independ. de Tecnologia
Späth e Grimson (2010)	✗	✗	✗	✗	✗
Georg et al.(2013)	✗	✓	✗	✗	✗
openEHR Node+Path	✗	✗	✗	✗	✗
Wang et al.(2015)	✓	✗	✗	✗	✗
Framework EhrScape	✗	✓	✗	✗	✗
Framework EHRGen	✓	✓	✗	✗	✗
Prasad e Sha (2013)	✗	✗	✗	✓	✗
Kaur e Rani (2015)	✗	✗	✗	✓	✗

A geração de interfaces gráficas de usuário utilizando arquétipos foi desenvolvida por Georg *et.al* (2013) [57], EhrScape e EHRGen. EhrScape permite a customização dos

componentes da interface, entretanto, não possui recursos de responsividade para acesso de qualquer dispositivo (e.g., computador, *tablet* e *smartphone*).

Já quanto ao recurso de persistência poliglota utilizando arquétipos, nenhum dos trabalhos investigados permite a persistência do RES em diferentes sistemas de bancos de dados a partir das interfaces gráficas geradas. A especificação de uma arquitetura de software que ilustre como aplicações de saúde podem fazer uso da persistência poliglota foi desenvolvida por *Prasad* e Sha (2013) [114] e *Kaur* e *Rani* (2015) [72]. No entanto, eles não fazem uso de arquétipos. Por fim, os trabalhos descritos no Quadro 3 não atendem ao requisito de geração de esquemas de dados deve independente de uma tecnologia de SGBD.

3.7 CONSIDERAÇÕES FINAIS

Neste seção, foi apresentada uma visão geral sobre os conceitos básicos utilizados no desenvolvimento desta tese, incluindo os trabalhos correlatos acerca de: i) desenvolvimento de aplicações de saúde baseadas na arquitetura da openEHR; ii) construção de esquemas de dados para o armazenamento do RES; iii) frameworks voltados para a geração de aplicações e serviços de saúde, por fim, iv) persistência poliglota do RES. Desses trabalhos, seis embasam as principais contribuições da tese descrita neste documento. Späth e Grimson (2010) [52] abordam a necessidade de se criar soluções automatizadas para o mapeamento de arquétipos em aplicações de saúde, Georg et al. (2013) [57] enfatizam a importância da geração de interfaces gráficas de usuário por parte dos próprios profissionais de saúde, e a openEHR Node+Path [104] propõe a extração dos atributos de dados dos arquétipos para criação de esquemas de dados. Wang et.al (2015) [105] mapeiam arquétipos em banco de dados para um domínio de problema específico, enquanto que os frameworks EhrScape e EHRGen propõem a criação de aplicações e serviços em saúde utilizando arquétipos. Por fim, *Prasad* e *Sha* (2013) [114] e *Kaur* e *Rani* (2015) [72] ressaltam a necessidade das aplicações de saúde utilizarem a persistência poliglota do RES.

Na seção 4, apresenta-se a arquitetura de Template4EHR, e discute-se como os esquemas de dados são gerados a partir da extração dos atributos de dados, das terminologias e das restrições dos arquétipos. Além disso, descreve-se por meio de um

conjunto de regras de mapeamento como as interfaces gráficas de usuários são geradas dinamicamente.

4 O FRAMEWORK TEMPLATE4EHR

Esta seção apresenta o framework desenvolvido para geração de esquemas de dados e interfaces gráficas de usuário utilizando arquétipos. A Seção 4.1 ilustra por meio de um metamodelo como a arquitetura dual da openEHR modela o RES, enquanto que a Seção 4.2 discute a arquitetura de software de Template4EHR. A Seção 4.3 descreve como são gerados os esquemas de dados relacionais e NoSQL e a Seção 4.4 especifica um conjunto de regras de mapeamento para a geração automática de interfaces gráficas de usuário. A Seção 4.5 apresenta um aplicativo móvel que reutiliza e executa as interfaces gráficas criadas por Template4EHR, enquanto que a Seção 4.6 exemplifica a criação de uma instância de aplicação de saúde utilizando o framework proposto. Por fim, a Seção 4.7 lista as limitações de Template4EHR e a Seção 4.8 traz as considerações finais de Template4EHR.

4.1 METAMODELO PARA CRIAÇÃO DE ESQUEMAS DE DADOS

O desafio inicial para a criação de esquemas de dados utilizando arquétipos consiste em entender como a arquitetura dual da openEHR está organizada, e quais relacionamentos existem entre os seus conceitos. Nesse sentido, apresenta-se nesta seção um metamodelo em UML que representa os conceitos, os relacionamentos e as restrições existentes na especificação do RES por meio de arquétipos. O metamodelo ilustrado na Figura 13 estende a proposta de [116] e acrescenta as classes e os relacionamentos das estruturas de dados genéricas, tipos de dados, restrições e terminologias [117]. A definição de cada classe é dada a seguir.

A classe *Composition* define os metadados que caracterizam um esquema de dados, e agrupa uma ou mais seções (i.e., Classe *Section*). Uma seção representa um tema ou um contexto da aplicação que se deseja modelar e organiza todos os demais conceitos do RES. A classe *EHR* é composta por todas as informações de cuidados clínicos do paciente e agrega dados das seguintes classes: *CareEntry* e *DataStructure*. A classe *CareEntry* modela qualquer serviço em saúde que tem como objetivo avaliar e tratar pacientes, registrando informações sobre os cuidados clínicos e o histórico de saúde (e.g.,

diagnósticos, atividades físicas, queixas, hábitos alimentares). Para caracterizar cada conceito clínico, a classe *CareEntry* é especializada nas seguintes classes: *Observation*, *Instruction*, *Activity*, *Action* e *Evaluation*, que são detalhadas a seguir.

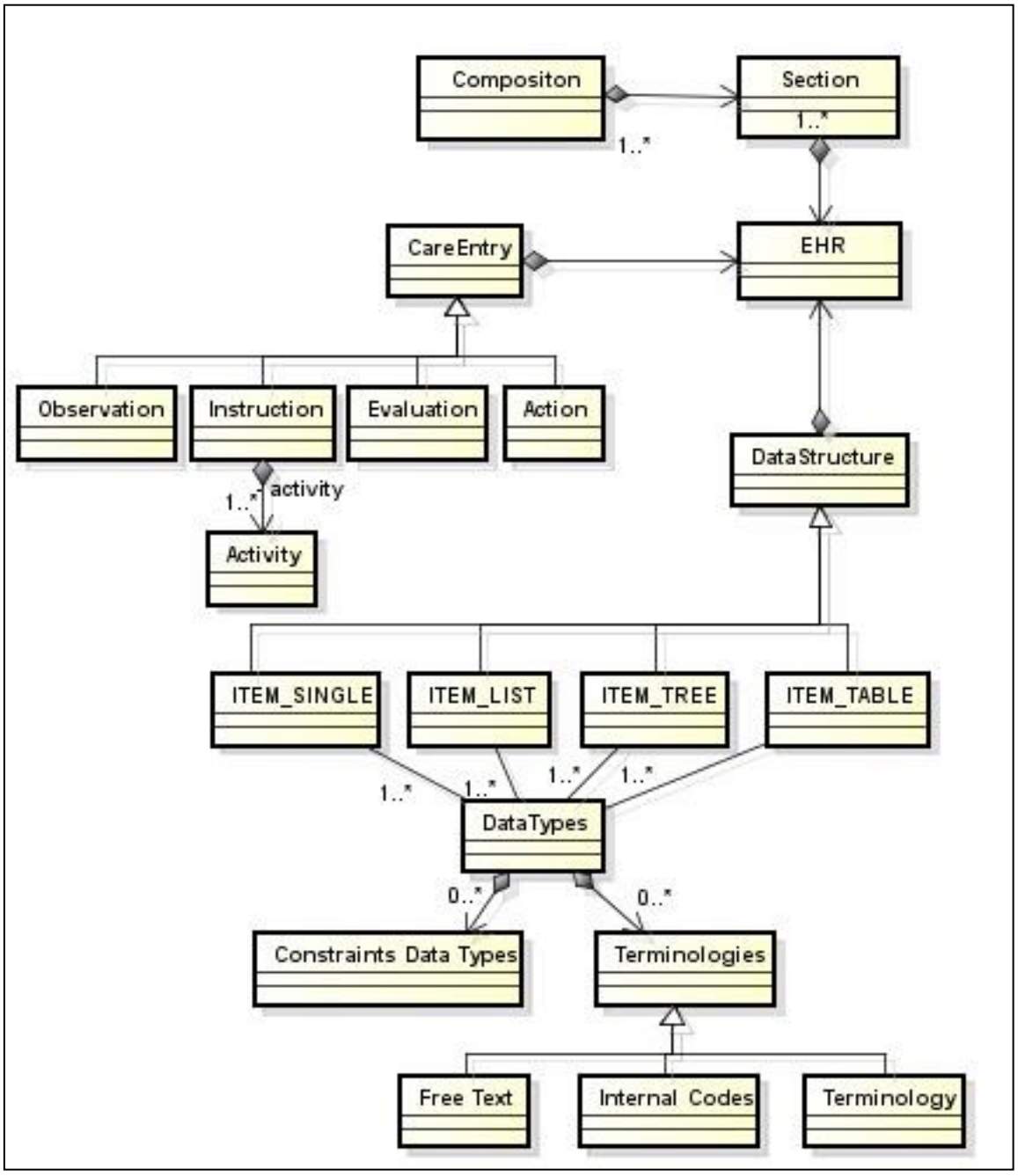


Figura 13 - Metamodelo para Representação do RES modelado por Arquétipos

A classe *Observation* representa informações sobre qualquer evento associado com a saúde ou o estado clínico do paciente, enquanto a classe *Instruction* modela declarações narrativas sobre as atividades de cuidados clínicos realizadas no paciente. A classe

Activity define informações que compõem uma atividade em saúde, devendo uma instância desta classe estar sempre relacionada a uma instância da classe *Instruction*. A classe *Action* contém o registro de todas as ações realizadas para cada atividade prescrita no atendimento clínico do paciente. Por fim, a classe *Evaluation* modela informações que avaliam ou investigam o diagnóstico ou risco de um paciente.

A classe *DataStructure* representa os construtores de modelagem utilizados para especificar os atributos de dados em um arquétipo e é especializada pelas classes: ITEM_SINGLE, ITEM_LIST, ITEM_TREE e ITEM_TABLE. ITEM_SINGLE especifica um único atributo de dado, enquanto o ITEM_LIST organiza um conjunto de atributos de dados dispostos verticalmente. ITEM_TREE permite a criação de uma estrutura hierárquica de vários níveis para representação dos atributos de dados. Por fim, ITEM_TABLE organiza os atributos por meio de linhas e colunas.

A classe *DataType* fornece um conjunto de tipos de dados que caracterizam cada atributo pertencente às estruturas de dados genéricas (i.e., ITEM_SINGLE, ITEM_LIST, ITEM_TREE ITEM_TABLE). Cada atributo de dado pode ser definido pelos tipos: DV_TEXT, DV_CODEDTEXT, DV_BOOLEAN, DV_COUNT, DV_INTERVAL, DV_DATE_TIME, DV_DATE, DV_TIME, DV_QUANTITY, DV_DURATION, DV_ORDINAL e DV_URI. Além disso, os atributos de dados possuem um conjunto de restrições de domínio que são representadas pelas instâncias da classe *Constraints Data Types.* A classe *Terminologies* modela todo o conhecimento do segundo nível da modelagem dual proposta pela openEHR e suas instâncias devem estar relacionadas com a informação que é especificada por meio das estruturas de dados genéricas (i.e., primeiro nível do modelo dual). Assim, a classe *Terminologies* define a terminologia utilizada na área da saúde, uma codificação interna de vocabulários ou uma informação textual definida pelo especialista do domínio, as quais são representadas respectivamente, pelas classes: *Terminology, Internal Code, e Free Text.*

A partir do entendimento de como os conceitos da modelagem dual estão organizados, principalmente, como estão dispostos e relacionados os atributos de dados, as terminologias e as restrições, descreve-se nas demais seções deste documento, como construir esquemas de dados e interfaces gráficas de usuários a partir de um padrão em saúde (i.e., arquétipos) que uniformiza o RES.

4.2 VISÃO GERAL E ARQUITETURA DO FRAMEWORK

O framework proposto neste trabalho consiste em um ambiente computacional voltado para a construção de esquemas de dados e interfaces gráficas de usuário a partir da especificação do RES existente nos arquétipos [118]. O ponto principal da abordagem aqui proposta consiste na extração dos atributos de dados, das terminologias e das restrições dos arquétipos para armazenamento do RES em bancos de dados relacionais e NoSQL [119,120]. A Figura 14 mostra uma visão conceitual do funcionamento de Template4EHR e a definição de cada componente é dada a seguir.

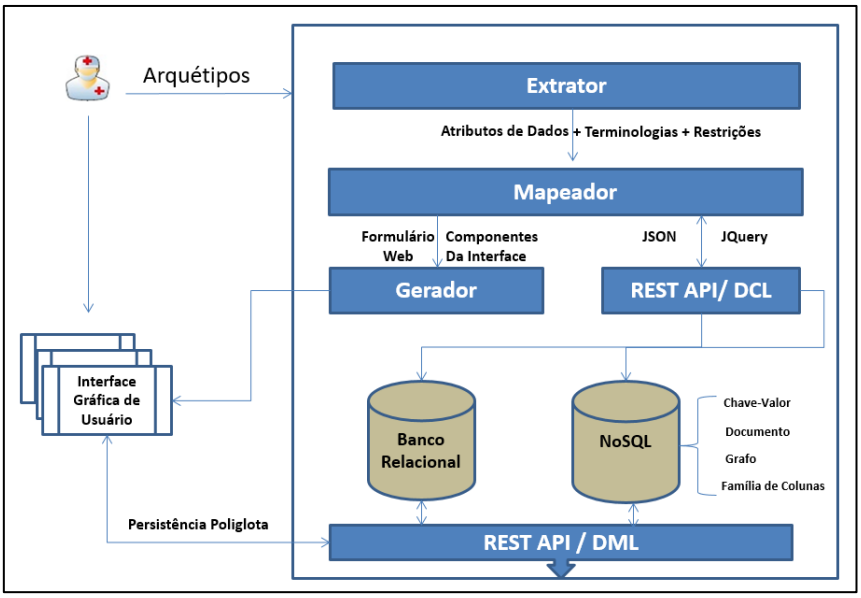


Figura 14 - Visão conceitual de Funcionamento de Template4EHR.

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* mostrado na Figura 14 retira do arquétipo informado os atributos que definem o RES, as terminologias e os vocabulários em saúde que dão significado semântico aos dados clínicos, além das restrições especificadas sobre os atributos. 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 posso 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. Terminada essa atividade, o componente *Gerador* organiza os componentes criados e, em seguida, disponibiliza a interface gráfica para uso.

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.

A Figura 15 mostra a funcionalidade de configuração de criação de esquema de dados (i.e., relacional ou NoSQL) de Template4EHR. Nela podem ser informados os parâmetros (e.g. servidor, usuário, senha, e nome para o banco de dados) que possibilitam a conexão de Template4EHR com diferentes tipos de SGBD.

Para garantir que os esquemas de dados gerados por Template4EHR sejam independentes de tecnologia, utilizou-se um padrão arquitetural que permite que as regras de negócios da aplicação acessem a camada de persistência de dados sem que a mesma tenha conhecimento da tecnologia do SGBD utilizada [121]. Chamado de repositório, este padrão encapsula o conjunto de objetos persistidos na camada de dados e as operações realizadas sobre eles. Além disso, a mediação entre os componentes das camadas de apresentação, regras de negócio, repositório e acesso a dados é realizada por meio de interfaces.

Figura 15 - Configuração de geração de Esquemas de Dados

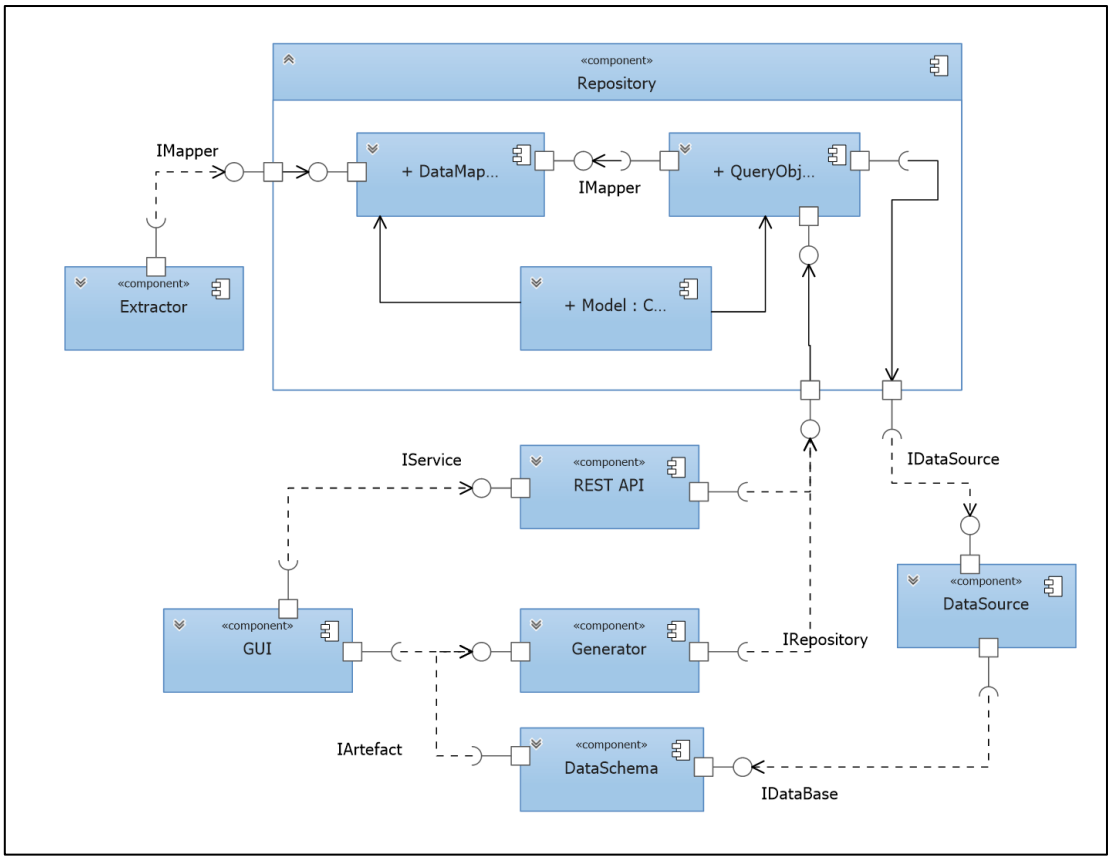


Figura 16 - Arquitetura de software de Template4EHR

A Figura 16 mostra a arquitetura de software criada para Template4EHR utilizando a notação do diagrama de pacotes da UML e do formalismo do padrão repositório. Nela é possível observar que os componentes se comunicam por meio de interfaces. A principal vantagem do uso de interfaces é que elas estabelecem a forma de comunicação entre os componentes e permitem que modificações sejam feitas nas regras de negócio sem afetar o funcionamento das entidades externas que interagem com o framework. Por exemplo, é possível que uma aplicação externa esteja consumindo os métodos de geração de esquemas de dados relacionais do componente *DataSchema* (Figura 16), enquanto que novos métodos estejam sendo implementados por meio do componente *Generator* para criar esquemas de dados NoSQL.

A Figura 17 exemplifica o funcionamento das interfaces especificadas na arquitetura de Template4EHR. A interface *IRepository* contém as assinaturas dos métodos de persistência a serem implementadas nos diferentes sistemas de bancos de dados (e.g., Oracle, SQL Server, Cassandra, Neo4J). A interface *IMapper* fornece os contratos de assinatura para que as classes da camada de negócio realizem o mapeamento dos

elementos dos arquétipos em esquemas de dados. Por fim, a interface *IDataSource* disponibiliza os parâmetros de conexão com diversos tipos de SGBD.

```
public interface IRepository<T> where T : class
{
    void Insert(T entity);
    void Update(T entity);
    void Delete(Func<T, bool> predicate);
    T Find(Expression<Func<T, bool>> predicate);
    IQueryable<T> Get(Expression<Func<T, bool>> predicate);
    IQueryable<T> GetAll();
    void SaveChanges();
}
public interface IMapper
{
    O ExtractFromXml<O, I>(I param) where I : XmlDocument;
    O ExtractFromAdl<O, I>(I param) where I : AdlObject;
}

public interface IDataSource : IObjectContextAdapter, IDisposable
{
    string Server { get; set; }
    string Port { get; set; }
    string UserId { get; set; }
    string Password { get; set; }
}
```

Figura 17 - Exemplo de Interfaces Implementadas na Arquitetura de Software

Toda interface gráfica de usuário criada dinamicamente pelo framework possui recursos de persistência de dados (i.e., inserção, atualização, consulta e exclusão). Como o framework trabalha com armazenamento em diferentes sistemas de banco de dados (i.e., persistência poliglota) a REST API/DML (Figura 14) captura os dados informados nas interfaces gráficas e persiste os mesmos nos diferentes bancos de dados criados por Template4EHR. Ressalta-se aqui que a persistência poliglota ocorre por intermédio da REST API/DML, não havendo comunicação direta entre os SGBD.

Além da geração de interfaces gráficas de usuário e esquemas de dados, Template4EHR possui um conjunto de outros recursos que permitem a criação de uma aplicação de saúde utilizando arquétipos. A Figura 18 mostra as principais

funcionalidades disponíveis no framework e a definição de cada uma delas é dada a seguir.

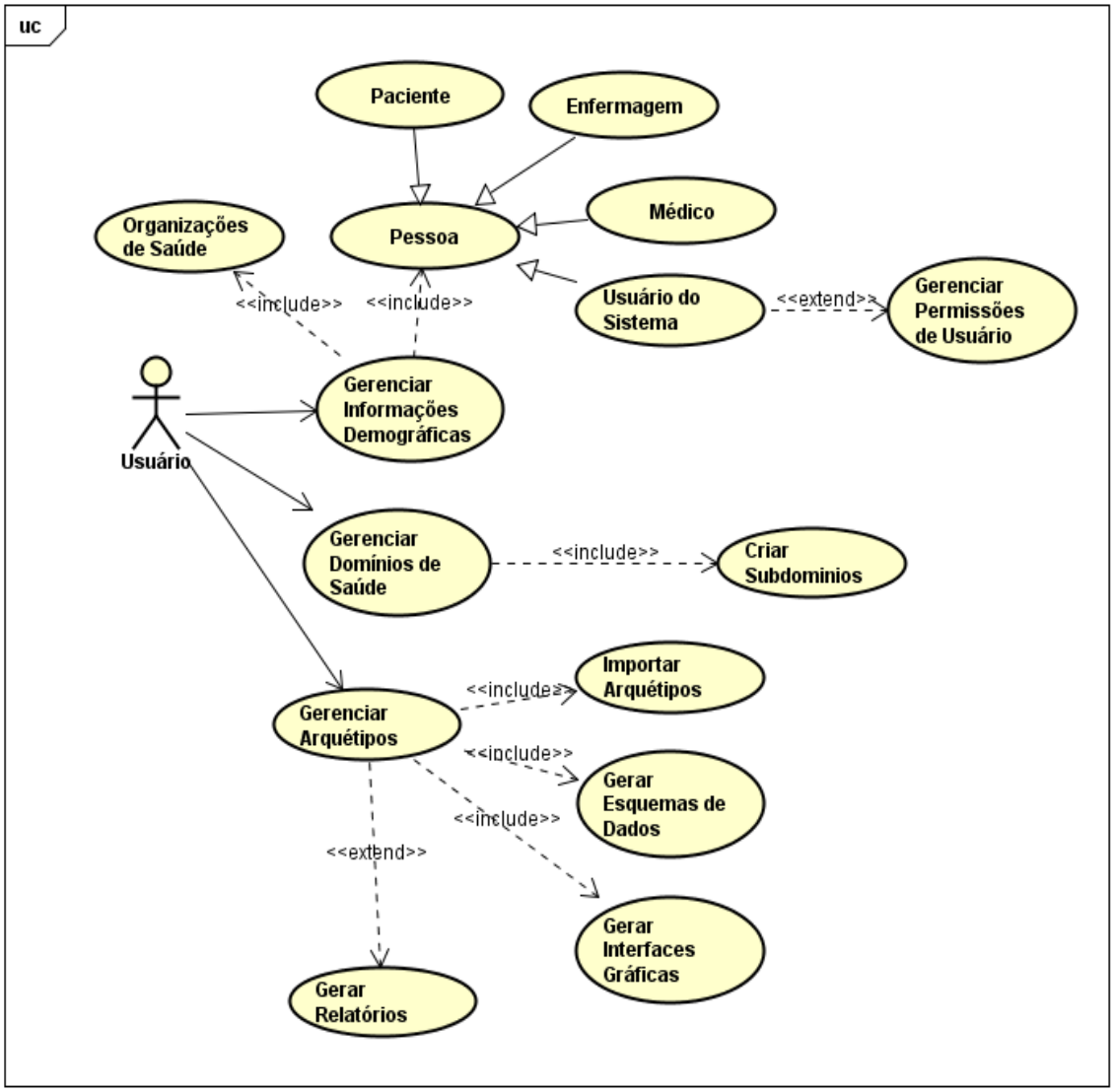


Figura 18 - Principais funcionalidades do Framework

Gerenciamento de informações demográficas: Esta funcionalidade permite o gerenciamento de organizações em saúde, pacientes, médicos, equipe de enfermagem e usuários de uma instância de aplicação criada pelo framework. Em um domínio da saúde, uma organização pode ser caracterizada como um hospital, uma clínica, um laboratório de análises clínicas, uma unidade básica de saúde, entre outros. Nesse contexto, o framework cria instâncias da aplicação por organização, isto é, pode-se ter uma instância de uma aplicação voltada para a internação de pacientes em um dado hospital, e outra instância da aplicação voltada para o atendimento ambulatorial de pacientes em uma

unidade de pronto atendimento. Além disso, as informações dos pacientes, médicos e equipe de enfermagem gerenciadas no framework podem ser adicionadas às interfaces gráficas de usuário geradas por Template4EHR.

Criação de Domínios em Saúde: Uma organização pode oferecer diversos tipos de serviços em saúde à sociedade. Por exemplo, um hospital pode realizar exames laboratoriais, diagnóstico por imagem, atendimento de urgência, internação, entre outros. Pensando nisso, o framework proposto permite criar e configurar na aplicação domínios e subdomínios que representam os serviços ofertados por cada organização. Nesse caso, utiliza-se os domínios e subdomínios criados para organizar e acessar as interfaces gráficas geradas pelo framework. Além disso, esta funcionalidade pode ser utilizada para criar uma sequência de atividades a serem realizadas por profissionais de saúde, por exemplo, uma aplicação pode ter um domínio chamado *Enfermagem*, no qual os profissionais dessa área devem realizar as atividades de avaliação de sinais vitais, balanço hídrico e exame físico do paciente.

Gerenciamento de Arquétipos: Esta funcionalidade permite importar e mapear os arquétipos para uma aplicação de saúde. Ao importar um arquétipo, o framework permite que o especialista do domínio escolha quais atributos de dados do arquétipo farão parte do esquema de dados e da interface da gráfica. Mesmo após ter escolhido os atributos de dados do arquétipo, pode-se gerenciar a interface gráfica gerada; acrescentando, retirando ou desabilitando os atributos. Nesse caso, o framework estende automaticamente o esquema de dados criado. Terminada esta atividade, deve-se associar a interface gráfica gerada com um domínio/subdomínio da aplicação e gerenciar as permissões de acesso dos usuários que terão acesso à aplicação criada. Além disso, permite-se que os próprios usuários criem seus relatórios a partir dos campos e das informações armazenadas no banco de dados.

Para a implementação do framework aqui proposto, utilizou-se a linguagem de programação *C#* e a tecnologia *ASP.NET* para a geração das interfaces gráficas na web. *Jquery* e *Cascading Style Sheets* (CSS) definiram e padronizaram a aparência dos componentes de interação do usuário final, e a notação JSON foi utilizada para a troca de dados entre os componentes internos do framework, e as aplicações externas que consomem os serviços em nuvem. Os SGBD *MS SQL Server* e Oracle foram utilizados para geração de esquemas de dados relacionais, enquanto que as tecnologias *Cassandra*

e *ArangoDB* foram utilizadas para a geração de bancos NoSQL. A tecnologia *ASP.NET* e a linguagem C# foram escolhidas para o implementar o framework descrito neste documento em virtude do domínio e conhecimento do autor desta tese nas soluções de desenvolvimento da Microsoft.

4.3 GERAÇÃO DE ESQUEMAS DE DADOS ARQUETIPADOS

Esta seção descreve e discute a geração de esquemas de dados utilizando arquétipos e está organizada da seguinte forma. A Seção 4.3.1 explica a criação de esquemas de dados relacionais, enquanto a Seção 4.3.2 apresenta a geração de esquemas de dados em banco de dados NoSQL.

4.3.1 Esquemas de Dados Relacionais

A geração de esquemas de dados relacionais realizada por Template4EHR consiste na criação de tabelas, de restrições de integridade referencial e de campos, a partir da extração dos elementos de arquétipos [122]. Dois algoritmos são responsáveis por realizar essa atividade. O primeiro extrai os atributos de dados, as terminologias e as restrições do arquétipo, enquanto o segundo recebe como parâmetros de entrada, os elementos extraídos do primeiro algoritmo e cria o esquema de dados correspondente. As Figuras 19 e 20 mostram os dois algoritmos, e a explicação de ambos é dada a seguir.

O Algoritmo 1 inicia sua execução recebendo como parâmetro de entrada um arquétipo XML/ADL e, em seguida, cria uma instância do arquétipo em memória para validar e extrair as informações necessárias. Os metadados do arquétipo descrevem como os elementos estão organizados e quais associações existem entre eles e, por isso, estas são as primeiras informações a ser extraídas. Na linha 7, a variável *Metadata* armazena os metadados extraídos do arquétipo por meio da função *GetMetaData.*

De posse da instância do arquétipo XML/ADL em memória, passa-se então a identificação, validação e extração dos atributos de dados, dos tipos de dados, das terminologias e das restrições. Para isso, desmembrou-se essa atividade em duas etapas. Primeiro, extrai-se todos os elementos do arquétipo por meio da função *GetElements* e em seguida armazena-se os itens encontrados na variável *Elements* da linha 11. Depois,

percorre-se os itens armazenados na variável *Elements* e identifica-se os atributos de dados, os tipos de dados, as terminologias e as restrições.

```
1  Algorithm 1 - Extractor
2  Input: An Archetype XML
3  Output: Void
4  BEGIN
5   READ xml file
6   SET MetaData as Array to empty
7   STORE GetMetaData in MetaData
8   SET DataStructures as Array to empty
9   SET Terminologies as Array to empty
10  SET Elements as Array to empty
11  STORE GetElements in Elements
12  SET i to 1
13  FOR each node on the Elements
14     SET Index GET index on the node
15     SET Description GET description on the node
16     SET DataType GET datatype on the node
17     SET Constraint GET constraint on the node
18     SET DataStructure as Array to Empty
19     STORE Index in DataStructure[i]
20     STORE Description in DataStructure[i]
21     STORE DataType in DataStructure[i]
22     STORE Constraint in DataStructure[i]
23     STORE DataStructure in DataStructures
24     SET Terms GET Terminologies on the node
25     SET j to 1
26     FOR each node2 on the Terms
27         SET Child GET child on the node2
28         SET Parent GET parent on the node2
29         SET Value GET value on the node2
30         SET Terminology as Array to Empty
31         STORE Child in Terminology[j]
32         STORE Parent in Terminology[j]
33         STORE Value in Terminology[j]
34         STORE Terminology in Terminologies
35         INCREMENT j
36     END FOR
37     INCREMENT i
38  END FOR
39  CALL Generate Relational DataSchema
40 END
```

Figura 19 - Algoritmo para extração dos elementos de arquétipos

O laço iniciado na linha 13 e finalizado na linha 38, captura somente os elementos que possuem tipos de dados. Os itens que possuem tipos de dados são caracterizados como elementos que permitem a entrada de dados em um arquétipo (i.e., estruturas de dados). Nesse caso, esses elementos são adicionados ao esquema de dados na forma de colunas. Para cada elemento que possui um tipo de dado definido, armazena-se sua posição (i.e., índice), sua descrição, seu tipo e suas restrições encontradas. A variável *DataStructures*, localizada na linha 23 do Algoritmo 1, armazena esse novo conjunto de elementos, contendo os atributos de dados, os tipos de dados e as restrições.

Em um arquétipo, um atributo de dado pode conter uma ou mais terminologias. A organização dessas terminologias se dá por meio de uma estrutura hierárquica de vários níveis. Para saber qual terminologia pertence a qual atributo de dado, faz-se necessário

identificar o elemento pai e seus respectivos filhos. O laço iniciado na linha 26 identifica e classifica as terminologias em uma hierarquia de elementos do tipo pai e filho. No final da execução do laço (linha 36), a lista de terminologias é armazenada na variável *Terminologies*, como mostra a linha 34 do Algoritmo 1.

Na linha 39 do Algoritmo 1, tem-se a chamada ao Algoritmo 2 (Figura 20), que é responsável por gerar as instâncias do esquema de dados relacional. Os elementos extraídos do arquétipo (i.e., metadados, atributos de dados, terminologias e restrições) são passados como parâmetros de entrada para o Algoritmo 2, e assim, cria-se as tabelas, colunas e as restrições de integridade referenciais.

```
1  Algorithm 2 - Generate Relational Schema
2  Input: MetaData, Elements, DataStructures, Terminologies
3  Output: Void
4  BEGIN
5   CREATE the Table 'Archetype'
6   CREATE the Table 'Archetype_Details'
7   CREATE the Table 'Terminology'
8   CREATE the Table 'Item_Table'
9   CREATE the Table 'Item_Tree'
10  CREATE the Table 'Data_Item'
11  ADD DataStructures Columns in 'Data_Item' Table
12  CREATE Referencial Integrity One-To-Many 'Archetype' to 'Archetype_Details'
13  CREATE Referencial Integrity One-To-Many 'Archetype' to 'Terminology'
14  CREATE Referencial Integrity One-To-Many 'Archetype' to 'Item_Table'
15  CREATE Referencial Integrity One-To-Many 'Archetype' to 'Item_Tree'
16  CREATE Referencial Integrity One-To-Many 'Archetype' to 'Data_Item'
17  CREATE Referencial Integrity One-To-Many 'Archetype_Details' to 'Terminology'
18  CREATE Referencial Integrity One-To-Many 'Archetype_Details' to 'Item_Table'
19  CREATE Referencial Integrity One-To-Many 'Archetype_Details' to 'Item_Tree'
20  CREATE Referencial Integrity One-To-Many 'Archetype_Details' to 'Data_Item'
21 END
```

Figura 20 - Algoritmo para Geração de Esquemas de Dados Relacionais

A Figura 21 mostra uma instância do esquema de dados relacional criado após a execução dos dois algoritmos. No exemplo citado, foram utilizados os arquétipos *Blood pressure* e *Apagar*, ambos disponíveis no repositório da openEHR.

O esquema de dados ilustrado na Figura 21 utiliza 6 tabelas, 5 restrições de integridades referencias e um conjunto de campos que armazenam os elementos dos arquétipos. A tabela *Archetype* armazena os metadados de configuração do arquivo informado, como por exemplo, o tipo do arquétipo, o autor e a versão. A tabela *Archetype_Details* armazena o tipo da estrutura de dados e restrições encontradas no arquétipo, enquanto a tabela *Terminology*, armazena a referência dos atributos de dados com suas respectivas terminologias, caso elas existam.

A especificação dos atributos de dados de um arquétipo pode ser realizada na forma de um único atributo, uma lista vertical de atributos, uma estrutura de dados hierárquica ou uma tabela com linhas e colunas. Para armazenar as diferentes formas que os atributos de dados são organizados em um arquétipo, utiliza-se as tabelas *Item_Tree* e *Item_Table*. Como a organização de uma estrutura hierárquica já contempla a definição de um único atributo ou um conjunto de atributos, mapeia-se os atributos dos tipos ITEM_SINGLE, ITEM_LIST e ITEM_TREE para a tabela *Item_Tree*. Como o nome sugere, a tabela ITEM_TABLE mapeia os atributos dos arquétipos que são organizados em linhas e colunas. Uma vez mapeados os atributos de dados em suas respectivas tabelas, o usuário pode escolher quais atributos de dados serão utilizados na geração das interfaces gráficas e, consequentemente armazenarão os dados manipulados pelos usuários finais. Nesse caso, os atributos escolhidos são adicionados em tempo de execução, como colunas na tabela *Data_Item*.

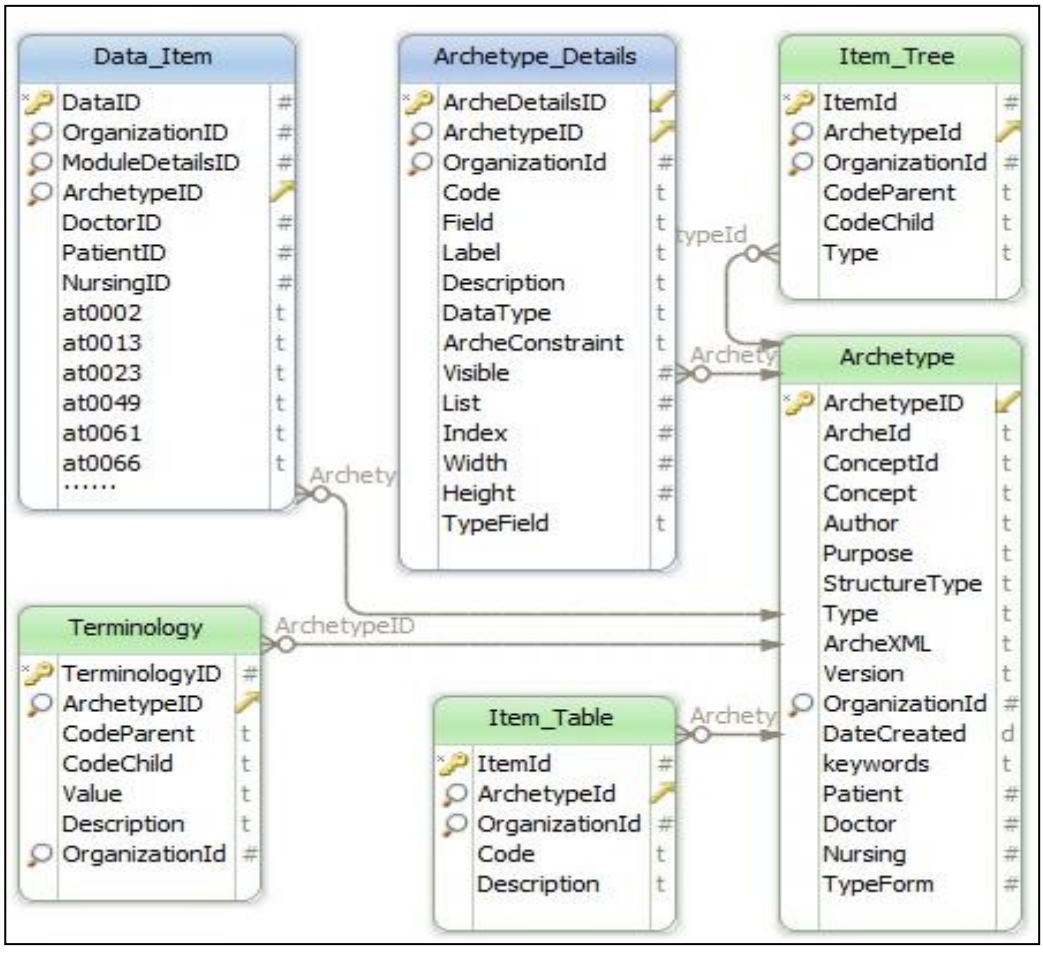


Figura 21 - Esquema de Dados Relacional criado a partir de Arquétipos

Cada elemento de um arquétipo é referenciado por um identificador. Todo identificador é iniciado pelas letras *at* e seguido por um valor sequencial. Por exemplo, o identificador *at000* armazena o nome dado a um arquétipo. Utiliza-se a nomenclatura definida por esse identificador para criar o nome dos campos no esquema de dados. Toda vez que um novo campo for inserido na tabela *Data_Item*, o framework verifica se o mesmo já existe, caso não exista o campo é adicionado.

4.3.2 Esquemas de Dados NoSQL

O armazenamento em diferentes sistemas de bancos de dados proposto no framework é realizado da seguinte forma. Os dados de natureza estruturada, como por exemplo, as informações demográficas (i.e., pacientes, médicos, enfermagem) e os domínios e subdomínios que são criados para agrupar as interfaces gráficas, são armazenados em um banco de dados relacional. Já os dados do RES especificados nos arquétipos são armazenados em quatro diferentes sistemas de banco de dados NoSQL[123].

A partir da identificação do tipo da estrutura de dados utilizada para especificar os atributos de um arquétipo (i.e., ITEM_SINGLE, ITEM_LIST, ITEM_TREE e ITEM_TABLE), cria-se os esquemas de dados obedecendo aos critérios definidos na Figura 22.

O ITEM_SINGLE é representado em um arquétipo por um único atributo, enquanto o tipo ITEM_LIST agrupa um conjunto de atributos representados fisicamente como uma lista vertical. Nesse caso, em virtude da organização dos atributos, utiliza-se o sistema de banco de dados chave-valor para armazenar os tipos ITEM_SINGLE e ITEM_LIST.

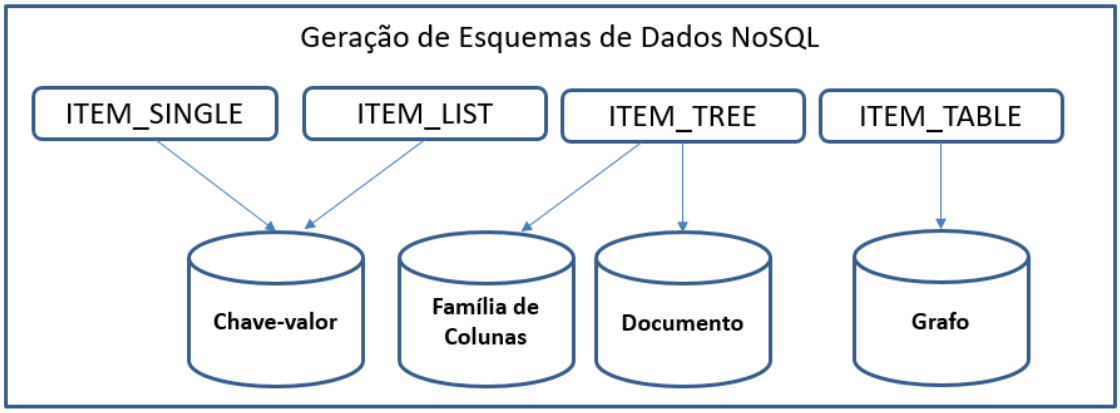


Figura 22 - Geração de Esquemas em Banco de Dados Heterogêneos

O ITEM_TREE permite que os atributos de dados sejam representados por meio de uma estrutura hierárquica de vários níveis em um arquétipo. Para garantir que a mesma representação seja mantida, utiliza-se o formato de armazenamento baseado em documento ou família de colunas. Os algoritmos das Figuras 23 e 24 descrevem como ocorre a de geração de esquemas de dados nesses dois sistemas de banco de dados NoSQL.

```
1  Algorithm 3 - Document Generator
2  Input: MetaData, Elements, DataStructures, Terminologies
3  Output: Void
4  BEGIN
5   INITIALIZE DataBase to DocumentsDataBase
6   CREATE Terminology as Collection
7   CREATE Constraints as Collection
8   CREATE Attributes  as Collection
9   SET i to 1
10  FOR each node on the DataStructures
11     SET Index GET index on the node
12     SET Description GET description on the node
13     SET DataType GET datatype on the node
14     SET Constraint GET constraint on the node
15     STORE Index in Attributes[i]
16     STORE Description in Attributes[i]
17     STORE DataType in Attributes[i]
18     STORE Constraint in Constraints[i]
19     SET Terms GET Terminologies on the node
20     SET j to 1
21     FOR each node2 on the Terms
22         SET Child GET child on the node2
23         SET Parent GET parent on the node2
24         SET Value GET value on the node2
25         STORE Child in Terminologies[j]
26         STORE Parent in Terminologies[j]
27         STORE Value in Terminologies[j]
28         INCREMENT j
29     END FOR
30     INCREMENT i
31  END FOR
32 END
```

Figura 23 - Algoritmo para Geração de Esquemas de Dados Documento

O algoritmo *DocumentGenerator* recebe como parâmetros de entrada os elementos extraídos do algoritmo *Extractor* (Figura 19) e, de posse destes elementos, cria um banco de dados (linha 5) e um conjunto de coleções para armazenar separadamente os atributos de dados, as terminologias e as restrições dos arquétipos, como mostra o código descrito entre as linhas 6 e 9 da Figura 23. Depois disso, passa-se então à inserção dos elementos dos arquétipos em suas respectivas coleções de documentos. O laço iniciado na linha 10 e finalizado na linha 31, percorre a lista de elementos dos arquétipos e adiciona na coleção de atributos (linha 13), o código, a descrição e o tipo de dado. Já para a coleção de restrições (linha 18), armazena-se o código de referência do atributo de dado e a descrição

da restrição. Por fim, caso o atributo de dado tenha uma ou mais terminologias, armazena-se na sua respectiva coleção (linha 21), o código de referência do atributo de dado, juntamente com a lista de terminologias encontradas.

O Algoritmo mostrado na Figura 24 é responsável pela geração de esquemas de dados do tipo família de colunas. Ele recebe como parâmetros de entrada os elementos extraídos dos arquétipos e inicia sua execução criando um banco de dados (i.e., keyspace) para agrupar as famílias de colunas do esquema de dados. Após a criação das famílias de colunas, realiza-se a inserção do código, da descrição e do tipo de dado na coleção de família de atributos (linha 13). Já a família de restrições (linha 18) armazena o código de referência do atributo de dado juntamente com a descrição da restrição. Por fim, armazena-se na família de terminologias (linha 27), o código de referência do atributo de dado e suas respectivas terminologias.

```
1  Algorithm 4 - Column Family Generator
2  Input: MetaData, Elements, DataStructures, Terminologies
3  Output: Void
4  BEGIN
5   INITIALIZE DataBase to Family KeySpace
6   CREATE Terminology as ColumnFamily
7   CREATE Constraints as ColumnFamily
8   CREATE Attributes  as ColumnFamily
9   SET i to 1
10  FOR each node on the DataStructures
11     SET Index GET index on the node
12     SET Description GET description on the node
13     SET DataType GET datatype on the node
14     SET Constraint GET constraint on the node
15     STORE Index in Attributes[i]
16     STORE Description in Attributes[i]
17     STORE DataType in Attributes[i]
18     STORE Constraint in Constraints[i]
19     SET Terms GET Terminologies on the node
20     SET j to 1
21     FOR each node2 on the Terms
22         SET Child GET child on the node2
23         SET Parent GET parent on the node2
24         SET Value GET value on the node2
25         STORE Child in Terminologies[j]
26         STORE Parent in Terminologies[j]
27         STORE Value in Terminologies[j]
28         INCREMENT j
29     END FOR
30     INCREMENT i
31  END FOR
32 END
```

Figura 24 - Algoritmo para Geração de Esquemas de dados Família de Colunas

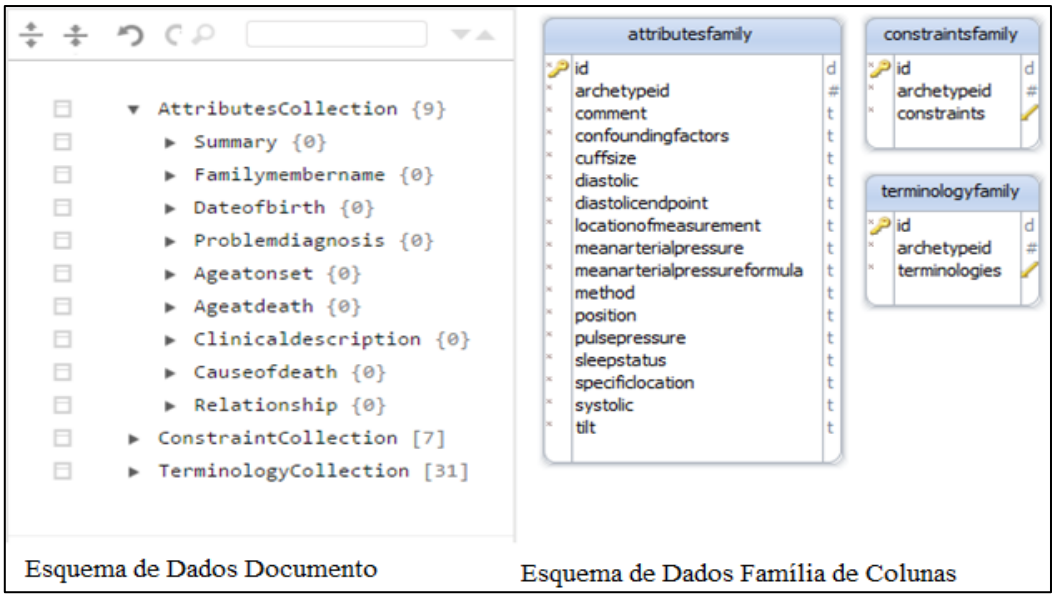


Figura 25 - Esquemas de dados NoSQL

A Figura 25 mostra dois esquemas de dados gerados a partir da execução dos algoritmos descritos nesta seção. O primeiro corresponde ao tipo documento gerado a partir do arquétipo *Apgar*, e o segundo, uma família de colunas gerado a partir do arquétipo *Family history*.

A estrutura de dados ITEM_TABLE organiza os atributos por meio de linhas e colunas. Para esse tipo, adotou-se um sistema de banco de dados orientado a grafo; onde as linhas são definidas como propriedades, as colunas correspondem aos vértices e a descrição do atributo representa o relacionamento entre eles.

4.4 GERAÇÃO DE INTERFACES GRÁFICAS ARQUETIPADAS

Esta seção descreve como as interfaces gráficas são geradas a partir dos atributos de dados, das terminologias e das restrições dos arquétipos e apresenta as funcionalidades de Template4EHR para customização de interfaces gráficas.

4.4.1 Serviço em Nuvem e Regras de Mapeamento

A geração de interfaces gráficas de usuário especificada neste trabalho, consiste em um serviço na nuvem que, a partir da leitura de um arquétipo, disponibiliza o artefato gerado

em um navegador na web [124, 125]. A Figura 26 mostra o funcionamento do serviço aqui proposto, e a definição dos principais componentes é apresentada a seguir.

A partir do algoritmo extrator (Figura 19) apresentado na Seção 3.3.1 deste documento, especificou-se um serviço na nuvem capaz de ler um arquétipo, extrair os atributos de dados, as terminologias e as restrições, e gerar as interfaces gráficas de usuário. A Figura 27 mostra o retorno dos dados em formato JSON, após a leitura de um arquétipo realizado pelo serviço em nuvem aqui proposto

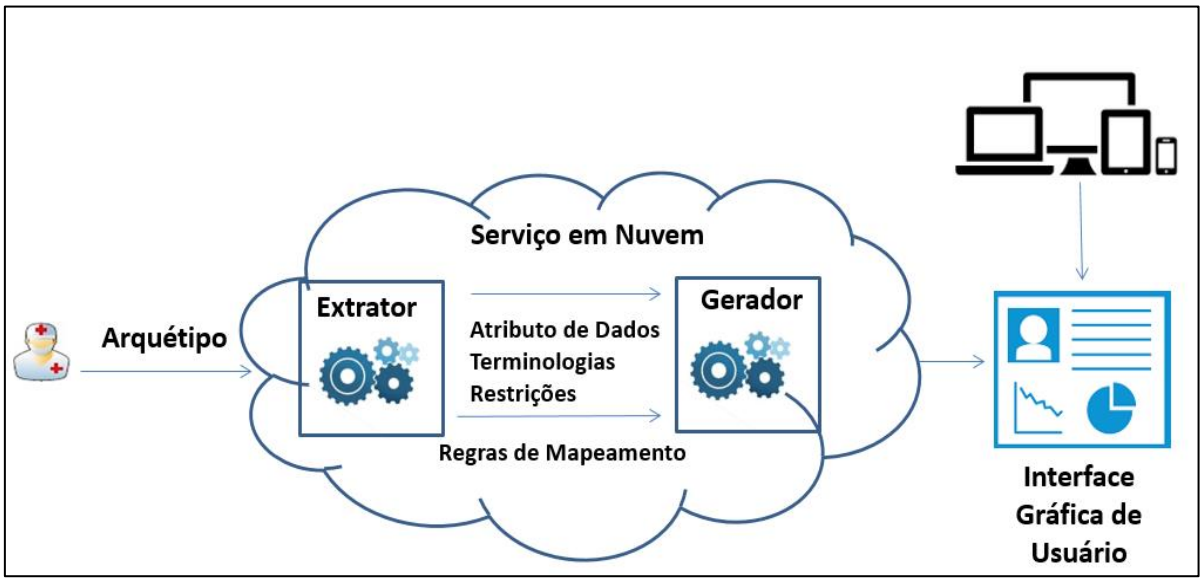


Figura 26 - Serviço em Nuvem para Geração de Interfaces Gráficas

Figura 27 - Exemplo de Elementos extraídos de um arquétipo em formato JSON

De posse dos elementos extraídos dos arquétipos, um conjunto de regras de mapeamento foram especificadas de modo a garantir a geração e a organização dos componentes das interfaces gráficas. As regras de mapeamento (RM) definidas nesse trabalho são:

- RM1: Toda interface gráfica é gerada a partir da leitura de um arquétipo XML/ADL ou de um template openEHR.
- RM2: Os atributos de dados extraídos dos arquétipos são mapeados em componentes da interface gráfica.
- RM3: As restrições especificadas em cada atributo de dado são utilizadas como mecanismo de validação de dados na interface gráfica.
- RM4: As terminologias existentes em cada atributo de dado são disponibilizadas na interface gráfica para dar significado semântico aos seus respectivos componentes.
- RM5: A disposição dos componentes da interface gráfica leva em consideração o tipo da estrutura de dados definida no arquétipo.
    - RM5.1 Para os tipos ITEM_SINGLE e ITEM_LIST, os componentes são organizados diretamente na interface gráfica, não havendo a necessidade de abas.
    - RM5.2 Para o tipo ITEM _TREE, os componentes são organizados na interface gráfica por meio de abas.
    - RM5.3 O nome da seção definida no arquétipo nomeará a(s) aba(s) criada(s) na interface gráfica.
    - RM5.4 Para o tipo ITEM_TABLE, a quantidade de colunas indica o número de abas que devem ser criadas na interface gráfica. O nome de cada aba seguirá a descrição dada no arquétipo.
- RM6: Os componentes da GUI serão criados de acordo com o tipo de dado encontrado no arquétipo e mostrado no Quadro 4.

Quadro 4 - Tipo de Componentes para geração na Interface Gráfica

Tipo de Dado do Arquétipo	Componente da Interface Gráfica
DV_TEXT	Caixa de Texto
DV_CODEDTEXT	Lista de seleção única
DV_BOOLEAN	Par de valores de seleção única
DV_COUNT	Caixa de Texto
DV_INTERVAL	Dupla caixa de Texto
DV_DATE_TIME	Caixa de Texto
DV_DATE	Caixa de texto
DV_TIME,	Caixa de Texto
DV_QUANTITY	Caixa de Texto
DV_DURATION	Caixa de Texto + Seleção única
DV_ORDINAL	Lista de seleção única
DV_URI	Caixa de texto

A execução das regras de mapeamento e a materialização das interfaces gráficas utilizando arquétipos é explicada na seção seguinte.

4.4.2 Algoritmo para Geração de Interface Gráfica

Os elementos extraídos dos arquétipos (Figura 19) são passados como parâmetros de entrada para o serviço de geração de interfaces gráficas, executado pelo Algoritmo 5 (Figura 28). O Algoritmo 5 cria uma interface gráfica vazia (i.e., linha 5) e atribui ao título do formulário o nome do arquétipo (linha 6). Para a geração dos componentes da interface gráfica, utiliza-se o mapeamento descrito no Quadro 4. Assim, os três vetores mostrados nas linhas 7, 8 e 9 do Algoritmo 5 definem o tipo do componente a ser criado para cada tipo de dado encontrado no arquétipo. Por exemplo, para DV_TEXT cria-se uma caixa de texto, enquanto um par de valores de única seleção é criado para DV_BOOLEAN.

O Algoritmo 5 também valida a informação que será manipulada em cada componente, por exemplo, o tipo DV_QUANTITY permite apenas valores inteiros, enquanto o tipo

DV_COUNT, valores decimais. O trecho de código entre as linhas 6 e 17, percorre os atributos de dados e, de acordo com o tipo encontrado, cria os respectivos componentes. Na sequência, valida-se as terminologias e as restrições associadas a cada atributo de dado. Para isso, o trecho de código entre as linhas 15 e 36 identifica e associa para cada atributo de dado, suas terminologias e restrições. Por fim, os componentes criados são adicionados na interface gráfica (linha 38), o laço é finalizado (linha 40), e a interface gráfica é mostrada para visualização (linha 41).

```
1  Algorithm 5 - GUI Generator
2  Input: MetaData, Elements, DataStructures, Terminologies
3  Output: Void
4  BEGIN
5   INITIALIZE Form to WebForm
6   SET the Form Title to Concept on the MetaData
7   CREATE TextBox as Collection
8   CREATE DropDownList as Collection
9   CREATE OptionBoolean as Collection
10  STORE {'DV_TEXT','DV_COUNT','DV_URI','DV_DATE_TIME','DV_QUANTITY',
11         'DV_DURATION'} in TextBox
12  STORE {'DV_CODEDTEXT','DV_ORDINAL'} in DropDownList
13  STORE {'DV_BOOLEAN'} in OptionBoolean
14  SET i to 1
15  FOR each node on the DataStructures
16     SET Index GET index on the node
17     SET Description GET description on the node
18     SET DataType GET datatype on the node
19     SET Constraints GET constraints on the node
20     INITIALIZE Object as Component
21     IF DataType contains in TextBox THEN
22         SET type to 'TextBox' in Object
23     ELSE IF DataType contains in DropDownList THEN
24         SET type to 'DropDownList' in Object
25     ELSE
26         SET type to 'Option' in Object
27     END
28     SET label to Description in Object
29     SET Terms GET Terminologies on the node
30     SET j to 1
31     FOR each terms on the Terms
32         IF code in node = code in term THEN
33             ADD terms in Object
34         END
35         INCREMENT j
36     END FOR
37     ADD Constraints in Object
38     ADD Object in Form
39     INCREMENT i
40  END FOR
41  SHOW Form
42 END
```

Figura 28 - Algoritmo de Geração da Interface Gráfica de Usuário

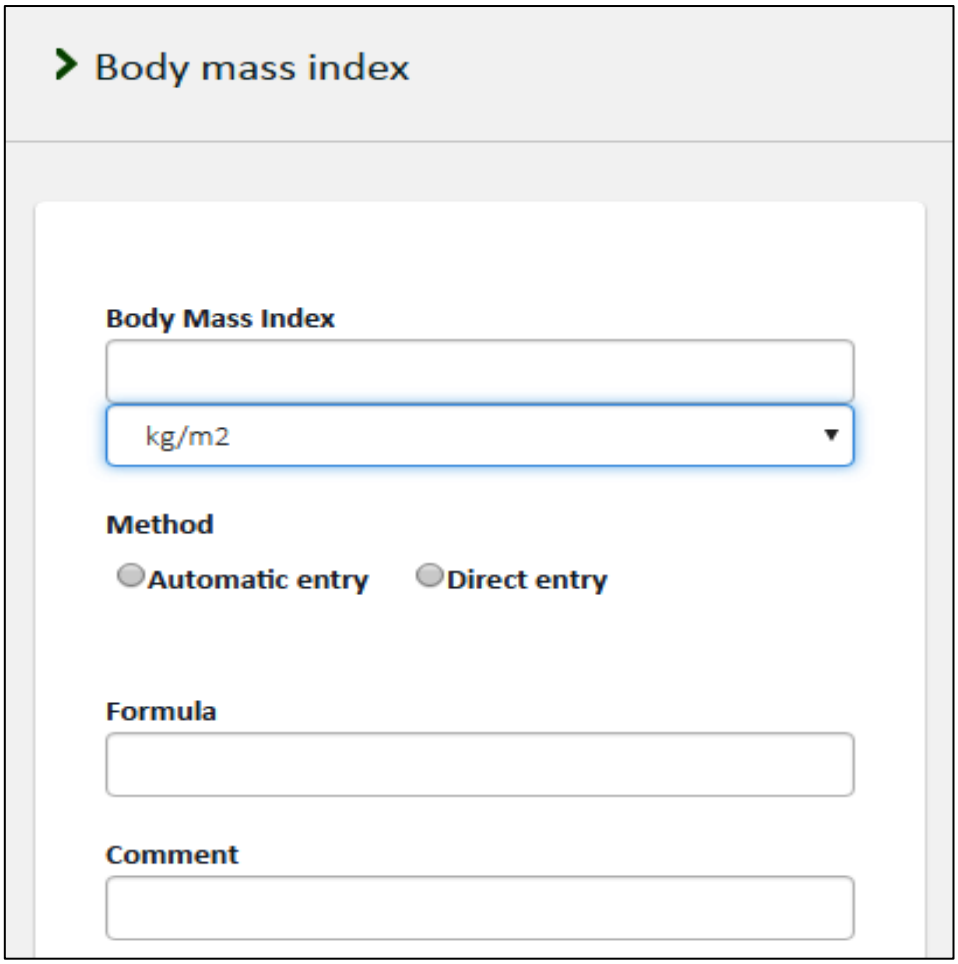


Figura 29 - Interface Gráfica gerada utilizando o Arquétipo Body Mass Index

A Figura 29 mostra uma interface criada por Template4EHR utilizando o arquétipo *Body mass index*. O exemplo da Figura 29 mostra que o serviço aqui proposto gerou o mesmo resultado da interface apresentada em [52].

4.4.3 Funcionalidades de Customização de Interfaces Gráficas

Template4EHR cria uma interface gráfica padrão a partir da leitura de um arquétipo ou template. No entanto, para melhorar a usabilidade e a interação do usuário, um conjunto de funcionalidades permitem a customização da interface gráfica. A Figura 30 mostra as principais funcionalidades de customização desenvolvidas e a definição de cada uma delas é dada a seguir.

- Escolher os atributos de dados: Esta funcionalidade permite que o usuário escolha quais atributos de dados do arquétipo estarão na interface gráfica.

- Configurar o tipo do formulário: Caso os atributos do arquétipo estejam organizados em uma estrutura hierárquica de vários níveis (i.e., ITEM_TREE), o usuário pode configurar as seções e os seus respectivos atributos em abas na interface gráfica.
- Ordenar os componentes da interface gráfica: Esta funcionalidade permite modificar a ordem de apresentação dos componentes da interface gráfica.

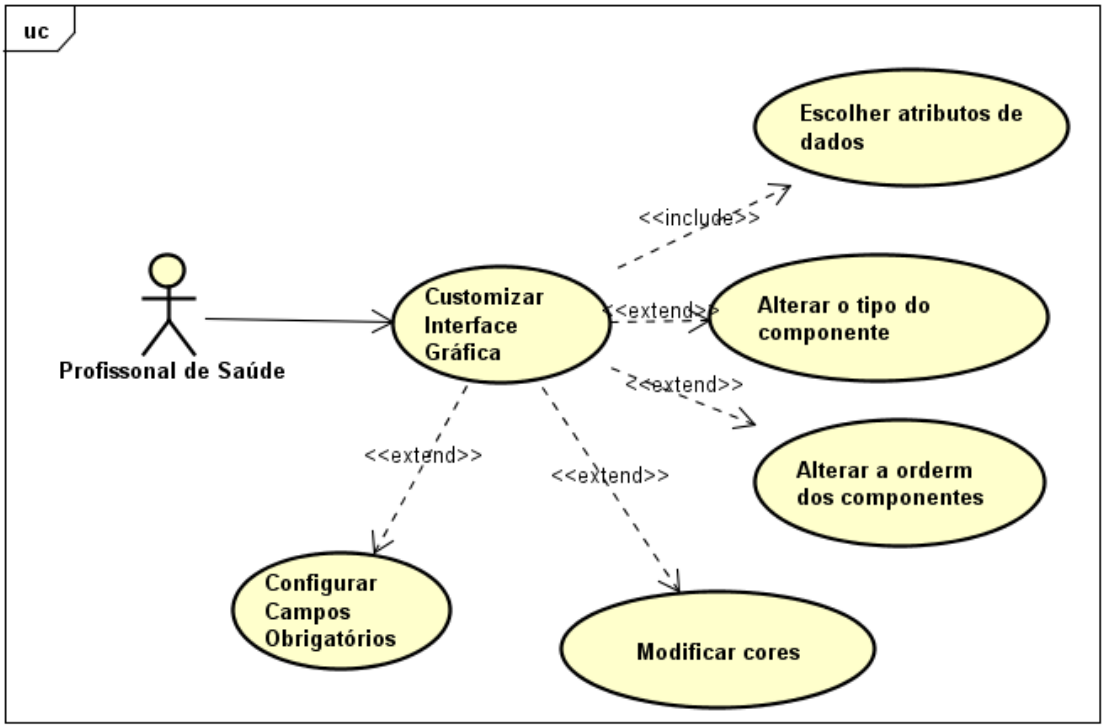


Figura 30 - Recursos de Customização da Interface Gráfica

- Modificar o tipo do componente da interface gráfica: De acordo com o tipo de dado, permite-se que o usuário escolha o tipo do componente que será apresentado na interface gráfica, por exemplo, um caixa de texto pode ser configurada como uma única linha ou múltiplas linhas.
- Modificar o comprimento e altura do componente da interface gráfica: Para cada componente da interface, permite-se alterar o valor do comprimento e da altura.
- Configurar campos obrigatórios: Configura os campos que serão de preenchimento obrigatório na interface gráfica.
- Configurar cores: Altera as cores de fundo da interface gráfica.

4.4.4 Paleta de Componentes Arquetipados

Para permitir que os componentes de interface gráfica propostos neste trabalho (Seção 3.4.1) sejam reutilizáveis e auxiliem no desenvolvimento de aplicações web, criou-se uma paleta de componentes baseada nos tipos de dados especificados pela openEHR (i.e., DV_TEXT, DV_CODEDTEXT, DV_BOOLEAN, DV_COUNT, DV_INTERVAL, DV_DATE_TIME, DV_DATE, DV_TIME, DV_QUANTITY, DV_DURATION, DV_ORDINAL, DV_URI).

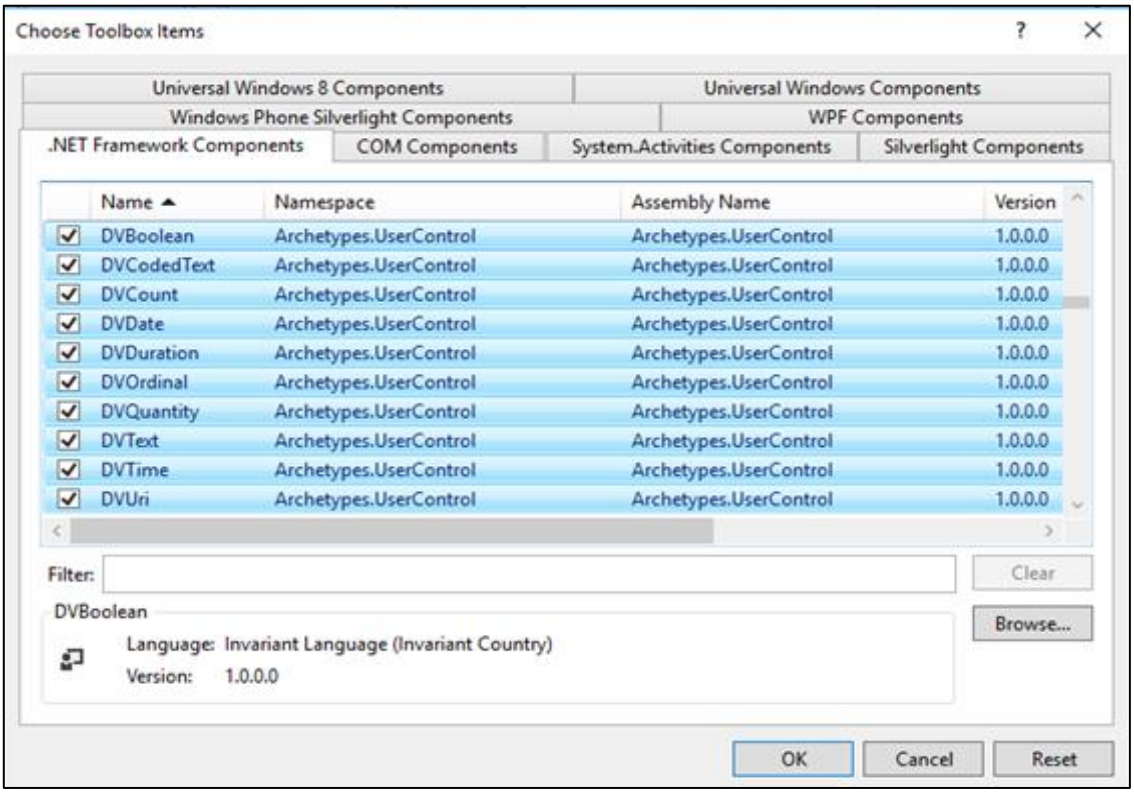


Figura 31 - Componentes Arquetipados

Chamados aqui de componentes arquetipados, esses componentes apoiam o desenvolvimento de projetos *ASP.Net* com suporte às linguagens de programação oferecidas no ambiente de programação do *Microsoft Visual Studio* (i.e., C#, Visual Basic, J#). Para utilizar este recurso, deve-se importar a *Dynamic link library* (*Dll)* que contém os componentes arquetipados (Figura 31), selecionar os itens desejados e confirmar a inclusão dos componentes no projeto da aplicação.

Cada componente contém um conjunto de propriedades, métodos e restrições que facilitam a codificação de uma aplicação utilizando arquétipos. Por exemplo, o *DVCount*

restringe a inserção de dados para conter apenas valores numéricos, enquanto que o *DVTime* contém máscaras que definem o formato correto de uma data. Além disso, é possível utilizar os métodos de manipulação de dados de cada componente. Para isso, deve-se realizar as seguintes tarefas: i) configurar a propriedade *DataSource* indicando um esquema de dados; ii) vincular o controle a um campo/tabela de um esquema de dados, e iii) acessar o ambiente de codificação e utilizar os métodos *add()*, *update()* e *delete()*.

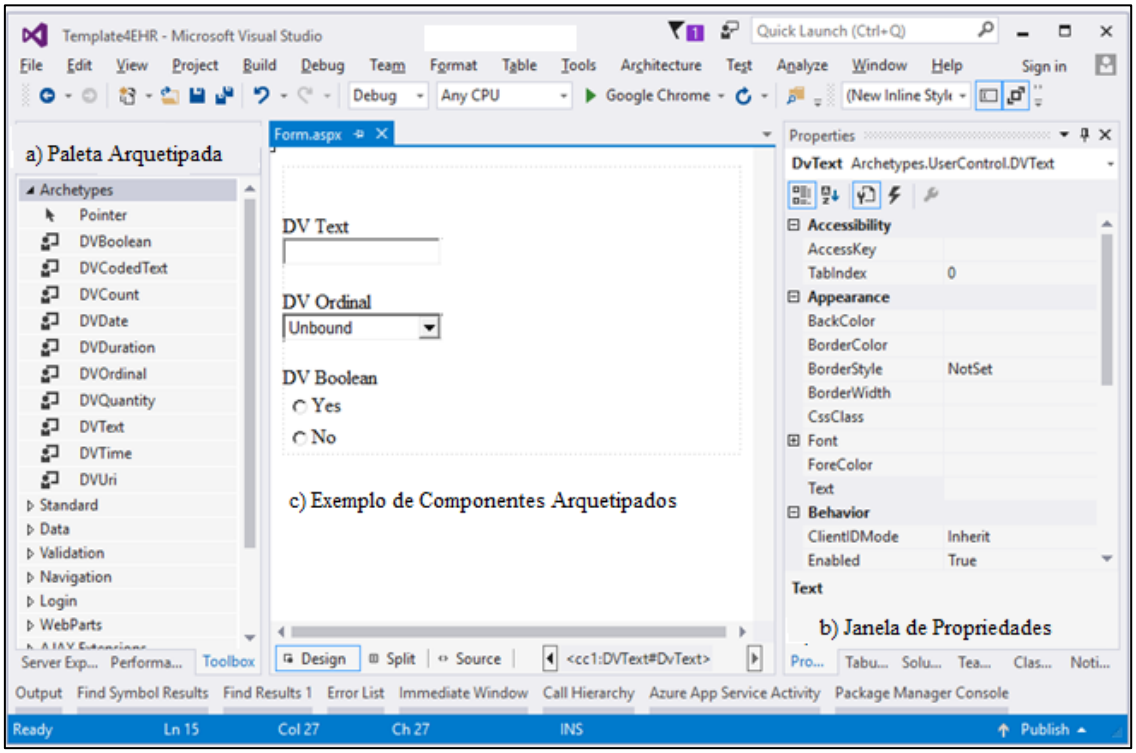


Figura 32 - Paleta de Componentes Arquetipados

A Figura 32 mostra a paleta de componentes arquetipados inserida no ambiente de desenvolvimento do *Microsoft Visual Studio*. Nela é possível observar que o programador dispõe de uma janela de propriedades e de um ambiente central no qual as interfaces gráficas são construídas arrastando os controles da paleta de componentes. No exemplo da Figura 32, tem-se três componentes arquetipados: i) uma caixa de texto para dados alfanuméricos (*DVText)*; ii) uma lista de valores suspensa (*DVOrdinal*) e iii) um par de valores de seleção única.

A vantagem principal no uso da paleta de componentes é que, como os controles já possuem as restrições, as propriedades e os métodos de persistência de dados encapsulados, o desenvolvimento de uma aplicação web utilizando arquétipos pode ser

otimizado por meio deste recurso. A *Dll* que contém os componentes apresentados nesta seção está disponível em http://template4ehr.azurewebsites.net.

4.5 GERANDO APLICAÇÕES MÓVEIS DE SAÚDE

Para permitir que os profissionais de saúde tenham mais agilidade no registro das informações clínicas do paciente, e de modo a reutilizar o serviço em nuvem apresentado na Seção 4.4.1 deste documento, foi desenvolvido um aplicativo móvel chamado de Mobile4EHR [126] que executa as instâncias de aplicações de saúde geradas por Template4EHR.

Figura 33 - Sincronizando a aplicação com Mobile4EHR

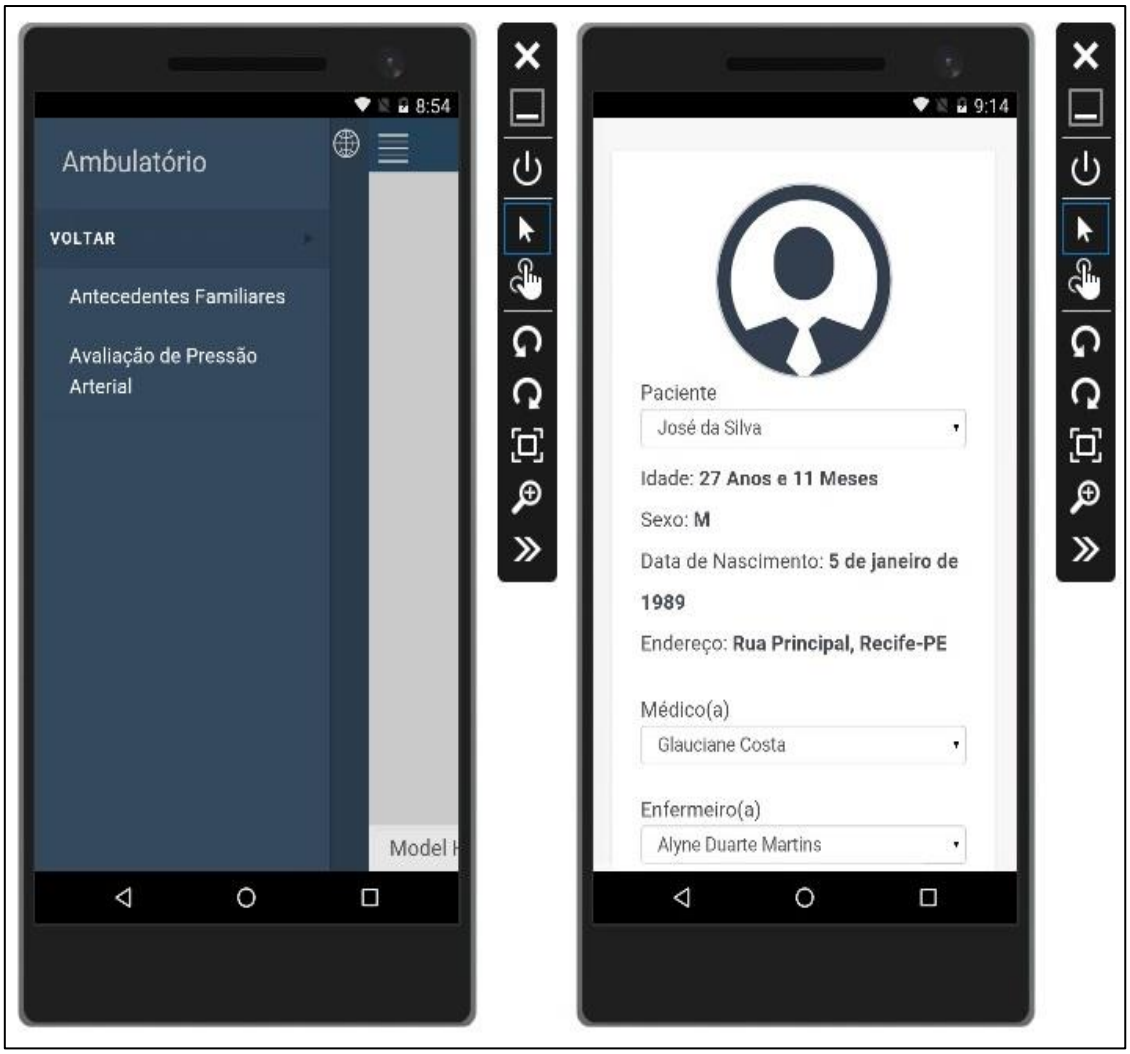


Figura 34 - Aplicação de Saúde executada em Mobile4EHR

A Figura 34 exibe uma aplicação gerada por Template4EHR e sendo executada em Mobile4EHR. Na aplicação aqui mostrada, os profissionais de saúde podem registrar os dados de antecedentes familiares e pressão arterial dos pacientes ambulatoriais de uma unidade de saúde.

4.6 EXEMPLIFICANDO A CRIAÇÃO DE UMA APLICAÇÃO

Nesta seção, demonstra-se a geração de interfaces gráficas de usuário e a criação de esquemas de dados em diferentes sistemas de bancos de dados. Para isso, utilizou-se os

arquétipos *Family History* e *Blood Pressure*, ambos mostrados na Figura 35 e disponíveis no repositório da openEHR[4].

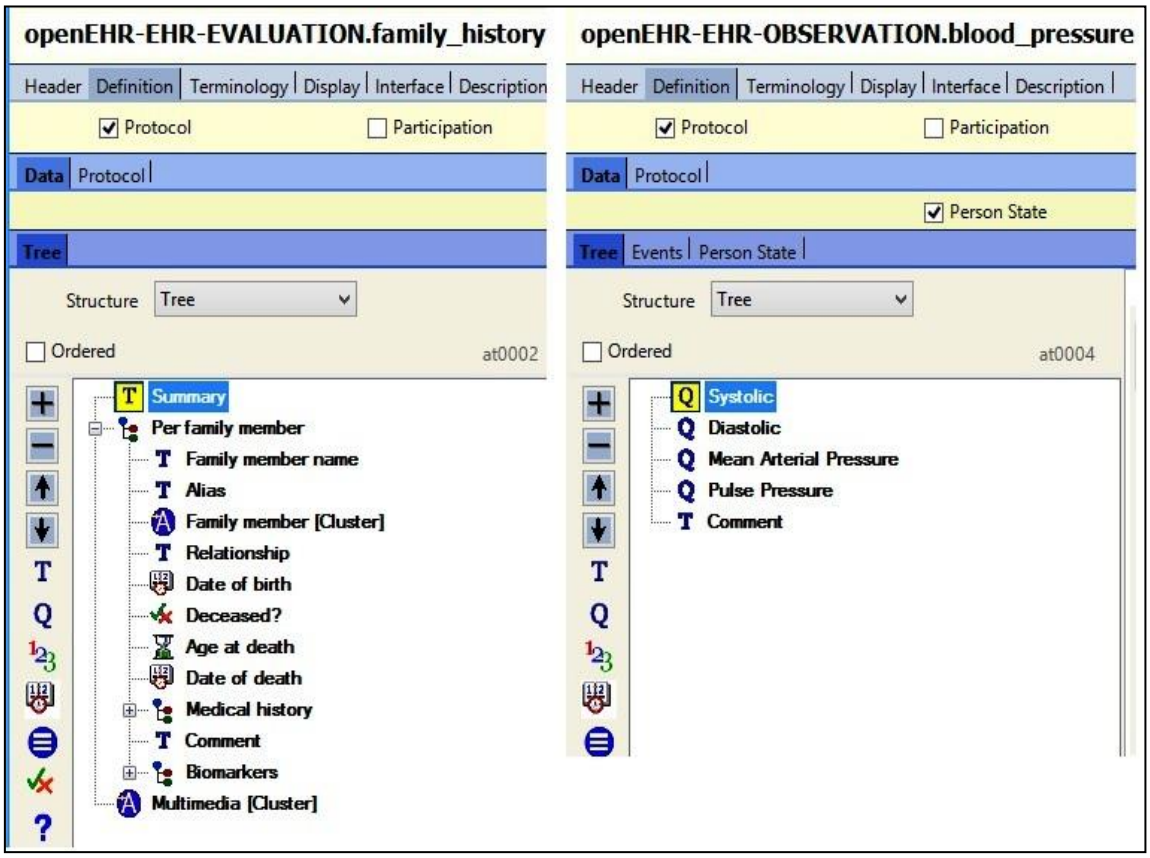


Figura 35 - Arquétipos Family History e Blood Pressure

Ao ser executado, Mobile4EHR busca e instala as interfaces gráficas geradas por Template4EHR no dispositivo móvel e cria um banco de dados local para o armazenamento do RES (Figura 33). Para a criação do banco de dados local, utiliza-se a geração de esquema de dados relacional descrita na Seção 4.3.1 deste documento. Mobile4EHR utiliza o recurso de armazenamento local quando não há conexão de rede móvel ou internet. Assim que Mobile4EHR identifica algum sinal de rede, os dados armazenados localmente são sincronizados e o armazenamento passa a ser no banco de dados criado por Template4EHR na nuvem. Neste caso, somente os registros que constam no banco local e não constam na nuvem são sincronizados.

[4] www.openehr.org/ckm/

Além disso, para cada alteração realizada nas interfaces gráficas ou esquemas de dados no serviço na nuvem, um alerta é gerado indicando que a aplicação móvel possui atualizações a ser feitas.

Para a criação da aplicação, cadastrou-se uma unidade de saúde chamada de Hospital Modelo e alguns pacientes, médicos, equipe de enfermagem e usuários da aplicação para manipulação de dados. Antes da importação dos arquétipos *Family History* e *Blood pressure*, criou-se um domínio chamado de *Ambulatório* e dois subdomínios chamados *Antecedentes familiares* e *Avaliação de pressão Arterial*, que posteriormente serão vinculados às interfaces gráficas criadas a partir dos respectivos arquétipos.

Depois disso, realizou-se a importação dos arquétipos utilizados na aplicação aqui descrita. Para a realização desta atividade, Template4EHR dispõe de uma funcionalidade que gerencia a escolha dos atributos de dados que farão parte da interface gráfica e do esquema de dados. A partir dessa funcionalidade é possível utilizar os recursos de customização das interfaces gráficas descritos na Seção 4.4.3 deste documento. A Figura 36 mostra a funcionalidade de customização e gerenciamento das interfaces gráficas gerada por Template4EHR

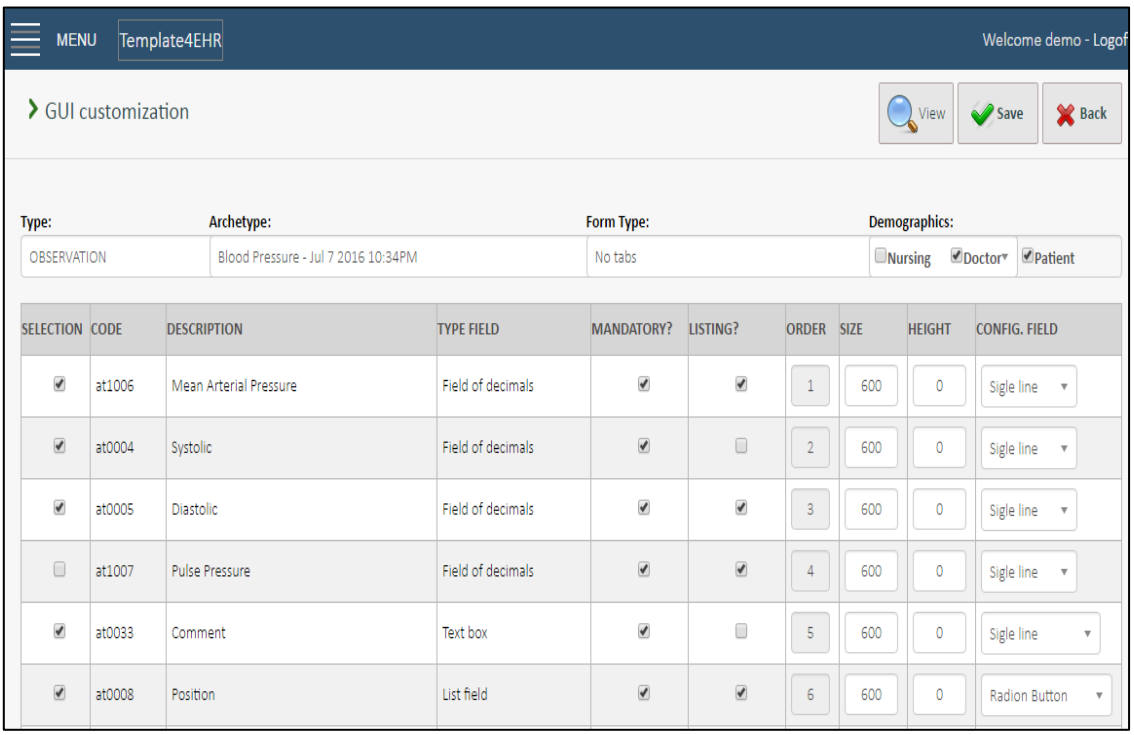


Figura 36 - Customização da Interface Gráfica de Usuário

Após a escolha dos atributos de dados, Template4EHR cria dinamicamente as interfaces gráficas e os esquemas de dados. Neste exemplo, configurou-se a criação de esquemas em diferentes sistemas de bancos de dados (i.e., relacional e NoSQL). Para disponibilizar a aplicação criada, associou-se a interface gráfica do arquétipo *Family History* com o subdomínio *Antecedentes familiares* e a interface gráfica do arquétipo *Blood Pressure* com o subdomínio *Avaliação de pressão Arterial.*

Neste exemplo, acrescentou-se à interface gráfica gerada a partir do arquétipo *Blood Pressure* as informações demográficas gerenciadas por Template4EHR, isto é, além dos atributos de dados escolhidos do arquétipo, a interface gráfica terá os seguintes componentes adicionais: o médico responsável pelos cuidados clínicos, o enfermeiro responsável pela aferição e o paciente em atendimento. A Figura 37 mostra a interface criada a partir da leitura do arquétipo *Blood Pressure*.

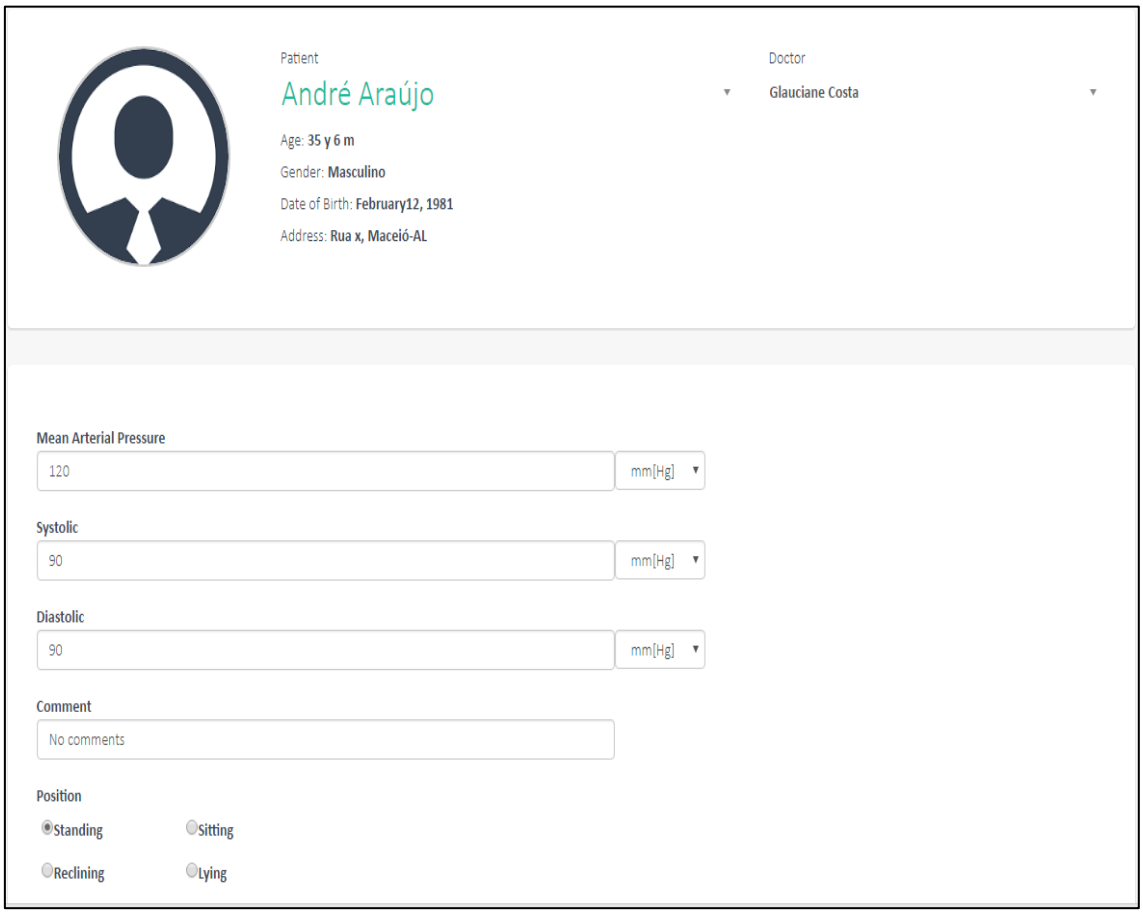


Figura 37 - Interface Gráfica para o Arquétipo Blood Pressure

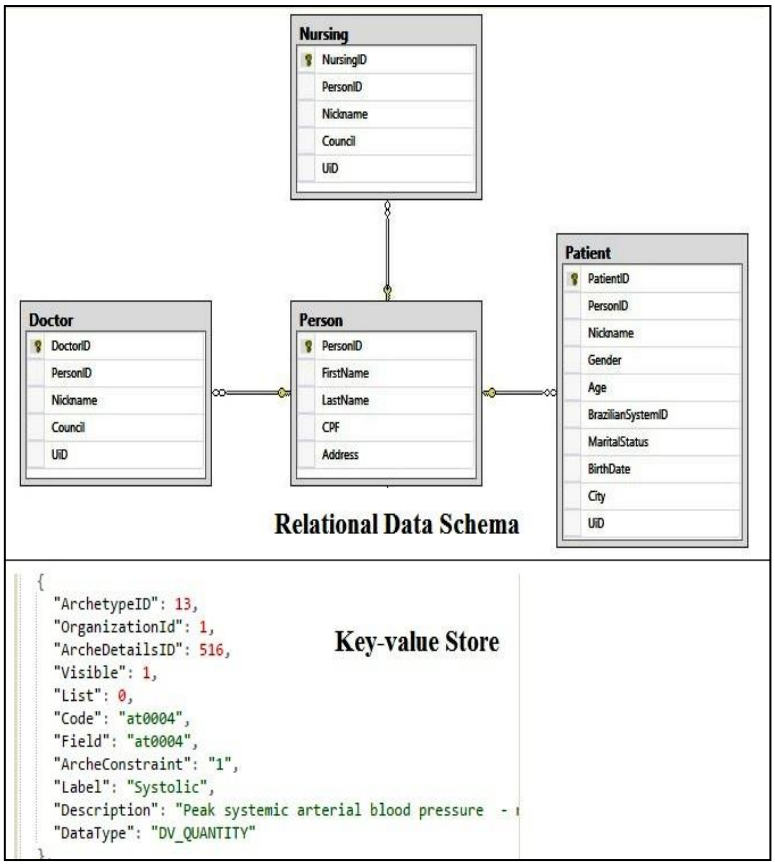


Figura 38 - Armazenamento do RES em Diferentes Sistemas de Banco de Dados

Como os atributos de dados do arquétipo *Blood pressure* foram especificados por meio de um ITEM_LIST, Template4EHR criou o esquema de dado em um banco do tipo chave-valor. Já as informações do paciente, médico e enfermagem foram armazenadas em um banco de dados relacional. A Figura 38 exibe partes do esquema de dados relacional e NoSQL utilizados neste exemplo para armazenar os dados inseridos na interface gráfica do arquétipo *Blood Pressure.*

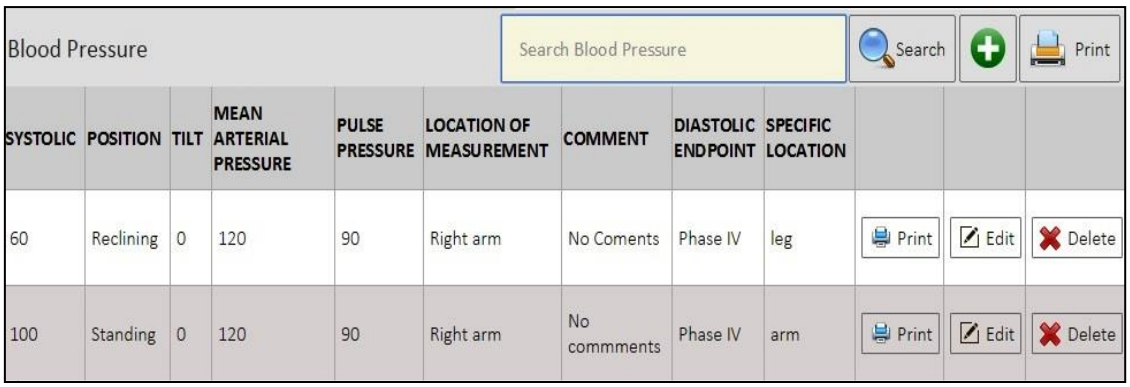


Figura 39 - Visualização dos dados armazenados

Como citado anteriormente, toda interface gráfica gerada por Template4EHR possui recursos de persistência de dados. Nela é possível, gravar, editar, consultar e excluir dados armazenados, como mostra a Figura 39. Nesse caso, Template4EHR faz uso de uma REST API de persistência que, independente do tipo do esquema adotado, realiza a manipulação dos dados.

4.7 LIMITAÇÕES DE TEMPLATE4EHR

Ressalta-se nesta seção as limitações de escopo de Template4EHR. Como descrito nas seções 4.3.1 e 4.3.2 deste documento, Template4EHR gera dinamicamente esquemas de dados relacionais e NoSQL a partir dos atributos de dados, das terminologias e das restrições de arquétipos. Quando novos atributos de dados são inseridos ou excluídos das interfaces gráficas de usuário, Template4EHR estende o esquema de dados criado por ele sem afetar o funcionamento da persistência de dados. No entanto, o framework aqui proposto não trata da evolução de esquemas de dados, isto é, não acrescenta novos objetos (e.g., tabelas, relacionamentos) ao esquema de dados criado, nem realiza a integração com sistemas de banco de dados legados. Além disso, o framework não gera interfaces gráficas de usuário utilizando-se do conceito de especialização ou agregação. Dessa forma, não é possível criar uma funcionalidade no qual uma interface gráfica tenha que relacionar dados de duas ou mais interfaces gráficas.

4.8 CONSIDERAÇÕES FINAIS

Esta seção apresentou o framework desenvolvido para a construção de esquemas de dados e interfaces gráficas de usuário utilizando arquétipos. Inicialmente, apresentou-se um metamodelo em UML que mostra como os conceitos da arquitetura dual e arquétipos modelam o RES. A arquitetura de Template4EHR foi apresentada e os principiais componentes desenvolvidos para geração de esquemas de dados relacionais, esquemas de dados NoSQL e interfaces gráficas de usuários foram discutidos. Por fim, a partir de arquétipos disponíveis no repositório da openEHR, mostrou-se como Template4EHR cria esquemas de dados e interfaces gráficas de usuário.

Na seção 5, apresenta-se um conjunto de estudos realizados para avaliar a arquitetura de Template4EHR. Os estudos envolvem a criação de uma aplicação com base em um cenário real de uma instituição de saúde e a avaliação das características de manutenibilidade, reusabilidade e extensibilidade por meio de métricas de software. Além disso, avalia-se a geração de esquemas de dados em diferentes sistemas de banco de dados e a usabilidade das interfaces gráficas por meio de um experimento controlado.

5 AVALIAÇÃO DA ARQUITETURA DE TEMPLATE4EHR

Esta seção apresenta e discute as avaliações da arquitetura de Template4EHR e está organizado da seguinte forma. A Seção 5.1 avalia o quanto de esforço Template4EHR pode reduzir para se construir uma aplicação de saúde. A Seção 5.2 investiga a conformidade de Template4EHR com as características de manutenibilidade, extensibilidade e reusabilidade, enquanto que a Seção 5.3 avalia a geração de esquemas de dados em diferentes sistemas de banco de dados. A Seção 5.4 apresenta um experimento realizado com profissionais de computação e de saúde para avaliar a criação e a usabilidade de interfaces gráficas geradas por Template4EHR, e por uma ferramenta correlata do mercado. A Seção 5.5 compara Template4EHR com outros trabalhos correlatos e a Seção 5.6 descreve a avaliação das principais funcionalidades do framework, que foi feita por profissionais de computação. A Seção 5.7 mostra como Template4EHR auxilia projetistas de banco de dados na geração de esquemas utilizando arquétipos. Por fim, relata-se as considerações finais na Seção 5.8.

5.1 DESENVOLVENDO UMA APLICAÇÃO COM TEMPLATE4EHR

Esta seção apresenta a avalição realizada para mensurar o quanto de esforço Template4EHR pode reduzir no desenvolvimento de uma aplicação em saúde. A Seção 5.1.1 descreve os objetivos da avaliação e o cenário utilizado, enquanto a Seção 5.1.2 discute os principais resultados obtidos.

5.1.1 Descrição do Cenário e Objetivos da Avaliação

Com base no modelo de avaliação de framework discutido na Seção 2.2.1 deste documento, investiga-se aqui dois pontos principais de Template4EHR: i) o custo de desenvolvimento de uma aplicação de saúde e ii) a redução de esforço na fase de codificação. Para isso, desenvolveu-se uma aplicação em saúde com base no fluxo de atendimentos à pacientes do Sistema único de Saúde (SUS) de um hospital localizado na região norte do país.

Para a realização de um atendimento hospitalar, registra-se inicialmente as informações cadastrais do paciente, tais como o número da carteira do SUS, nome, sexo,

data de nascimento, filiação, CPF e as formas de contato. Em seguida, vincula-se o médico responsável pelos cuidados do paciente juntamente com o tipo da acomodação (e.g., enfermaria, apartamento) e o procedimento de internação. O procedimento registrado no atendimento caracteriza o tipo da internação do paciente, podendo ser uma internação clínica ou cirúrgica.

Enquanto o paciente estiver nas acomodações do hospital, o médico responsável pelo paciente segue sua rotina diária de acompanhamento e cuidados clínicos. Esta rotina inclui a avaliação física do paciente, a prescrição de materiais e medicamentos a serem administrados pela equipe de enfermagem, a solicitação de exames laboratoriais e de imagem e, se necessário, a solicitação de um parecer médico a outro profissional especialista. A equipe de enfermagem é responsável por administrar toda a medicação descrita na prescrição médica e por acompanhar e registrar a evolução clínica do paciente diariamente, coletando os sinais vitais e o balanço hídrico. Ao realizar suas atividades de acompanhamento, o médico avalia os resultados dos exames juntamente com a evolução registrada pela equipe de enfermagem e, se necessário, solicita a realização de novos exames.

Quando o médico julga que o paciente já está em condições de deixar as dependências do hospital, o mesmo realiza a alta médica informando o CID (Código Internacional de Doenças), o motivo da alta (e.g., alta curada) e o procedimento que indica a melhora do paciente. Então, a equipe de enfermagem reúne todos os documentos vinculados a este atendimento (i.e., prescrições, checagem de enfermagem e exames), anexa-os ao prontuário do paciente e informa ao setor de faturamento que a alta hospitalar do paciente já pode ser realizada.

Para facilitar o entendimento sobre as funcionalidades que devem ser implementadas com o auxílio do framework, especificou-se um caso de uso (Figura 40) contendo os requisitos funcionais da aplicação, que são:

- R1: Manter paciente
- R2: Gerenciar prescrição médica
- R3: Manter internação hospitalar
- R4: Manter evolução do paciente
- R4.1: Registrar exame físico
- R4.2 Coletar sinais vitais

- R4.3 Coletar balanço hídrico
- R5. Manter alta do paciente

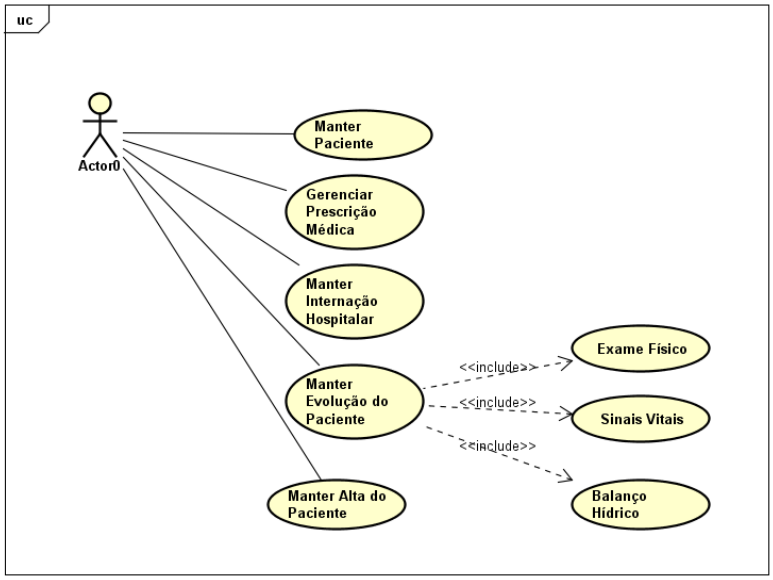


Figura 40 - Funcionalidades da Aplicação em Saúde

Neste cenário, dois tipos de atores fazem parte desta avalição. Primeiro, solicitou-se a dois profissionais de saúde que escolhessem no repositório da openEHR, arquétipos e seus respectivos templates que representassem os requisitos de dados das funcionalidades apresentadas na Figura 40. Assim, foram escolhidos os arquétipos *Admission*, *Family_history*, *blood_pressure*, *exam e respiration*. Três programadores com experiência em mais de 10 anos no setor de saúde foram convidados a construir a aplicação fazendo uso de Template4EHR e dos arquétipos/templates selecionados pelos profissionais de saúde. Antes de iniciar a avaliação, cada programador recebeu a documentação do framework contendo a descrição das funcionalidades, a definição da arquitetura de software com seus respectivos componentes e as instruções de geração dos artefatos de software. Para esta avaliação, as atividades de cada programador consistiram em:

- Instalar Template4EHR na estação de trabalho
- Instalar e configurar o SGDB para geração do esquema de dados
- Configurar o servidor web para execução da aplicação em um browser
- Codificar as funcionalidades da aplicação
- Responder o questionário de avaliação

5.1.2 Resultados da Avaliação

Esta seção discute os resultados coletados a partir da aplicação do questionário descrito na Seção 2.2.1 deste documento e apresenta a aplicação de saúde desenvolvida com o auxílio de Template4EHR. Após o término do desenvolvimento das funcionalidades solicitadas, cada programador avaliou o framework e os seus respectivos resultados estão sumarizados nas Tabelas 1, 2 e 3.

Nesse sentido, calculou-se inicialmente o custo total máximo, o custo de Template4EHR e o benefício do seu uso. Como todos os programadores utilizaram o mesmo cenário para o desenvolvimento da aplicação, o custo total máximo é o mesmo para as três avaliações, isto é, 95. Ressalta-se que o custo das categorias: sistema, projeto, aprendizagem, implementação e testes, é calculado multiplicando-se a quantidade de perguntas do questionário por 5. O custo de Template4EHR é resultante do somatório das respostas dadas pelo participante, enquanto que o benefício de uso do framework é a subtração do custo total máximo pelo custo do framework.

Tabela 1 - Resultados da Avaliação do Programador 1

	Custos					
Prog. 1	**Sistema**	**Projeto**	**Aprendizagem**	**Implementação**	**Testes**	**Total**
Custo Total	15	20	20	25	15	95
Custo do Framework	6	2	7	6	2	23
Benefício	9	18	13	14	8	62
Taxa de Redução	62/95 =	0,65				65%

Para o programador 1, o custo do framework foi igual a 23 e o benefício do seu uso corresponde a 62 (Tabela 1). Calculando a taxa de redução de custos do framework (i.e., 62/95), encontra-se o valor de 0,65, ou seja, 65% de esforços e custos foram reduzidos com o uso de Template4EHR para construção da aplicação aqui avaliada.

Os resultados da avaliação do programador 2 mostram que o custo do framework foi igual a 28, e o seu benefício de uso corresponde a 57 (Tabela 2). A taxa de redução de custos do framework foi de 0,6 (i.e., 57/95), o que representa 60% de esforços e custos reduzidos com o uso de Template4EHR.

Tabela 2 - Resultados da Avaliação do Programador 2

	Custos					
Prog. 2	**Sistema**	**Projeto**	**Aprendizagem**	**Implementação**	**Testes**	**Total**
Custo Total	15	20	20	25	15	95
Custo do Framework	7	1	10	8	2	28
Benefício	8	19	10	12	8	57
Taxa de Redução	57/95 =	0,6				60%

Por fim, os seguintes resultados foram obtidos na avaliação do programador 3. O custo do framework foi igual a 26 e o benefício do seu uso corresponde a 59. A taxa de redução de custos do framework equivale a 0,62, isto é, 62% de esforços e custos foram reduzidos com o uso de Template4EHR (Tabela 3).

Tabela 3 - Resultados da Avaliação do Programador 3

	Custos					
Prog. 3	**Sistema**	**Projeto**	**Aprendizagem**	**Implementação**	**Testes**	**Total**
Custo Total	15	20	20	25	15	95
Custo do Framework	5	4	9	7	1	26
Benefício	10	16	11	13	9	59
Taxa de Redução	59/95 =	0,62				62%

Analisando os resultados dos três participantes (i.e., 65%, 60% e 62%), nota-se uma redução média de 62,3% nos custos e esforços de desenvolvimento da aplicação de saúde aqui avaliada. Ao final da avaliação, além de responder ao questionário solicitado, os participantes puderam expor suas opiniões e sugestões de melhorias. A arquitetura de software especificada e os serviços de geração de esquemas de dados e interfaces gráfica de usuários foram destacados pelos programadores como pontos fortes de Template4EHR, enquanto a forma de disponibilização das interfaces gráficas para os usuários finais foi relatada como uma sugestão de melhoria.

As Figuras 41 e 42 mostram duas funcionalidades da aplicação desenvolvida nesta avaliação. O menu principal (Figura 41) foi desenvolvido por meio do recurso de criação de domínios e subdomínios (Seção 4.2.) de Template4EHR e permitiu disponibilizar as interfaces gráficas geradas para acesso do usuário final. A Figura 42 exibe a interface

gráfica gerada a partir da leitura do arquétipo *Admission* e que representa a funcionalidade de internação do paciente em uma unidade de saúde. A aplicação completa desenvolvida neste estudo está disponível em: http://template4ehr.azurewebsites.net/.

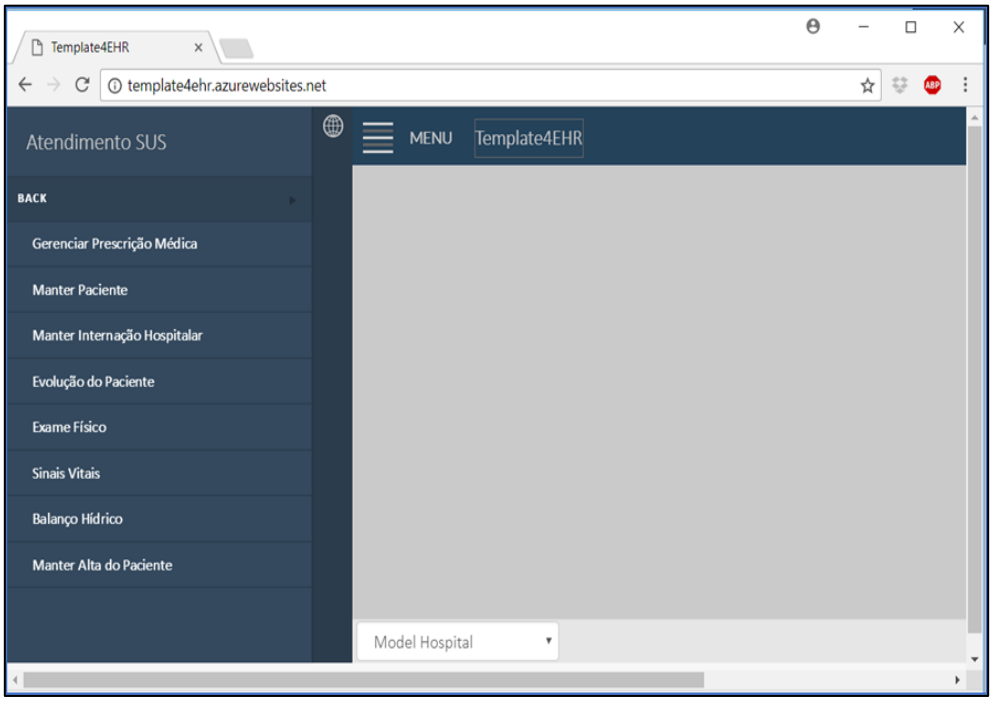


Figura 41 - Menu Principal da Aplicação

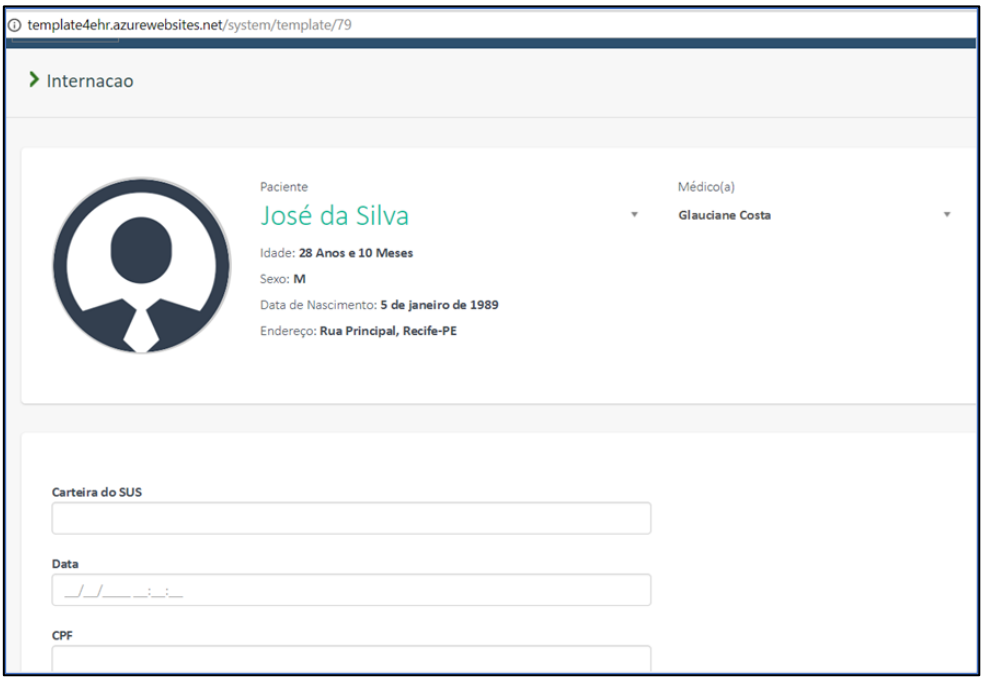


Figura 42 - Funcionalidade de Internação do Paciente

Os resultados apresentados neste estudo são restritos a um contexto local com poucos participantes e poucas funcionalidades avaliadas. Assim, os resultados aqui apresentados não podem ser generalizados. No entanto, ressalta-se a importante taxa de redução de codificação proporcionada por Template4EHR no desenvolvimento de uma aplicação de saúde. Para garantir que testes independentes sejam realizados por outros pesquisadores, uma versão sem fins comercias de Template4EHR está disponível em: https://doi.org/10.5281/zenodo.167623.

5.2 MÉTRICAS DE SOFTWARE PARA AVALIAÇÃO DA ARQUITETURA

A avaliação de uma arquitetura de software permite mensurar como estão as características de manutenibilidade, extensibilidade, e reusabilidade de um framework. Uma das formas de se investigar tais características é analisando, o padrão arquitetural utilizado, o potencial de reuso das classes, as interfaces e o mecanismo de persistência de dados.

Para avaliar a arquitetura de Template4EHR, utilizou-se o benchmark da Microsoft descrito na Seção 2.2.2 deste documento. Nesse sentido, investiga-se as seguintes métricas de software:

- Índice de manutenibilidade
- Complexidade ciclomática
- Profundidade de herança
- Acoplamento das classes
- Linhas de código implementadas

Neste estudo, Template4EHR foi configurado na plataforma de desenvolvimento do *Microsoft Azure* e a ferramenta que dá suporte ao benchmark foi executada. Após a análise de toda a estrutura e código fonte do framework, a ferramenta gerou um relatório (Figura 43) contendo os valores de cada métrica (i.e., índice de manutenibilidade, complexidade ciclomática, profundidade de herança, acoplamento das classes e linhas de código implementadas) agrupados pelas bibliotecas (i.e., *NameSpaces*) de Template4EHR.

Hierarchy	Maintainability Index	Cyclomatic Complexity	Depth of Inheritance	Class Coupling	Lines of Code
Template4EHR (Debug)	90	7.609	5	585	21.113
{} SysArcheER	69	97	5	80	826
{} SysArcheER.Forms	51	4.484	4	300	16.128
{} SysArcheER.Generico	89	1.312	2	133	1.483
{} SysArcheER.Generico.Nosql	93	121	1	19	131
{} SysArcheER.v101.ARCHETYPE	95	588	5	73	930
{} SysArcheER.v101.Composition	95	421	5	55	688
{} SysArcheER.v102.ARCHETYPE	95	586	5	73	927

Figura 43 - Métricas de software computadas para Template4EHR

Os valores apresentados na Figura 43 indicam que o framework possui um bom índice de manutenibilidade. Para o cálculo desta métrica, avalia-se a descentralização das camadas da arquitetura de software, a forma de comunicação entre as camadas (i.e., interfaces) e as classes implementadas na camada de negócio.

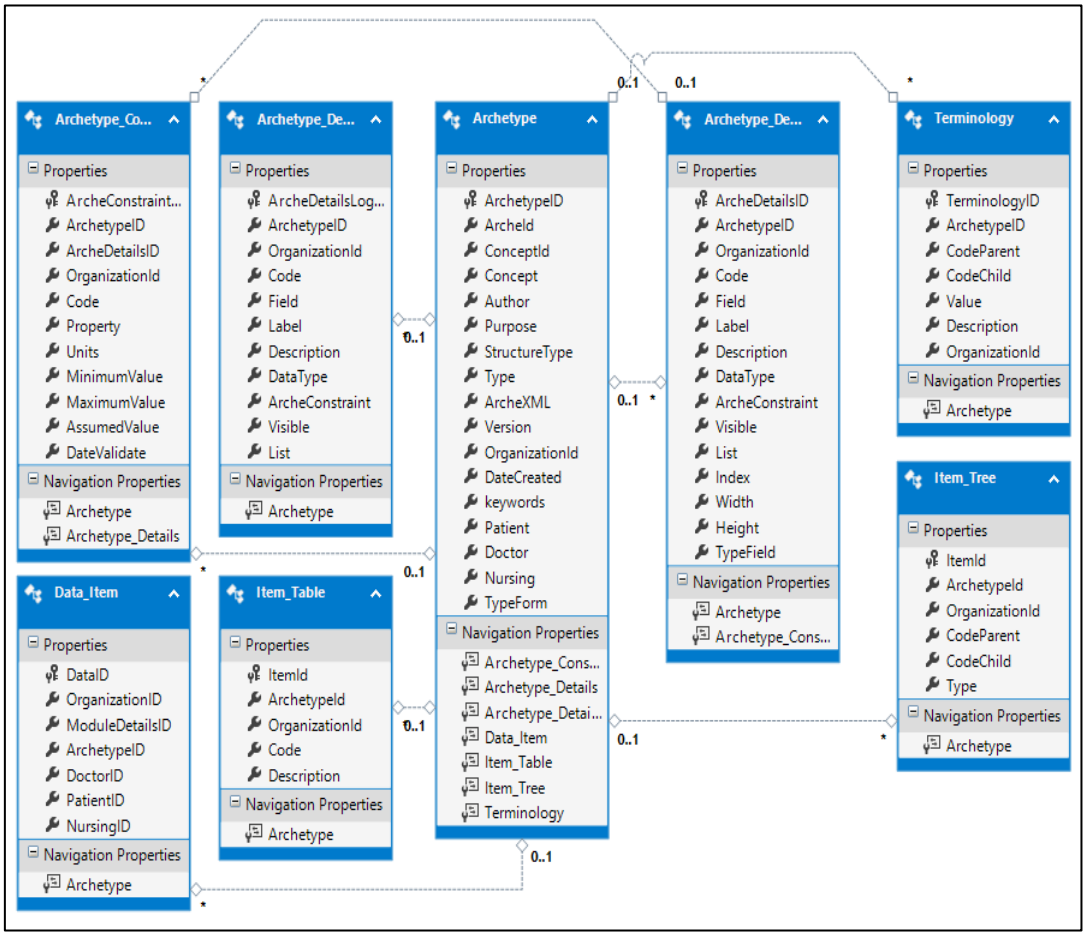


Figura 44 - Classes de Template4EHR

Conforme descrito na definição do benchmark *Microsoft Developer Network*, dispara-se um erro se o índice de manutenibilidade encontrado for inferior a 10, um aviso de melhoria se o índice estiver entre 10 e 20, e uma indicação de bom nível se o valor for maior que 20. Como mostra a Figura 43, todos os índices desta métrica estão acima de 51. Além disso, o índice de manutenibilidade é um indicador de qualidade das métricas de complexidade ciclomática, herança e acoplamento. A Figura 44 mostra parte das classes de Template4EHR que foram objeto de análise da métrica de manutenibilidade.

A complexidade ciclomática verifica a organização, a legibilidade das decisões lógicas e fluxos de controles identificados no código fonte de Template4EHR. Os valores encontrados para esta métrica ressaltam a complexidade de se gerar dinamicamente esquemas de dados e interfaces gráficas de usuário. A profundidade de herança indica o potencial de reuso das classes, enquanto o acoplamento expressa o nível de dependência entre as classes. Por fim, a última métrica exibe as 21.113 linhas de código que foram consideradas neste estudo.

Nesta avaliação, as métricas de software foram utilizadas para se investigar se existe algum componente de software que necessita de refatoração de código fonte ou ajuste para aumentar a coesão e baixar o acoplamento das classes do framework. Os resultados apresentados na Figura 44 apontam as boas práticas de desenvolvimento adotadas na codificação de Template4EHR.

5.3 AVALIAÇÃO DA GERAÇÃO DE ESQUEMA DE DADOS

De acordo com o *Software Engineering Institute* (SEI), a avaliação de uma arquitetura de software permite investigar se os atributos de qualidade de um sistema de software, como por exemplo, a usabilidade, a funcionalidade e a extensibilidade, estão em conformidade com o seu objetivo.

Um dos aspectos principais do framework proposto neste trabalho é a geração de esquemas de dados em diferentes sistemas de bancos de dados. Para isso, propõe-se uma arquitetura de software que possibilita a partir da leitura de arquétipos, a geração de esquemas de dados independente da tecnologia de um SGBD. Para avaliar tal característica, analisou-se a geração de esquemas de dados nas tecnologias de armazenamento descritas no Quadro 5.

Quadro 5 - Tecnologias de Armazenamento para Geração de Esquemas de Dados

Tecnologia de Armazenamento	**Paradigma de Banco de Dados**
SQL Server 2015	Relacional
Oracle DataBase 11g	Relacional
Cassandra 3.9	NOSQL – Família de Colunas
ArangoDB 3.1.12	NoSQL – Documento

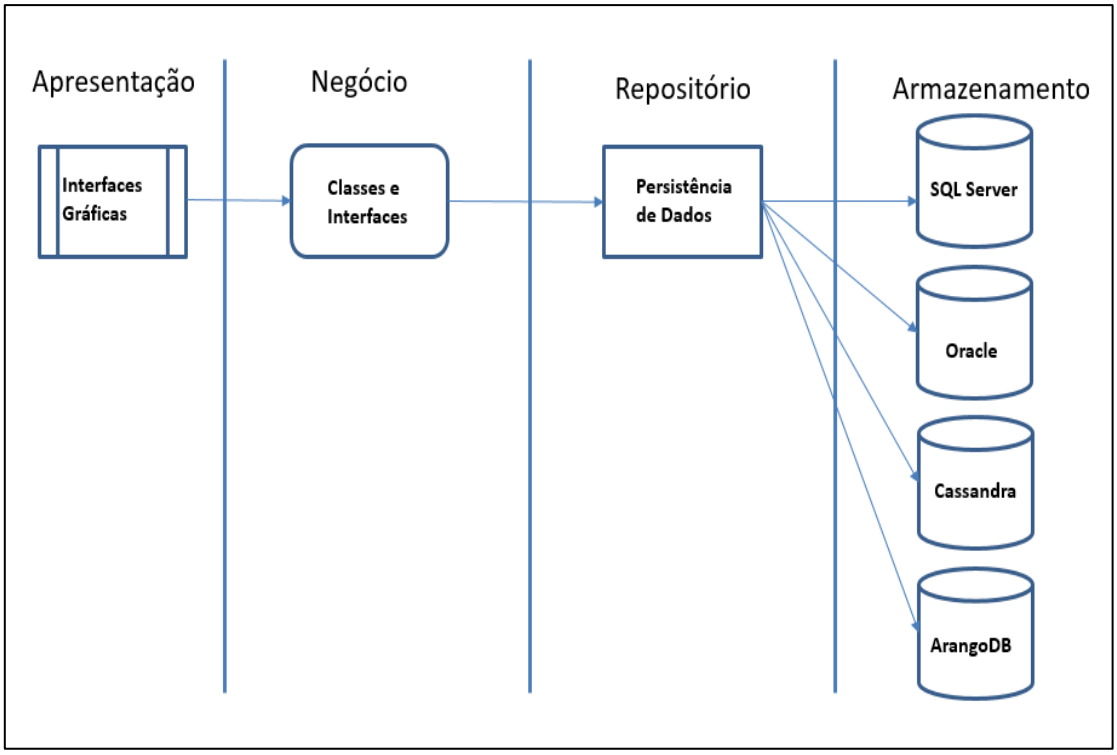


Figura 45 - Geração de esquemas de dados em diferentes SGBD

Como mostra a Figura 45, criou-se a partir da camada repositório, interfaces específicas para geração de esquemas de dados relacionais (i.e., *MS SQL Server* e *Oracle*) e NoSQL (i.e., *Cassandra* e *ArangoDB*). Para os esquemas de dados relacionais, investigou-se a geração das tabelas, dos campos e das restrições de integridades referenciais, enquanto para o esquema família de colunas, a organização dos atributos de dados, das terminologias e das restrições em suas respectivas colunas. Por fim, para o

esquema de dados orientado a documento, investigou-se a representação hierárquica dos elementos extraídos do arquétipo.

Nesta avaliação, utilizou-se o arquétipo (i.e., *Admin_Entry_Admission*) que contém os requisitos de dados da admissão de um paciente em uma unidade de saúde. O arquétipo está disponível no site da openEHR e foi escolhido por ter mais de 50 atributos de dados, 70 terminologias e 25 restrições. Além disso, os atributos de dados estão organizados por meio de uma estrutura hierárquica de diversos níveis.

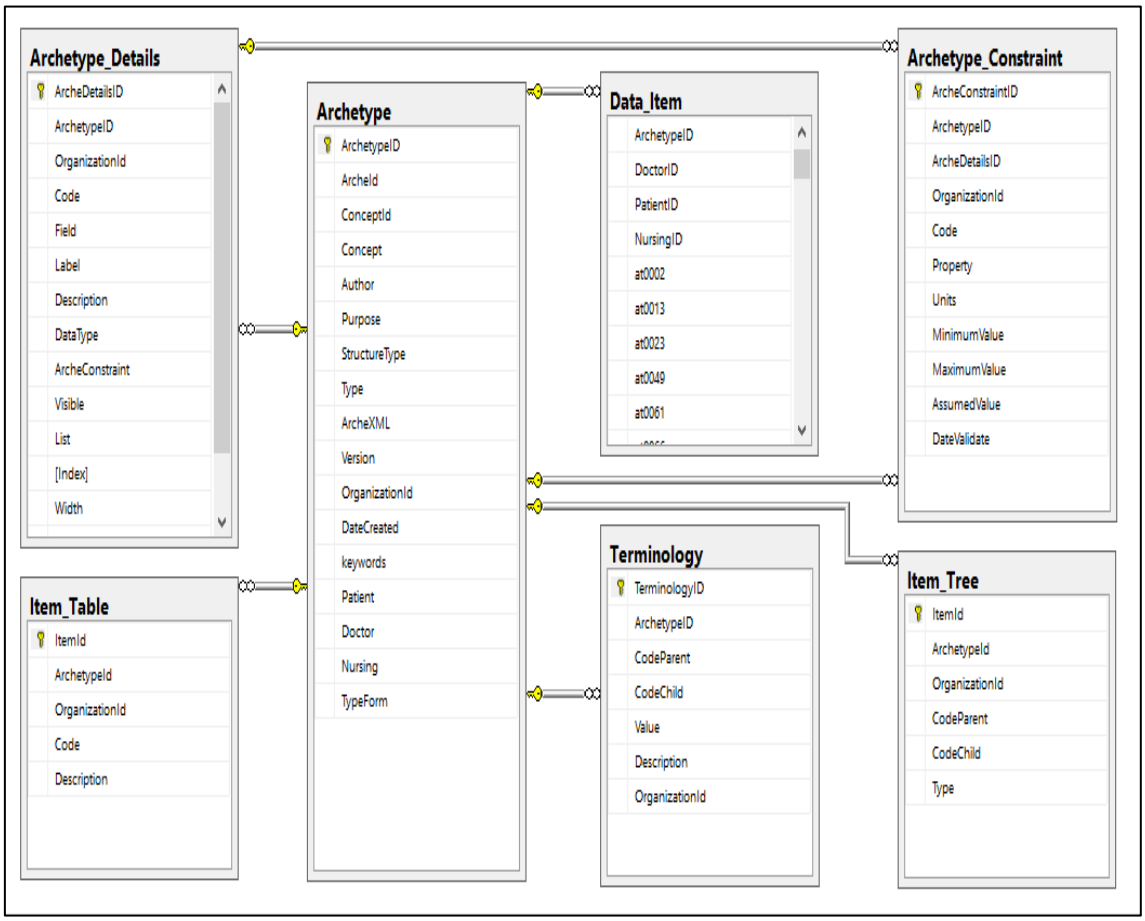


Figura 46 - Esquema de dados relacional gerado para o SQL Server

A partir da extração dos elementos do arquétipo, a camada de repositório recebe e organiza os atributos de dados, as terminologias e as restrições e permite que interfaces sejam desenvolvidas para a geração de esquemas de dados em diferentes sistemas de bancos de dados. Nesta avaliação, criou-se interfaces para a geração de esquemas de dados relacionais, família de colunas e esquemas orientados a documento. Uma das principais vantagens nesse modelo de arquitetura é que, caso seja necessário, novas

interfaces para outras tecnologias de SGBD podem ser desenvolvidas sem afetar o funcionamento das demais camadas da arquitetura de software. Para isso, basta mapear (*IMapper*) os elementos dos arquétipos em esquemas de dados relacionais ou NoSQL e utilizar as assinaturas dos métodos de persistência a serem implementadas em diferentes sistemas de bancos de dados (*IRepository*), conforme descrito na Seção 4.2 deste documento.

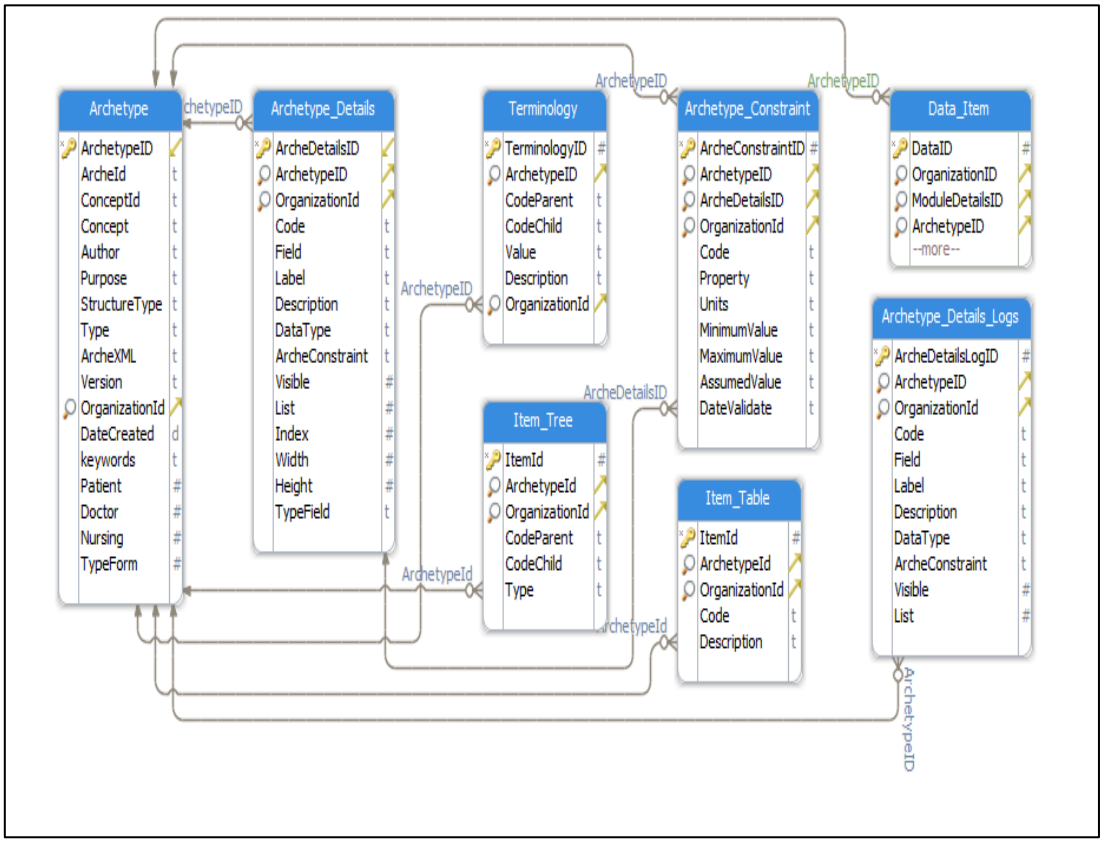


Figura 47 - Esquema de dados Relacional gerado para o Oracle

As Figura 46 e 47 mostram os dois esquemas de dados relacionais gerados para os SGBD *MS SQL Server* 2015 e *Oracle* 11g, respectivamente. Observa-se que as tabelas, os campos e as restrições de integridade referenciais foram geradas corretamente em dois SGDB de diferentes fabricantes. Nesse caso, foram realizados ajustes nos parâmetros de conexão de cada interface, e nos tipos de dados de cada SGDB, como por exemplo, o tipo de dado que armazena informações alfanuméricas; *VARCHAR*() para o *SQL Server* e o *VARCHAR2()* para o *Oracle*.

A Figura 48 mostra o esquema de dados gerado no formato de família de colunas gerado para o SGBD *Cassandra*. Nela é possível observar que os metadados que

caracterizam o arquétipo importado são armazenados em uma coluna chamada de *archetypes*, enquanto que os atributos de dados são armazenados na coluna *attributesfamily*. Por fim, as terminologias e as restrições do arquétipo são armazenadas nas colunas *terminologyfamily* e *constraintsfamily*, respectivamente.

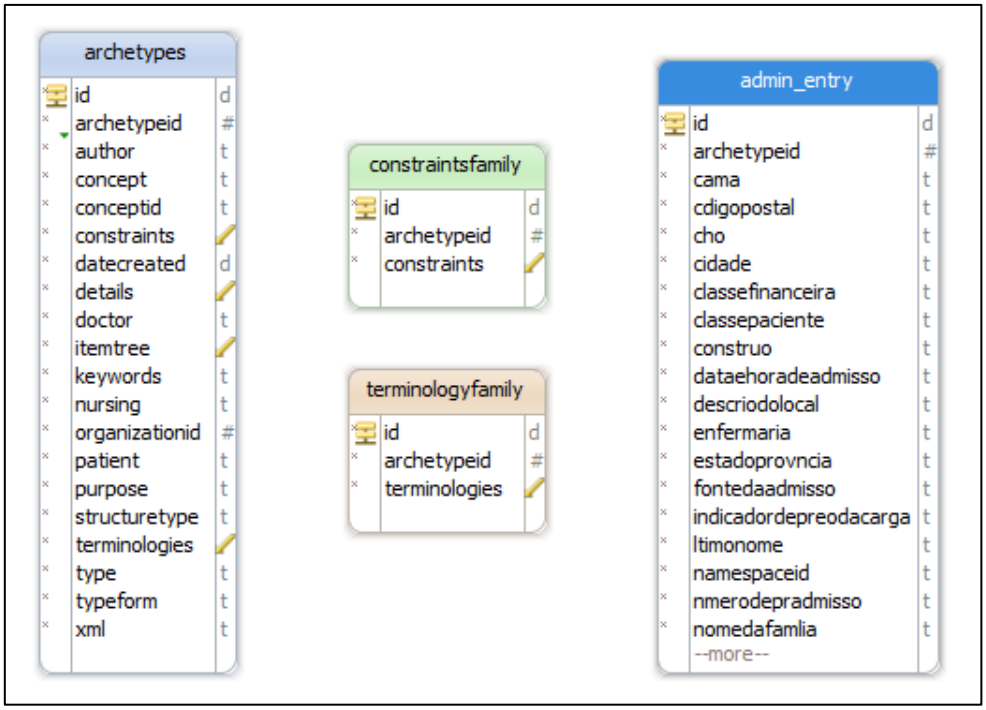


Figura 48 - Esquema de Dados gerado para o SGBD Cassandra

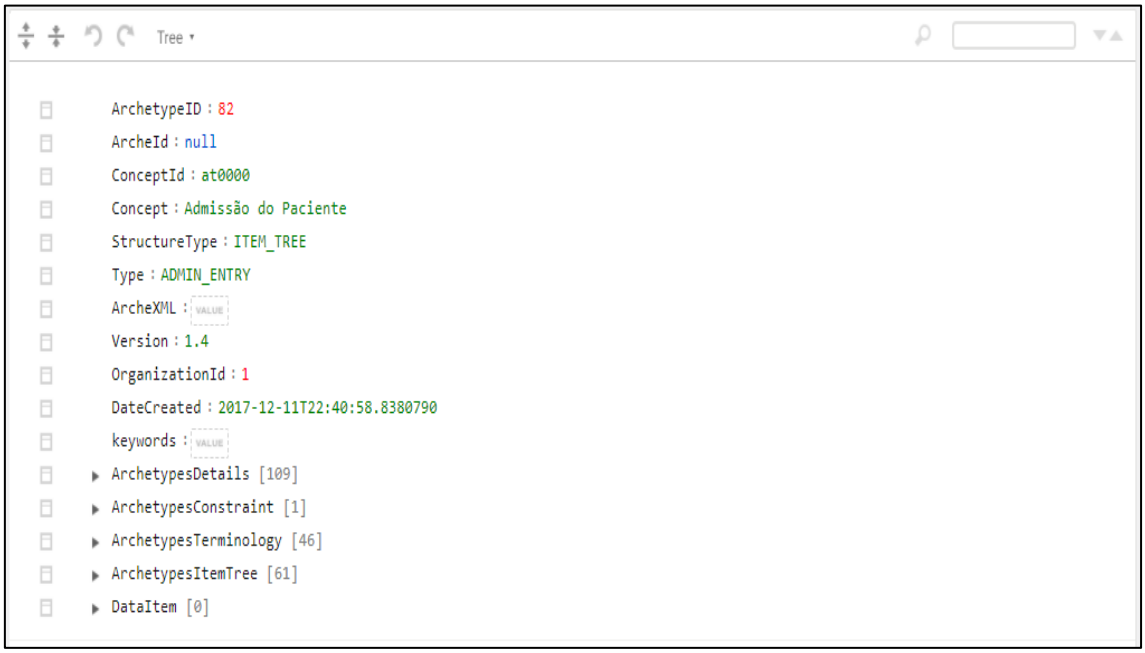


Figura 49 - Esquema de Dados Gerado para o SGBD ArangoDB

Esquemas de dados orientado a documento utilizam coleções para armazenar a estrutura hierárquica dos elementos de um arquétipo. Como mostra a Figura 49, armazena-se em uma coleção, os metadados do arquétipo, os atributos de dados com a mesma representação hierárquica, as terminologias e as restrições. O esquema de dados da Figura 49 foi gerado a partir da interface que gera documentos NoSQL para o SGBD *ArangoDB*. Outros SGBD que utilizam o mesmo sistema de banco de dados podem utilizar a mesma interface para a geração de esquemas de dados. Nesse caso, deve-se ajustar os parâmetros de conexão com o SGBD e alterar a sintaxe de criação das coleções de documentos.

As interfaces desenvolvidas para a geração dos esquemas de dados desta avaliação estão disponíveis em https://doi.org/10.5281/zenodo.167623.

5.4 EXPERIMENTO CONTROLADO

Esta Seção descreve um experimento controlado realizado com profissionais de computação e de saúde utilizando Template4EHR e EhRScape. EhRScape foi escolhido por três motivos. Primeiro, EhRScape auxilia no desenvolvimento de aplicações de saúde por meio de um conjunto de API baseadas nas especificações da openEHR. Segundo, EhRScape dispõe de um conjunto de funcionalidades para geração de interfaces gráficas, e por fim, trata-se de uma ferramenta de larga utilização no mercado internacional.

O estudo apresentado nesta seção baseia-se na experimentação descrita na Seção 2.3.3 deste documento e tem como objetivo avaliar a geração e a customização de interfaces gráficas de usuário por meio das ferramentas EhRScape e Template4EHR.

Para isso, realizou-se dois experimentos com diferentes profissionais. Primeiro, seis profissionais com experiência superior a dois anos na área de usabilidade foram recrutados para projetar interfaces gráficas utilizando arquétipos. Além da experiência na área de usabilidade, os seis voluntários recrutados nesse estudo possuem formação em computação, sendo um doutor, quatro mestres e um graduado. Em seguida, seis profissionais de saúde com experiência de mais de dois anos no uso de aplicações de saúde participaram de uma nova rodada do experimento. Destes, quatro são enfermeiros e dois são médicos. Para garantir a igualdade de condições para todos os participantes, o experimento ocorreu em um laboratório de informática com todos os computadores tendo

as mesmas configurações, isto é, um processador *intel* core i5, 16 GB de memória RAM, HD de 250 GB e browser Mozilla Firefox.

O objetivo desta avaliação não é determinar a melhor ferramenta para geração e customização de interfaces gráficas, mas sim entender as características de cada ferramenta, e avaliar o processo de construção de aplicações de saúde utilizando arquétipos. Nesse sentindo, investigou-se as seguintes questões:

Questão 1 (Q1): O tempo gasto na construção de interfaces gráficas de usuário utilizando EhRScape é maior do que em Template4EHR?

Questão 2 (Q2): Template4EHR oferece maior facilidade de geração de interfaces gráficas de usuário do que EhRScape?

Questão 3 (Q3): Template4EHR oferece mais recursos de customização de interfaces gráficas de usuário do que EhRScape?

Questão 4 (Q4): A usabilidade da interface gráfica gerada por Template4EHR é melhor que a de EhRScape?

A formulação das hipóteses deste experimento baseia-se em:

- Hipótese 1 (H1): O uso Template4EHR diminui o tempo gasto na construção de interfaces gráficas de usuários durante o ciclo de vida de desenvolvimento de uma aplicação em saúde.
- Hipótese 2 (H2): A geração dinâmica de interfaces gráficas de Template4EHR oferece facilidade aos usuários finais.
- Hipótese 3 (H3): Template4EHR permite que os usuários finais customizem as interfaces gráficas de uma aplicação de saúde de acordo com suas necessidades.
- Hipótese 4 (H4): Template4EHR cria interfaces gráficas de usuário de boa usabilidade para aplicações de saúde.

Para as quatro hipóteses apresentadas, as variáveis que serão consideradas neste experimento são:

- ✓ **Fator**
- F1 – construção de interfaces gráficas de usuário utilizando arquétipos
    - Nível do fator T1: Interface gráfica criada por EhRScape (F1→T1)
    - Nível do fator T2: Interface gráfica criada por Template4EHR (F1→T2)
- ✓ **Variáveis Independentes**

- P1: o guia de uso de cada ferramenta
- P2: os arquétipos *Family History* e *Blood Pressure* para a geração das interfaces gráficas
- P3: o roteiro de atividades a serem executadas em EhRScape e Template4EHR
- P4: o questionário de avaliação

✓ **Variáveis Dependentes**

- R1: tempo total utilizado para construir as interfaces gráficas de usuário
- R2: avaliação da facilidade de geração de interfaces gráficas
- R3: avaliação da customização de interfaces gráficas
- R4: avaliação da usabilidade das interfaces gráficas

Definidas as variáveis e hipóteses do experimento, formula-se então as alternativas e hipóteses nulas. Para facilitar o entendimento, cada métrica desta avaliação será representada por uma sigla. Cada métrica aferida terá variações provenientes das ferramentas EhRScape (**E**) e Template4EHR (**T**).

- TIG - Tempo médio de criação da interface gráfica
- FIG - Facilidade de geração da interface gráfica
- CIG - Recursos de customização da interface gráfica
- UIG - Usabilidade da interface gráfica criada

Hipótese Nula ($H0_1$): o tempo total gasto na construção de uma interface gráfica de usuário utilizando EhRScape é igual ao tempo gasto com Template4EHR.

$$\mathbf{H0_1: TIG_E \sim= TIG_T}$$

Hipótese Nula ($H0_2$): a facilidade de geração de interfaces gráficas de usuário é igual para as ferramentas EhRScape e Template4EHR.

$$\mathbf{H0_2: FIG_E \sim= FIG_T}$$

Hipótese Nula ($H0_3$): os recursos de customização de interfaces gráficas de usuário são iguais para as ferramentas EhRScape e Template4EHR.

$$\mathbf{H0_3: CIG_E \sim= CIG_T}$$

Hipótese Nula ($H0_3$): a usabilidade das interfaces gráficas de usuário é igual para as ferramentas EhRScape e Template4EHR.

$$\mathbf{H0_4: UIG_E \sim= UIG_T}$$

Hipótese Alternativa ($H1_1$): o tempo total gasto na construção de uma interface gráfica de usuário utilizando EhRScape é maior do que o tempo gasto utilizando Template4EHR.

$$\mathbf{H1_1: TIG_E \geq TIG_T}$$

Hipótese Alternativa ($H1_2$): a avaliação para a facilidade de geração de interfaces gráficas criadas por EhRScape é menor que a avaliação de Template4EHR.

$$\mathbf{H1_2: FIG_E \leq FIG_T}$$

Hipótese Alternativa ($H1_3$): a avaliação da customização de interfaces gráficas criadas por EhRScape é menor que a avaliação de Template4EHR.

$$\mathbf{H1_3: CIG_E \leq CIG_T}$$

Hipótese Alternativa ($H1_4$): a avaliação da usabilidade das interfaces gráficas de EhRScape é menor que a de Template4EHR.

$$\mathbf{H1_4: UIG_E \leq UIG_T}$$

Os sujeitos selecionados para participar deste estudo foram divididos em dois grupos de trabalho (i.e., G1 e G2), escolhidos aleatoriamente por meio de sorteio. Para eliminar a influência da experiência dos sujeitos selecionados nas variáveis de resposta do experimento e para garantir que a variação observada nos resultados seja oriunda apenas das variáveis dependentes descritas neste trabalho, foi adotado o delineamento do quadrado latino 2x2. Considerando que Exp1 e Exp2 correspondem aos objetos experimentais que serão atribuídos aleatoriamente por sorteio às variáveis, o delineamento do experimento é descrito no Quadro 6.

Quadro 6 - Delineamento do Experimento

	Quadrado Latino	
Grupo 1	(F1→T1)	(F1→T2)
Grupo 2	(F1→T2)	(F1→T1)

Conforme descrito no Quadro 6, quatro experimentos foram executados para eliminar a influência da variável indesejada, G1 e G2:

- Sujeitos do grupo G1 aplicam o tratamento F1→T1 no objeto experimental Exp1;
- Sujeitos do grupo G1 aplicam o tratamento F1→T2 no objeto experimental Exp2;
- Sujeitos do grupo G2 aplicam o tratamento F1→T2 no objeto experimental Exp1;
- Sujeitos do grupo G2 aplicam o tratamento F1→T1 no objeto experimental Exp2.

O grupo 1 iniciou a avaliação por EhrScape (F1→T1) e finalizou com Template4EHR (F1→T2), enquanto o grupo 2 iniciou com Template4EHR (F1→T2) e finalizou com EhrScape (F1→T1). Após a criação e execução das atividades de customização das interfaces gráficas, cada participante respondeu a um questionário com 68 questões para cada ferramenta investigada, indicando uma nota de 1 a 5 para cada item. Nesta avaliação, adotou-se a escala de resposta onde 5 representa muito satisfeito e 1 insatisfeito, seguindo a referência de construção de questionário descrita na Seção 2.3.3 deste documento.

Para calcular o tempo gasto por cada voluntário na realização do experimento, utilizou-se um software (i.e., *CamStudio*) que grava as ações e registra o tempo da atividade realizada. As outras métricas deste estudo (i.e., facilidade de geração, recursos de customização e usabilidade da interface gráfica) foram calculadas a partir das respostas do questionário (https://drive.google.com/open?id=10) construído para este experimento.

5.4.1 Avaliação com Profissionais de Computação

Após a consolidação dos dados, as Tabela 4 e 5 mostram os resultados estatísticos computados para os profissionais de computação. A partir da observação dos dados consolidados, pode-se constatar o seguinte.

O tempo médio gasto pelo grupo 1 para a realização das atividades solicitadas foi de 30 minutos com EhRScape, e 21 minutos com Template4EHR. Já para o grupo 2, gastou-se em média 26 minutos com EhRScape e 29 minutos com Template4EHR. Observando a média geral dos dois grupos, o tempo gasto com EhRScape foi superior a Template4EHR. No entanto, ao aplicar o teste estatístico *t student*, verificou-se que o valor de *P value* encontrado (i.e., 0,43) é maior que o nível de significância adotado de 0,05.

Tabela 4 - Resultados Estatísticos para o Grupo 1 de Computação

	Grupo 1	
Métricas	EhRScape	Template4EHR
Tempo médio (min)	30	21
Facilidade de geração da GUI	2,44	3,98
Recursos de customização	2,56	3,56

Usabilidade da Interface gráfica	3,29	3,98

Tabela 5 - Resultados Estatísticos para o Grupo 2 de Computação

	Grupo 2	
Métricas	EhRScape	Template4EHR
Tempo médio (min)	26	29
Facilidade de geração da GUI	3,33	3,90
Recursos de Customização	3,03	3,53
Usabilidade da Interface gráfica	3,67	4,07

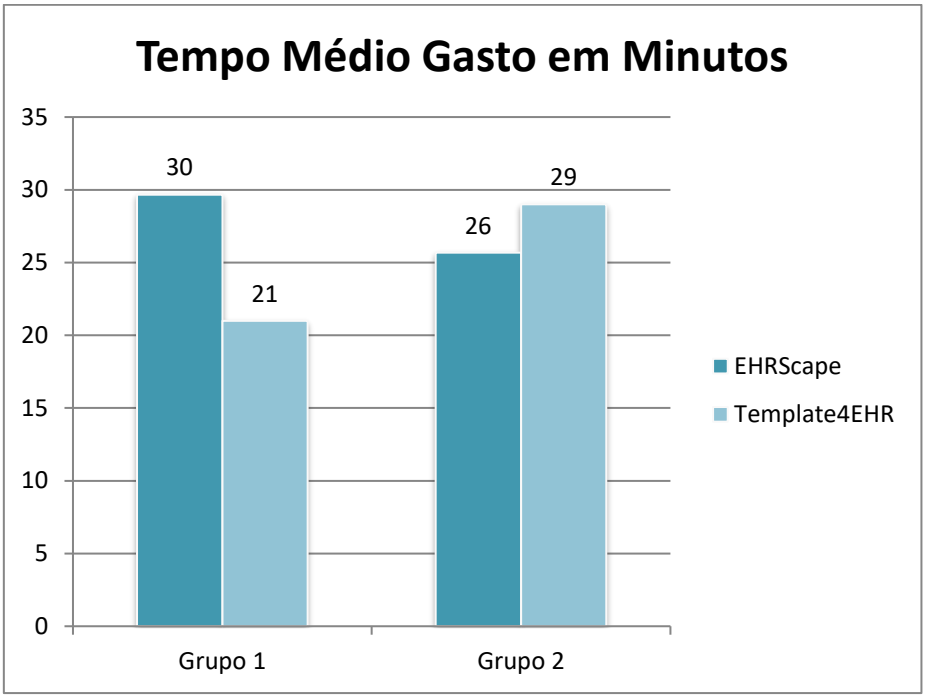


Figura 50 - Tempo Médio gasto para criar Interfaces Gráficas

Assim, conclui-se que, para o grupo de profissionais de computação deste estudo, não há diferença estatística significativa no tempo de construção de interfaces gráficas utilizando ambas ferramentas. Nesse caso, aceita-se a hipótese nula $H0_1$ que diz que o tempo total gasto na construção de uma interface gráfica de usuário utilizando EhRScape é igual ao tempo gasto com Template4EHR. A Figura 50 mostra o gráfico do tempo médio gasto pelos dois grupos de computação na realização desse estudo.

Os resultados obtidos para as métricas de facilidade de geração de interfaces e recursos de customização apontam que Template4EHR foi melhor avaliado pelos profissionais

participantes desse estudo. Numa escala de 1 a 5, foram calculados os seguintes resultados para facilidade de geração de interfaces gráficas: EhRScape obteve uma média de 2,44 para o grupo 1 e 3,33 para o grupo 2, enquanto que Template4EHR alcançou 3,98 de média para o grupo 1 e 3,90 para o grupo 2. Já para o recurso de customização de interfaces gráficas, EhRScape obteve média de 2,56 para o grupo 1 e 3,03 para o grupo 2, enquanto que Template4EHR atingiu 3,56 de média para o grupo 1 e 3,53 para o grupo 2. No entanto, realizando novamente o teste *t student*, verificou-se que o *P value* encontrado tanto para facilidade de geração de interfaces (i.e., 8,0337), quanto para customização (i.e., 1,512) são maiores que o nível de significância adotado de 0,05. Dessa forma, pode-se concluir que, não há diferença estatística significativa na geração e customização de interfaces com EhRScape e Template4EHR para os profissionais de computação deste estudo. Assim, as hipóteses nulas $H0_2$ (i.e., a facilidade de geração de interfaces gráficas de usuário é igual para as ferramentas EhRScape e Template4EHR) e $H0_3$ (i.e., os recursos de customização de interfaces gráficas de usuário são iguais para as ferramentas EhRScape e Template4EHR) são aceitas. A Figura 51 mostra os resultados da avaliação de customização de GUI que foi realizada pelos profissionais de computação em ambas ferramentas.

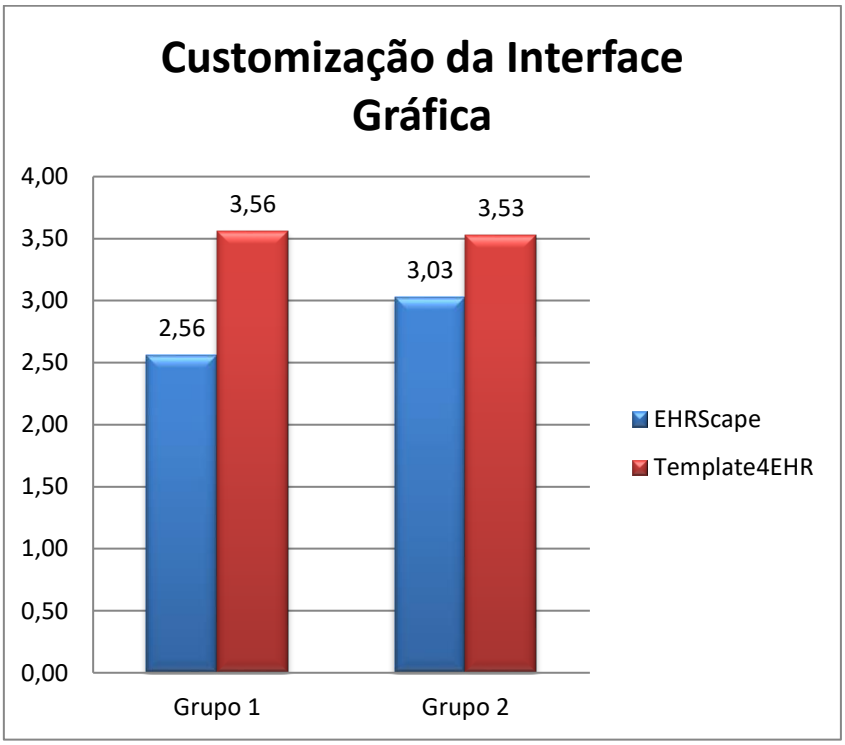


Figura 51 - Avaliação de Customização de Interfaces Gráficas

Por fim, avaliando a métrica de usabilidade das interfaces gráficas geradas, constatou-se o seguinte. EhRScape obteve 3,29 de média para o grupo 1 e 3,67 para o grupo 2, enquanto Template4EHR alcançou 3,98 de média para o grupo 1 e 4,07 para o grupo 2. Os resultados apresentados nas Tabelas 4 e 5 mostram que para os dois grupos, Template4EHR obteve uma melhor avaliação. Além disso, verificou-se no teste *t student* que, o valor de *P value* (i.e. 0,00016) é menor que 0,05, sendo assim, conclui-se que, para os profissionais desse estudo, as interfaces geradas por Template4EHR possuem melhor usabilidade. Nesse caso, rejeita-se a hipótese nula $H0_4$ e aceita-se a hipótese alternativa $H1_4$ que diz que a avaliação da usabilidade das interfaces gráficas de EhRScape é menor que a de Template4EHR. A Figura 52 exibe o gráfico com os dados computados para a usabilidade das interfaces gráficas.

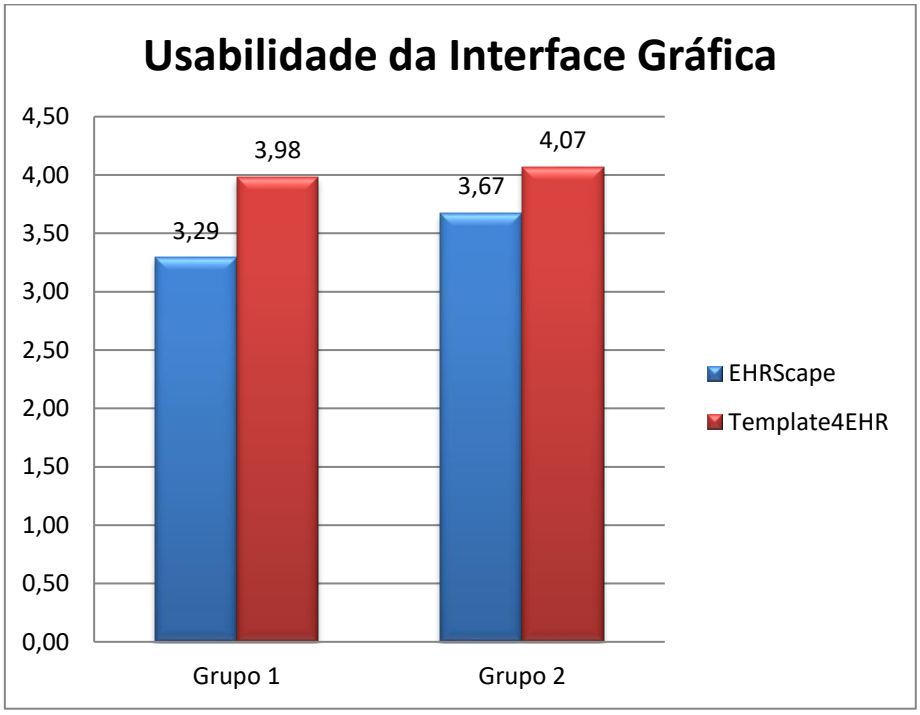


Figura 52 - Avaliação da Usabilidade das Interfaces Gráficas

Ao final da avaliação, solicitou-se que cada participante respondesse um questionário sobre a satisfação de uso dos frameworks aqui avaliados. A Figura 53 mostra que EhRScape obteve uma média de 68,68% de satisfação, enquanto que Template4EHR produziu um valor de satisfação média igual a 81,14%.

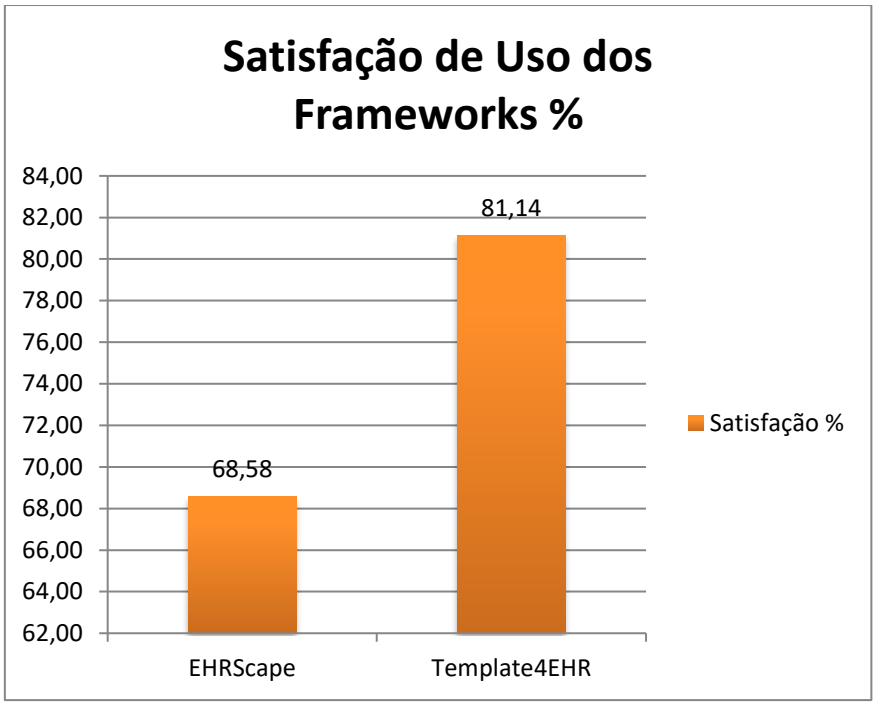


Figura 53 - Satisfação de Uso dos Frameworks

5.4.2 Avaliação com Profissionais de Saúde

Utilizando a mesma abordagem do primeiro experimento, consolidou-se os dados das atividades realizadas pelos profissionais de saúde e os resultados são apresentados nas Tabelas 6 e 7.

Tabela 6 - Resultados Estatísticos para o Grupo 1 de Saúde

	Grupo 1	
Métricas	EhRScape	Template4EHR
Tempo médio (min)	45	40
Facilidade de geração da GUI	2,26	3,16
Recursos de customização	3	3,4
Usabilidade da interface gráfica	3	3,07

O tempo médio gasto pelo grupo 1 foi de 45 minutos com EhRScape, e 40 minutos com Template4EHR. Já para o grupo 2, gastou-se um tempo médio de 35 minutos com EhRScape e 42 minutos com Template4EHR. Para a métrica de facilidade de geração de interfaces gráficas, EhRScape obteve uma média de 2,26 para o grupo 1 e 2,31 para o

grupo 2, enquanto que Template4EHR alcançou uma média de 3,16 para o grupo 1 e 4,1 para o grupo 2. Executando o teste estatístico *t student* novamente, constatou-se que o valor de *P value* encontrado para as métricas de tempo médio gasto (i.e., 0,73) e facilidade de geração de interfaces (i.e., 2,82) são maiores que o nível de significância de 0.05. Isto quer dizer que, para os profissionais de saúde deste experimento, não existe diferença estatística significativa entre EhRScape e Template4EHR para essas duas métricas. Nesse caso, aceita-se as hipóteses nulas $H0_{1}$ e $H0_{2}$.

Tabela 7 - Resultados Estatísticos para o Grupo 2 de Saúde

	Grupo 2	
Métricas	EhRScape	Template4EHR
Tempo médio (min)	35	42
Facilidade de geração da GUI	2,31	4,1
Recursos de customização	2,07	4
Usabilidade da interface gráfica	3,08	4,43

Os resultados calculados para a métrica de customização da interface gráfica mostram que EhRScape obteve uma média de 3,0 para o grupo 1 e 2,07 para o grupo 2. Já Template4EHR alcançou uma média de 3,4 para o grupo 1 e 4,0 para o grupo 2. Por fim, para a métrica de usabilidade, EhRScape obteve uma média de 3,0 para o grupo 1 e 3,08 para o grupo 2, enquanto que Template4EHR alcançou uma média de 3,07 para o grupo 1 e 4,43 para o grupo 2.

Executando o teste estatístico *t student*, constatou-se que os valores de *P value* encontrado para as métricas de customização (i.e., 0,00019) e usabilidade (i.e., 0,00084) são menores que 0.05.

Portanto, conclui-se que, para os profissionais de saúde que participaram desse estudo, Template4EHR oferece melhores recursos de customização e usabilidade de interfaces gráficas geradas utilizando arquétipos. Assim, rejeita-se as hipóteses nulas $H0_{3}$ e $H0_{4}$, e aceita-se as hipóteses alternativas, $H1_{3}$ e $H1_{4}$. Além disso, para os profissionais de saúde, Template4EHR alcançou uma satisfação média de uso igual a 81,3%, enquanto que EhRScape obteve uma satisfação média de 61%.

5.4.3 Ameaças à Validade do Experimento

Duas ameaças à validade do experimento foram identificadas neste estudo. A primeira, chamada de testagem, proporciona que os participantes aprendam com seus próprios erros. Como o experimento consistia na realização de uma atividade longa, é possível que no decorrer do experimento os participantes tenham aprendido com os seus erros, o que pode afetar a avaliação dos frameworks investigados. A segunda diz respeito ao comportamento competitivo do ser humano. Como os participantes eram de áreas distintas de formação (i.e., computação e saúde), é possível que alguns membros de um determinado grupo de controle tenham se sentido preteridos frente aos do outro grupo. Nesse caso, este fator pode estimular a competição entre os participantes e afetar também a avaliação dos frameworks.

5.5 ANÁLISE COMPARATIVA E AVALIAÇÃO DO ALGORITMO

Nesta seção, compara-se a interface gráfica de usuário gerada porTemplate4EHR, EHRGen e ArchiMed. O objetivo desta avaliação é verificar se os atributos de dados, as terminologias e as restrições dos arquétipos estão presentes na interface gráfica gerada. Para isso, o mesmo arquétipo (i.e., *body mass index)* utilizado por EHRGen e ArchiMed para gerar uma interface gráfica foi escolhido. A interface gráfica gerada por Template4EHR utilizando o arquétipo *body mass index* é mostrada na Figura 54, e as principais similaridades e diferenças são descritas a seguir.

A interface gráfica da Figura 54 mostra que Template4EHR gerou o mesmo resultado de EhRScape e ArchiMed, isto é, os atributos de dados, as terminologias e as restrições do arquétipo *body mass index* estão presentes na interface gráfica. No entanto, observando outras características da geração de interfaces gráficas, destacam-se os seguintes avanços do framework proposto nesta tese: i) Template4EHR permite a escolha dos atributos de dados para geração da interface gráfica; ii) habilita a inserção e remoção dos componentes da interface gráfica; e iii) possui mecanismo de persistência de dados em bancos de dados relacionais e NoSQL.

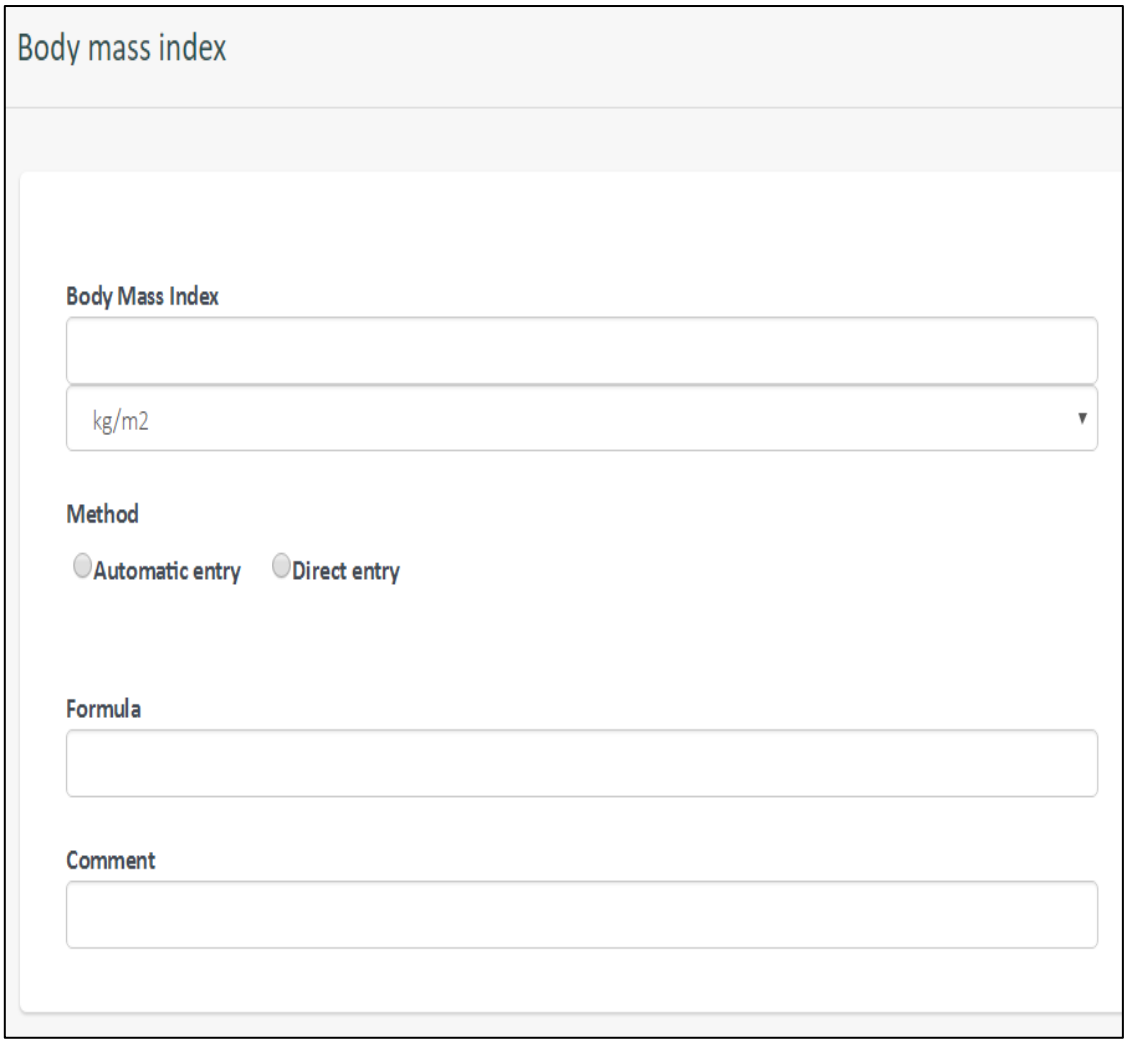


Figura 54 - Interface Gráfica do Arquétipo Body Mass index

O arquétipo utilizado para gerar a interface gráfica da Figura 54 possui poucos atributos de dados e os mesmos estão organizados por meio de um ITEM_LIST. Para avaliar o algoritmo (Seção 4.3.1) de Template4EHR responsável por extrair os atributos de dados, as terminologias e as restrições, procurou-se arquétipos disponíveis no repositório da openEHR (i.e., http://www.openehr.org/ckm/) que atendessem aos seguintes critérios: i) o arquétipo deve possuir pelo menos 5 atributos de dados; ii) atributos de dados com terminologias e/ou restrições associadas; iii) atributos de dados organizados na forma de uma estrutura hierárquica de vários níveis; por fim, iv) variedade em seus tipos de dados (i.e., Text, Ordinal, Boolean, Count, Quantity, Date and Time) especificados. Assim, foram escolhidos os arquétipos: *Family history*, *Apgar e Respiration*.

Para facilitar o entendimento da avaliação do algoritmo, a Tabela 8 mostra para cada arquétipo selecionado, duas colunas nomeadas de: *Elementos e Template4EHR*. A coluna *Elementos* exibe a quantidade de atributos de dados, terminologias e restrições definidas nos arquétipos, enquanto que a coluna *Template4EHR* mostra a quantidade de elementos extraídos pelo algoritmo aqui avaliado. Por exemplo, no arquétipo *Family history*, existem 17 atributos de dados, 4 terminologias e 1 restrição, sendo que o algoritmo extraiu 14, 4 e 1, respectivamente.

Tabela 8 - Resultados dos elementos mapeados pelo Algoritmo

	Family history		Apgar		Respiration	
	Elementos	*Template4EHR*	*Elementos*	*Template4EHR*	*Elementos*	*Template4EHR*
Atributos	17	14	6	6	5	5
Terminologias	4	4	15	15	6	6
Restrições	1	1	1	1	1	1

Como mostra a Tabela 8, para o arquétipo *Family history*, verificou-se que 3 atributos dos 17 não foram extraídos pelo algoritmo de Template4EHR. É importante notar que os atributos não importados referem-se a um tipo de dado da openEHR denominado *Slot*. O tipo *Slot* indica o reuso do conteúdo de outro arquétipo. Por exemplo, caso um especialista necessite representar o endereço de uma pessoa em um arquétipo, uma alternativa é inserir o tipo *Slot*, e vincular a esse tipo, um arquétipo que já contenha os requisitos do endereço de uma pessoa, como por exemplo, o arquétipo *openEHR-DEMOGRAPHIC-ADDRESS.address*. Como o tipo *Slot* não representa um atributo de entrada de dados em um arquétipo (i.e., text, ordinal, boolean, count, quantity, date and time), Template4EHR não considera atributos desse tipo de dado.

As terminologias e restrições especificadas no arquétipo *Family history* tiveram todos os seus itens importados. Além disso, o arquétipo *Family history* possui estruturas hierárquicas de diversos níveis para representar seus atributos de dados. O algoritmo proposto demonstrou eficácia na extração dos atributos de dados de arquétipos, independentemente da quantidade de níveis hierárquicos que existam, diferente de ArchiMed (Seção 3.3.2), que não conseguiu mapear as estruturas de dados do tipo ITEM_TREE.

Os arquétipos *Apgar* e *Respiration* tiveram todos os atributos de dados, as restrições e as terminologias extraídas, como mostram os resultados da Tabela 8. A Figura 55 mostra

os resultados obtidos após a importação dos três arquétipos selecionados nesta avaliação. Nela é possível observar que, dos 28 atributos de dados existentes nos três arquétipos, 25 foram extraídos pelo algoritmo de Template4EHR. Além disso, as 25 terminologias e as 3 restrições definidas nos arquétipos foram todas extraídas pelo algoritmo proposto nesta tese.

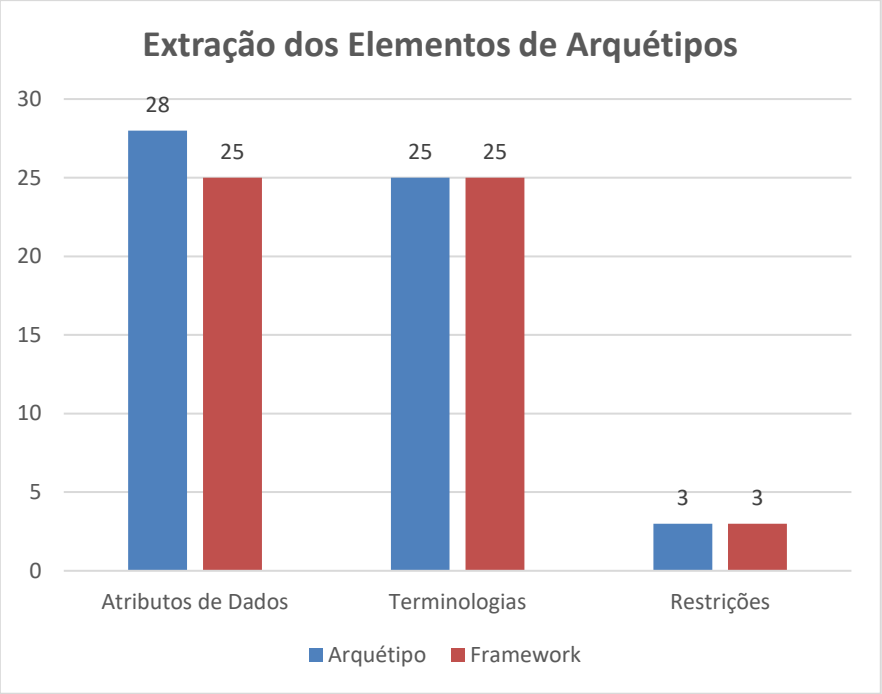


Figura 55 - Elementos Extraídos pelo Algoritmo do Framework

5.6 AVALIAÇÃO DAS FUNCIONALIDADES DO FRAMEWORK

Nesta seção, relata-se os testes realizados com profissionais de TI na avaliação das principais funcionalidades de Template4EHR. Nesse estudo, foram computados a acurácia do framework na geração de esquemas de dados e interfaces gráficas e a eficácia do esquema gerado para manipulação de dados. Dessa forma, tem-se as seguintes métricas de avaliação: a acurácia do framework e a eficácia do esquema de dados. A definição de cada métrica é dada a seguir.

- **Acurácia do Framework ($_{AF}$):** Pontuação dada pelo participante do estudo nas funcionalidades de geração automática do esquema de dados e geração das interfaces gráficas. Cada participante avaliou a importação do arquétipo, a seleção

dos atributos de dados de cada arquétipo, a geração do esquema de dados e a geração das interfaces gráficas.

- **Eficácia do Esquema de dados (EED):** Pontuação dada pelo participante para o processamento de dados sobre o esquema gerado por Template4EHR. O processamento de dados sobre o esquema foi realizado por meio das interfaces gráficas geradas. Esta métrica avalia se o esquema de dados criado no SGBD permite que o usuário realize as operações de inserção, atualização, exclusão e consulta de dados.

Cinco profissionais da área de TI com experiência na área de informática em saúde foram selecionados para avaliar as métricas deste estudo. Cada voluntário avaliou Template4EHR utilizando os seguintes arquétipos: i) *Blood pressure;* ii) *Family history;* iii) *Glasgow coma escale,* e iv) *Anatomic location*. Para cada arquétipo importado no framework, o participante respondeu às perguntas descritas no Quadro 7.

Quadro 7 - Questionário de Avaliação das Funcionalidades do Framework

Identificação do Arquétipo				
Nome				
Objetivo				
Qtd. Elementos de Dados				
Qtd. Elementos Import.				
Avaliação				
S - Satisfatório |**PS -** Parcialmente Satisfatório| **I -** Insatisfatório		**S**	**PS**	**I**
Acurácia do Framework				
1	A importação do Arquétipo foi realizada com sucesso?			
2	Permitiu a seleção de elementos para a importação?			
3	A interface gráfica foi gerada com sucesso?			
4	As terminologias estão presentes na interface gráfica?			
5	As restrições de entrada de dados estão na interface gráfica?			
Eficácia do Esquema de Dados				
6	Inserção de dados realizada com sucesso?			
7	Atualização de dados realizada com sucesso?			
8	Consulta de dados realizada com sucesso?			
9	Exclusão de dados realizada com sucesso?			

Nesta avaliação, adotou-se a referência citada na Seção 2.2.3 deste documento para confeccionar o questionário descrito no Quadro 7. Para cada pergunta, o participante respondeu se o resultado obtido foi satisfatório, parcialmente satisfatório ou insatisfatório.

Cada resposta obteve o peso de 1, 0.5 e 0, respectivamente. Assim, as métricas deste estudo foram calculadas de acordo com a Equação 1.

$$\text{Métrica} = \frac{1}{n}\sum_{i=1}^{n} x_i * 100\% \qquad \textbf{(Equação 1)}$$

Na Equação 1, *x* é o peso de cada resposta e *n* é a quantidade de perguntas da métrica analisada. A Tabela 9 mostra os resultados obtidos após a consolidação das avaliações dos participantes. A avaliação individual de cada participante está disponível em http://archeer.dotdesign.com.br/

Tabela 9 - Resultado da avaliação do Framework

	Acurácia do Framework	Eficácia do esquema de dados
Voluntário 1	97,5 %	100 %
Voluntário 2	100 %	100 %
Voluntário 3	100 %	100 %
Voluntário 4	100 %	100 %
Voluntário 5	100 %	100 %

Os resultados obtidos mostram que as funcionalidades de geração do esquema de dados e de interfaces gráficas tiveram 100% de acurácia, exceto para o voluntário 1, que no quesito extração das restrições do arquétipo *Glasgow coma escale*, avaliou como parcialmente satisfatório, mesmo com todas as restrições estando presentes na interface gráfica.

A Figura 56 mostra a interface gerada utilizando o arquétipo *Family history*. Vale ressaltar que, além dos atributos de dados que são mapeados em componentes da interface gráfica, as terminologias e restrições dos arquétipos também estão presentes na interface. Por exemplo, o campo *Date of birth* da Figura 56, contém uma restrição que define o formato da data em dia, mês e ano. Já o campo *Deceased* e *Age of death*, contém uma lista de terminologias pré-definidas que dão significado semântico a estes campos. Como mencionado anteriormente, caso o usuário queira adicionar ou remover componentes da interface gráfica, basta acessar a funcionalidade de customização de interfaces gráficas e realizar tal configuração.

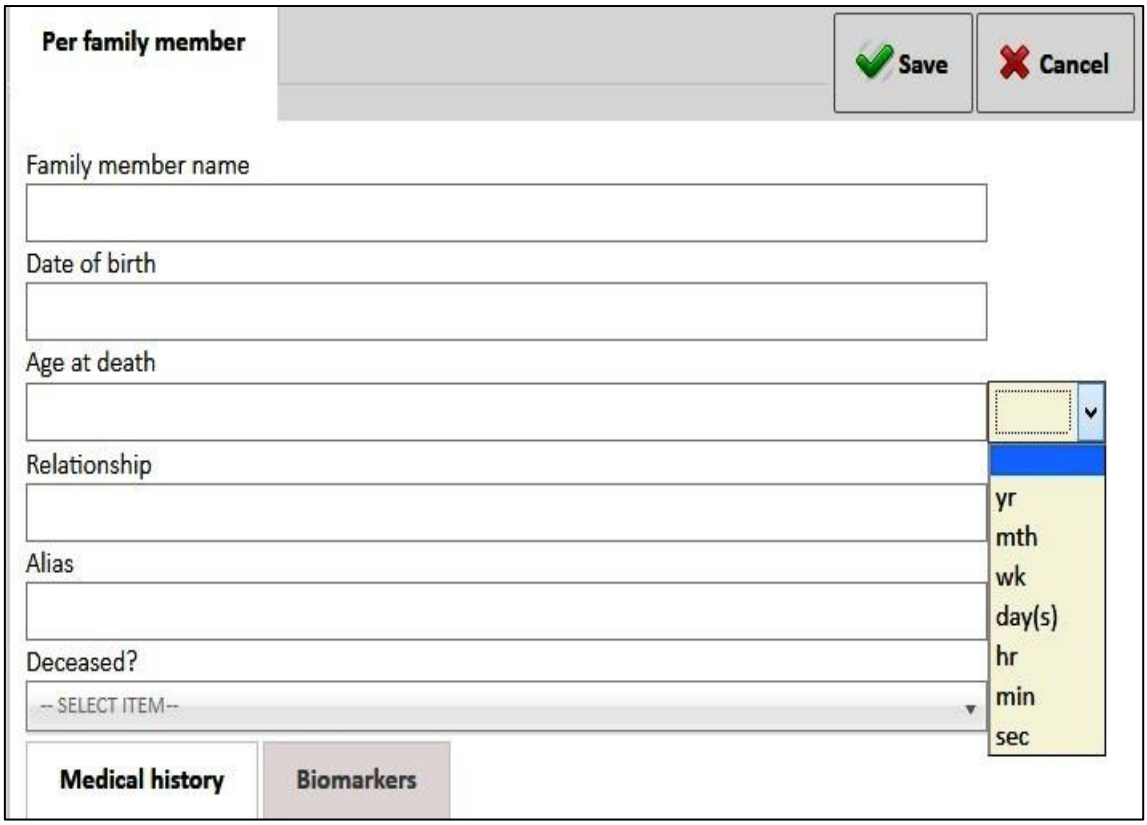
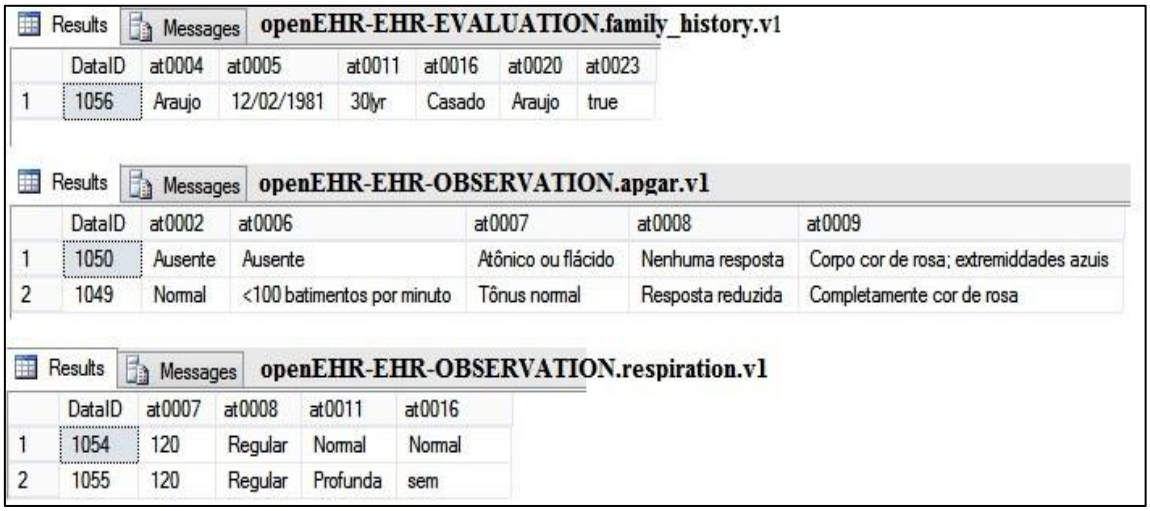


Figura 56 - Interface Gráfica do Arquétipo Family history



Results Messages **openEHR-EHR-EVALUATION.family_history.v1**

	DataID	at0004	at0005	at0011	at0016	at0020	at0023
1	1056	Araujo	12/02/1981	30|yr	Casado	Araujo	true

Results Messages **openEHR-EHR-OBSERVATION.apgar.v1**

	DataID	at0002	at0006	at0007	at0008	at0009
1	1050	Ausente	Ausente	Atônico ou flácido	Nenhuma resposta	Corpo cor de rosa; extremiddades azuis
2	1049	Normal	<100 batimentos por minuto	Tônus normal	Resposta reduzida	Completamente cor de rosa

Results Messages **openEHR-EHR-OBSERVATION.respiration.v1**

	DataID	at0007	at0008	at0011	at0016
1	1054	120	Regular	Normal	Normal
2	1055	120	Regular	Profunda	sem

Figura 57 - Esquema de Dados Criado a partir de Arquétipos

Por fim, após a geração da interface gráfica de cada arquétipo, cada voluntário realizou operações de inserção, atualização, exclusão e consulta de dados. Os resultados apresentados na Tabela 9 mostram que as operações processadas sobre o esquema de dados foram executadas corretamente. A Figura 57 mostra uma instância do esquema de

dados relacional criado por Template4EHR, e alguns registros inseridos pelos participantes por meio das interfaces gráficas.

5.7 AVALIAÇÃO DE ESQUEMAS DE DADOS RELACIONAIS

Conforme descrito na Seção 3.3.2 deste documento, é comum a prática de se mapear arquétipos em SGBD legados. Para avaliar as dificuldades encontradas nessa atividade, solicitou-se a três projetistas de banco de dados que realizassem a tarefa de mapear os mesmos arquétipos da Seção 5.4 (i.e., *Blood pressure, Family history, Glasgow Coma escale* e *Anatomic location*).

Nesta avaliação, nosso objetivo é mensurar o tempo gasto por um projetista de banco de dados para mapear arquétipos em um SGBD e comparar com o tempo gasto pelo framework para criar o mesmo. Da mesma forma que é realizada por Template4EHR, solicitou-se aos projetistas que especificassem um esquema de dados relacional (i.e., tabelas, campos e restrições de integridades referenciais) capaz de armazenar os quatro arquétipos utilizados nesse estudo.

Template4EHR está disponível em template4ehr.azurewebsites.net e o acesso foi realizado por meio de uma estação de trabalho com a seguinte configuração: processador de 2.67 GHz, memória *RAM* de 2GB e navegador *Mozila Firefox*. A Tabela 10 apresenta o tempo gasto em minutos pelos projetistas e por Template4EHR na criação de esquemas dados relacionais utilizando os quatro arquétipos avaliados.

Tabela 10 - Tempo de Mapeamento dos Arquétipos

	Anatomical Location (min)	Family History (min)	Blood Pressure (min)	Glasgow Scale (min)	Redução de Tempo %
Projetista de BD1	06:45	12:33	02:13	05:23	96,94
Projetista de BD2	02:22	07:28	02:02	02:12	94,13
Projetista de BD3	03:23	09:57	02:12	02:23	95,34
Template4EHR	0,01	0,03	0,02	0,02	

Analisando o tempo de criação dos esquemas de dados criados pelos projetistas, constatou-se o seguinte. O primeiro projetista gastou 06:45 minutos para mapear o

primeiro arquétipo, 11:53 min, 12:33 min, 02:13 min e 05:23 min, respectivamente, para mapear os demais. Comparando o tempo gasto pelo projetista 1, com os resultados gerados pelo framework (i.e. 0.01 min, 0.03 min, 0.02 min e 0.02 min), nota-se uma redução de tempo de mapeamento e construção de esquemas de dados em 96,94% com base nos arquétipos testados. Como mostra a Tabela 10, os projetistas 2 e 3 gastaram tempos similares na realização da mesma atividade, porém o tempo gasto por ambos é superior ao tempo gasto pelo framework. Os resultados obtidos mostram que o framework reduziu o tempo de construção de esquemas de dados para os arquétipos testados, em 94,15% em relação ao projetista 2, e 95,34 % em relação ao projetista 3.

Ao final desta avaliação, realizou-se uma pesquisa qualitativa para se identificar as principais dificuldades encontradas durante a realização da atividade solicitada. Duas questões centrais foram relatadas pelos participantes. Primeiro, a dificuldade de se mapear e representar as estruturas de dados hierarquizadas (i.e., ITEM_TREE) de um arquétipo em um SGDB. Segundo, a dificuldade de armazenar e relacionar as terminologias e restrições especificadas nos arquétipos com cada tabela/campo do SGBD. Template4EHR foi descrita pelos participantes como uma ferramenta capaz de auxiliar na criação de esquemas de dados utilizando arquétipos e solucionar as dificuldades acima mencionadas.

5.8 CONSIDERAÇÕES FINAIS

Esta seção apresentou resultados de estudos realizados para avaliar a arquitetura de Template4EHR. Os estudos envolveram desde a análise comparativa do framework com trabalhos correlatos, até a realização de experimentos com profissionais de computação e de saúde para avaliar as principais funcionalidades e os artefatos gerados.

Como principais resultados, destaca-se: a redução de esforço de programação proporcionada por Template4EHR no desenvolvimento de uma aplicação do SUS e o uso de métricas de software para avaliar a conformidade de Template4EHR com as características de manutenibilidade, reusabilidade e extensibilidade. Por fim, a partir de um experimento que avaliou os recursos de geração e customização de interfaces gráficas por meio de Template4EHR e EhRScape, constatou-se que Template4EHR oferece melhores recursos de customização e usabilidade para interfaces gráficas geradas

utilizando arquétipos. Para os usuários que participaram do experimento, a satisfação média de uso das ferramentas foi de 81,22% para Template4EHR e 63,29% para EhRScape.

Na seção 6, apresenta-se as principais contribuições já alcançadas nesta pesquisa, e as indicações de trabalhos futuros.

6 CONCLUSÃO

Esta seção apresenta as considerações finais sobre os tópicos abordados nesta tese, incluindo as principais contribuições alcançadas e as indicações de trabalhos futuros.

6.1 CONSIDERAÇÕES FINAIS

Diversos trabalhos envolvendo as especificações openEHR têm mostrado como modelar um determinado domínio da saúde por meio da modelagem dual, ou como identificar arquétipos que atendam aos requisitos de uma determinada aplicação em saúde. Porém, pouca atenção tem sido dada na construção de uma ferramenta que forneça aos usuários finais do setor de saúde, recursos para gerar interfaces gráficas de usuário customizáveis e criar esquemas de dados em diferentes sistemas de banco de dados utilizando arquétipos.

Este trabalho especificou um framework voltado para o desenvolvimento de aplicações em saúde utilizando arquétipos, chamado de Template4EHR, o qual tem como objetivo construir esquemas de dados para o armazenamento do RES nos paradigmas relacional e NoSQL, e gerar interfaces gráficas de usuário com recursos de persistência poliglota do RES. Para isso, um metamodelo em UML ilustrou os conceitos e relacionamentos da arquitetura dual da openEHR para modelar o RES. Um algoritmo mostrou como os atributos de dados, as terminologias e as restrições são extraídas dos arquétipos e, um conjunto de regras de mapeamento descreveram como as interfaces gráficas de usuário são geradas dinamicamente. Além disso, para garantir que os esquemas de dados gerados por Template4EHR sejam independentes de tecnologia de SGBD, utilizou-se um padrão arquitetural que permite que as regras de negócios da aplicação acessem a camada de persistência de dados sem que a mesma tenha conhecimento da tecnologia de armazenamento dos dados escolhida. Por meio das funcionalidades desenvolvidas no framework, demonstrou-se a criação de uma instância de aplicação em saúde utilizando arquétipos do repositório da openEHR.

Template4EHR foi utilizado para auxiliar no desenvolvimento de uma aplicação em saúde, e verificou-se que 62% de esforços de codificação foram reduzidos. Um conjunto de métricas de software avaliou a conformidade do framework com as características de manutenibilidade, reusabilidade e extensibilidade. As interfaces gráficas e os esquemas

de dados gerados foram avaliados por profissionais de computação e saúde, e o resultados mostraram que todas as operações de persistência de dados nos diferentes SGBD foram corretamente realizadas. Além disso, um experimento avaliou os recursos de geração e customização de interfaces gráficas por meio de Template4EHR e EhRScape, e constatou-se que Template4EHR oferece melhores recursos de customização e usabilidade para as interfaces gráficas geradas utilizando arquétipos.

A partir dos resultados apresentados nesta tese, pode-se responder a pergunta de pesquisa desta tese, que é: como construir esquemas de dados independentes de tecnologias em aplicações de saúde utilizando os atributos de dados, as terminologias e as restrições de arquétipos Além disso, nos locais onde os estudos foram conduzidos: a utilização de um framework de aplicação baseado nas especificações openEHR reduziu o esforço de programação para desenvolver uma aplicação de saúde, melhorou a flexibilidade e a extensibilidade de esquemas de dados e permitiu que os usuários finais criassem suas próprias funcionalidades utilizando arquétipos.

O restante desta seção está organizada como segue. As principais contribuições obtidas neste trabalho são descritas na Seção 6.2. Por fim, os trabalhos futuros que podem dar continuidade ao projeto de pesquisa realizado nesta tese são detalhados na Seção 6.3.

6.2 PRINCIPAIS CONTRIBUIÇÕES

As principais contribuições deste trabalho são detalhadas a seguir. A ordem em que elas são listadas indica a forma como este trabalho foi desenvolvido.

- Metamodelo em UML que ilustra como estão organizados os conceitos da arquitetura dual da openEHR para modelar o RES utilizando arquétipos.
- Algoritmo capaz de extrair de um arquétipo, os atributos de dados, as terminologias e as restrições, independente da quantidade de níveis hierárquicos que organizam os atributos.
- Regras de mapeamento para geração das interfaces gráficas de usuário a partir dos elementos extraídos dos arquétipos.
- Arquitetura de software para geração de esquemas de dados independente da tecnologia de SGBD.
- Serviço em nuvem para geração de interfaces gráficas de usuário customizáveis.

- Aplicação móvel que reutiliza as interfaces gráficas e esquema de dados de Template4EHR.
- Estudos experimentais com profissionais de computação e saúde para validar a acurácia dos esquemas de dados gerados e a usabilidade das interfaces gráficas de usuário.

6.3 TRABALHOS FUTUROS

Para dar continuidade ao trabalho de pesquisa descrito nesta tese, lista-se nesta seção propostas de trabalhos futuros a serem realizadas.

(i) Criação de uma notação para modelar o fluxo de atividades no setor de saúde.

A criação de aplicações em saúde utilizando arquétipos tem evidenciado a falta de uma notação para modelar o fluxo de atividades a serem desempenhadas por profissionais de saúde. Nesse sentido, sugere-se a especificação de uma linguagem de modelagem visual para representar o fluxo, as atividades e os processos de um domínio da saúde utilizando arquétipos. Além disso, sugere-se ainda a criação de um formalismo para descrever os construtores de modelagem, suas propriedades e restrições. Por fim, o desenvolvimento de uma ferramenta CASE é vista como importante para auxiliar os usuários finais na modelagem dos fluxos e processos no setor de saúde.

(ii) Extensão de Template4EHR

Template4EHR utiliza a criação de domínios e subdomínios para vincular as interfaces gráficas de usuário e organizar as funcionalidades de uma aplicação de saúde. No entanto, faz-se necessário que uma aplicação tenha um fluxo de atividade definido para a execução dos cuidados clínicos de um paciente. Nesse sentido, recomenda-se a extensão da arquitetura de Template4EHR, de modo que o formalismo que descreve a linguagem de modelagem de fluxos e processos, seja utilizada para definir a sequência de ações e atividades a serem realizadas em uma aplicação de saúde.

(iii) Especificação de uma linguagem de consulta e índices para otimizar consultas sobre dados arquetipados

É considerada importante a criação de uma taxonomia de operações de consultas sobre esquemas de dados arquetipados, a definição e implementação de uma linguagem de consulta arquetipada e a criação de índices para otimizar consultas baseadas em arquétipos.

REFERÊNCIAS

[1] A. Noor, M.; Nur F. A. A; Nurul A. M. Z. A novel conceptual framework of Health information systems (HIS) sustainability. International Conference on Research and Innovation in Information Systems (ICRIIS)2017, IEEE, Langkawi, Kedah, pp. 1-6, 10.1109/ICRIIS.2017.8002458.

[2] C. C. Martínez, T. M. Menárguez, B. J. T. Fernández, and J. A. Maldonado, "A model-driven approach for representing clinical archetypes for Semantic Web environments," Journal of Biomedical Informatics, pp.150–164, 2009.

[3] Nahid Z.; Abbas A.; Sougand S. The conceptual model to solve the problem of interoperability in health information systems. 8th International Symposium on Telecommunications (IST) 2016, IEEE, Tehran, Iran, pp. 684-689, 10.1109/ISTEL.2016.7881909

[4] Varela, G.; Paz-Lopez, A.; Becerra, J.A.; Duro, R. A Framework for the Development of Context-Adaptable User Interfaces for Ubiquitous Computing Systems. Sensors 2016, 16, 1049. doi: 10.3390/s16071049

[5] Van Zyl, H.; Kotze, M.; Laubscher, R. Using a Theoretical Framework to Investigate Whether the HIV/AIDS Information Needs of the AfroAIDSinfo Web Portal Members Are Met: A South African eHealth Study. Int. J. Environ. Res. Public Health 2014, 11, 3570-3585. doi: 10.3390/ijerph110403570

[6] E. Marco, A. Thomas, R. Jorg, D. Asuman, L. Gokce, "A Survey and Analysis of Electronic Healthcare Record Standards"; ACM Computing Surveys, pp. 277–315, 2005.

[7] K. k. Lee, W. Tangb, K. Choia. "Alternatives to relational database: Comparison of NoSQL and XML approaches for clinical data storage", Computer Methods and Programs in Biomedicine, pp. 99-109, 2013.

[8] H. Jumaa, P. Rubel, J. Fayn, "An XML-based framework for automating data exchange in healthcare", IEEE International Conference on e-Health Networking Applications and Services - Healthcom, pp.264 – 269, 2010

[9] International Organization for Standardization: ISO/TS 18308 health informatics - requirements for an electronic health record architecture, disponível em http:// www.iso.org/iso/iso_catalogue.htm.

[10] A. L. Jeffrey, L. S, Jeffrey, M. Blackford, “Method of Electronic Health Record Documentation and quality of primary care”, J Am Med Inform Assoc, pp.1019-1024, 2012.

[11] Moner D, Maldonado JA, Robles M,"Archetype modeling methodology."J Biomed Inform. 2018 Mar;79:71-81. doi: 10.1016/j.jbi.2018.02.003

[12] C. A. C. Bezerra, A. M. C Araújo, “Middleware For Heterogeneous Healthcare Data Exchange: A Survey. The Tenth International Conference on Software Engineering Advances”, pp. 409-414,2015, Barcelona, Spain

[13] T. Schuler, S. Garde, S. Heard, T. Beale, “Towards automatically generating graphical user interfaces from openEHR archetypes, ” Stud Health Technol Inform, pp. 221–226, 2006.

[14] B. Bernd, “Advances and Secure Architectural EHR Approaches”; International Journal of Medical informatics, pp. 185-190, 2006

[15] S. Garde, E. Hovenga, J. Buck, P. Knaup, “Expressing clinical data sets with openEHR archetypes: A solid basis for ubiquitous computing”, International Journal of Medical Informatics, pp. 334–341, 2007.

[16] H.V. Lindena, T. Austinb, J. Talmona, “Generic screen representations for future-proof systems, is it possible? There is more to a GUI than meets the eye”, Computer Methods and Programs in Biomedicine, pp. 213–226, 2009.

[17] E. Hovenga, S. Garde, S. Heard, “Nursing constraint models for electronic health records: A vision for domain knowledge governance”, International Journal of Medical Informatics, pp. 886-898, 2005.

[18] L. Xiao, G. Cousinsb, L. Hedermanc, T. Faheyb, B. Dimitrovb, “The Design of an EHR for Clinical Decision Support”, International Conference on Biomedical Engineering and Informatics - BMEI, pp. 2525-2531, 2010.

[19] Dias RD, Cook TW, Freire SM, “Modeling healthcare authorization and claim submissions using the openEHR dual-model approach.” BMC Med Inform Decis Mak. 2011 Oct 2;11:60. doi: 10.1186/1472-6947-11-60.

[20] Chen R, Klein G,"The openEHR Java reference implementation project." Stud Health Technol Inform. 2007;129(Pt 1):58-62.

[21] Legaz-García Mdel C, Menárguez-Tortosa M, Fernández-Breis JT, Chute CG, Tao C, "Transformation of standardized clinical models based on OWL technologies: from CEM to OpenEHR archetypes." J Am Med Inform Assoc. 2015 May;22(3):536-44. doi: 10.1093/jamia/ocu027.

[22] Nogueira JR, Cook TW, Cavalini LT,"Mapping a Nursing Terminology Subset to openEHR Archetypes. A Case Study of the International Classification for Nursing Practice." Methods Inf Med. 2015;54(3):271-5. doi: 10.3414/ME14-01-0053.

[23] Haarbrandt B, Tute E, Marschollek M, "Automated population of an i2b2 clinical data warehouse from an openEHR-based data repository." J Biomed Inform. 2016 Oct;63:277-294. doi: 10.1016/j.jbi.2016.08.007.

[24] Sundvall E, Wei-Kleiner F, Freire SM, Lambrix P, "Querying Archetype-Based Electronic Health Records Using Hadoop and Dewey Encoding of openEHR Models." Stud Health Technol Inform. 2017;235:406-410.

[25] El Helou S, Karvonen T, Yamamoto G, Kume N, Kobayashi S, Kondo E, Hiragi S, Okamoto K, Tamura H, Kuroda T, "Generation of openEHR Test Datasets for Benchmarking." Stud Health Technol Inform. 2017;245:1266

[26] Cardoso de Moraes JL, de Souza WL, Pires LF, do Prado AF,"A methodology based on openEHR archetypes and software agents for developing e-health applications reusing legacy systems." Comput Methods Programs Biomed. 2016 Oct;134:267-87. doi: 10.1016/j.cmpb.2016.07.013

[27] Papež V, Mouček R, "Applying an Archetype-Based Approach to Electroencephalography/Event-Related Potential Experiments in the EEGBase Resource." Front Neuroinform. 2017 Apr 6;11:24. doi: 10.3389/fninf.2017.00024

[28] Santos MR, Bax MP, Kalra D,"Dealing with the archetypes development process for a regional EHR system." Appl Clin Inform. 2012 Jul 6;3(3):258-75. doi: 10.4338/ACI-2011-12-RA-0074

[29] Anani N, Chen R, Prazeres Moreira T, Koch S,"OpenEHR-based representation of guideline compliance data through the example of stroke clinical practice guidelines" Stud Health Technol Inform. 2012;180:487-91

[30] Marco-Ruiz L, Moner D, Maldonado JA, Kolstrup N, Bellika JG,"Archetype-based data warehouse environment to enable the reuse of electronic health record data." BMC Med Inform Decis Mak. 2013 Jan 22;13:11. doi: 10.1186/1472-6947-13-11.

[31] Sundvall E, Nyström M, Karlsson D, Eneling M, Chen R, Örman H,"Applying representational state transfer (REST) architecture to archetype-based electronic health record systems." BMC Med Inform Decis Mak. 2013 May 9;13:57. doi: 10.1186/1472-6947-13-57

[32] Demski H, Garde S, Hildebrand C,"Open data models for smart health interconnected applications: the example of openEHR." BMC Med Inform Decis Mak. 2016 Oct 22;16(1):137.

[33] Kropf S, Chalopin C, Lindner D, Denecke K,"Domain Modeling and Application Development of an Archetype- and XML-based EHRS. Practical Experiences and Lessons Learnt." Appl Clin Inform. 2017 Jun 28;8(2):660-679. doi: 10.4338/ACI-2017-01-RA-0009.

[34] Braun M, Brandt AU, Schulz S, Boeker M,"Validating archetypes for the Multiple Sclerosis Functional Composite."BMC Med Inform Decis Mak. 2014 Aug 3;14:64. doi: 10.1186/1472-6947-14-64

[35] Lin CH, Fann YC, Liou DM, "An exploratory study using an openEHR 2-level modeling approach to represent common data elements."J Am Med Inform Assoc. 2016 Sep;23(5):956-67. doi: 10.1093/jamia/ocv137

[36] Hailemichael MA, Marco-Ruiz L, Bellika JG,"Privacy-preserving Statistical Query and Processing on Distributed OpenEHR Data."Stud Health Technol Inform. 2015;210:766-70.

[37] Kohl CD, Garde S, Knaup P,"Facilitating secondary use of medical data by using openEHR archetypes."Stud Health Technol Inform. 2010;160(Pt 2):1117-21

[38] Saalfeld B, Tute E, Wolf KH, Marschollek M, "Introducing a Method for Transformation of Paper-Based Research Data into Concept-Based Representation with openEHR."Stud Health Technol Inform. 2017;235:151-155.

[39] Haarbrandt B, Wilschko A, Marschollek M,"Modelling of Operative Report Documents for Data Integration into an openEHR-Based Enterprise Data Warehouse."Stud Health Technol Inform. 2016;228:407-11.

[40] Christensen B, Ellingsen G,"Evaluating Model-Driven Development for large-scale EHRs through the openEHR approach."Int J Med Inform. 2016 May;89:43-54. doi: 10.1016/j.ijmedinf.2016.02.004.

[41] Marcos M, Maldonado JA, Martínez-Salvador B, Boscá D, Robles M, "Interoperability of clinical decision-support systems and electronic health records using archetypes: a case study in clinical trial eligibility." J Biomed Inform. 2013 Aug;46(4):676-89. doi: 10.1016/j.jbi.2013.05.004.

[42] Chen R, Georgii-Hemming P, Ahlfeldt H, "Representing a chemotherapy guideline using openEHR and rules."Stud Health Technol Inform. 2009;150:653-7.

[43] Kohl CD, Garde S, Knaup P,"Facilitating the openEHR approach - organizational structures for defining high-quality archetypes."Stud Health Technol Inform. 2008;136:437-42.

[44] Pedersen R, Wynn R, Ellingsen G, "Semantic Interoperable Electronic Patient Records: The Unfolding of Consensus based Archetypes."Stud Health Technol Inform. 2015;210:170-4

[45] Tatsukawa A, Shinohara EY, Kawazoe Y, Imai T, Ohe K,"An analysis of the openehr archetype semantics based on a typed lambda theory."Stud Health Technol Inform. 2013;192:990

[46] Porn AM, Peres LM, Didonet Del Fabro M, "A Process for the Representation of openEHR ADL Archetypes in OWL Ontologies."Stud Health Technol Inform. 2015;216:827-31

[47] Pedersen R, Ulriksen GH, Ellingsen G, "The Contextualization of Archetypes: Clinical Template Governance".Stud Health Technol Inform. 2015;218:40616

[48] Menárguez-Tortosa M, Fernández-Breis JT,"Validation of the openEHR archetype library by using OWL reasoning."Stud Health Technol Inform. 2011;169:789-93.

[49] Moner D, Moreno A, Maldonado JA, Robles M, Parra C, "Using archetypes for defining CDA templates."Stud Health Technol Inform. 2012;180:53-7.

[50] Hägglund M, Chen R, Koch S,"Modeling shared care plans using CONTsys and openEHR to support shared homecare of the elderly." Am Med Inform Assoc. 2011 Jan-Feb;18(1):66-9. doi: 10.1136/jamia.2009.000216

[51] Leslie H,"International developments in openEHR archetypes and templates."Health Inf Manag. 2008;37(1):38-9.

[52] Späth M. B., Grimson J. “Applying the archetype approach to the database of a biobank information management system”, International Journal of Medical Informatics, pp. 1-22, 2010.

[53] An open domain-driven platform for developing flexible e-health systems, disponível em https://www.openehr.org/

[54] M. R. Santos, M. P. Bax, L. M. F. Diniz, “Codificando Extratos de Dados Clínicos com Base no Modelo de Referência da Norma ISO 13606 e Arquétipos”, XII Congresso Brasileiro de Informática em Saúde, pp. 1-15, 2010.

[55] L. Lezcano, A. S. Miguel, S. C. Rodríguez, “Integrating reasoning and clinical archetypes using OWL ontologies and SWRL rules”, Journal of Biomedical Informatics, pp. 1-11, 2010.

[56] Teodoro D, Sundvall E, João Junior M, Ruch P, Miranda Freire S,"ORBDA: An openEHR benchmark dataset for performance assessment of electronic health record servers."PLoS One. 2018 Jan 2;13(1):e0190028. doi: 10.1371/journal.pone.0190028

[57] R. Chen, G. O. Klein, E. Sundvall, D. Karlsson, H. Åhlfeldt, “Archetype-based conversion of EHR content models: pilot experience with a regional EHR system”, BMC Medical Informatics and Decision Making, pp. 9-33, 2009.

[58] J. Buck , S. Garde, C. D. Kohl, G. P. Knaup, “Towards a comprehensive electronic patient record to support an innovative individual care concept for premature infants using the openEHR approach”, International Journal of Medical Informatics, pp.521-531, 2009.

[59] D. Georg, C. Judith, R. Christoph, “Towards plug-and-play integration of archetypes into legacy electronic health record systems: the ArchiMed experience”, BMC Medical Informatics and Decision Making, pp.1-12, 2013.

[60] K. Bernstein, R. M. Bruun, S. Vingtoft , S. K. Andersen, C. Nøhr, “Modelling and implementing electronic health records in Denmark”, International Journal of Medical Informatic, pp. 213-220, 2005.

[61] B. Kitchenham, T. Dybã, M. Jørgensen, “Evidence-Based Software Engineering”. 26th International Conference Software Engineering, IEEE CS Press, pp. 273-281, 2005.

[62] R. E. Johnson, B. Foote, “Designing reusable classes. Journal of object-oriented programming”. p. 22-35, 1998.

[63] W. Pree, “Design Patterns for object oriented software development”. Addison-Wesley, 1995.

[64] M. Charles, “ERP II: a conceptual framework for next-generation enterprise systems?”, Journal of Enterprise Information Management, pp. 483-497, 2005.

[65] K. Henricksen, J. Indulska, “Developing Context-Aware Pervasive Computing Applications: Models and Approach”, Pervasive and Mobile Computing Journal, pp. 37-64, 2006.

[66] E. Lopes, U. V. Schiel, A. V. Salgado, “A Conceptual Framework for the Development of Applications Centred on Context and Evidence-Based Practice”.12th International Conference on Enterprise Information Systems - ICEIS 2010, pp. 60-69, 2010.

[67] D. Ingram, P. Schloeffel, S. Heard, D. Kalra, D. Lloyd, T. Beale, “Introducing openEHR: The openEHR Foundation, 2007 ”, disponível em http://www.openehr.org/ releases/1.0.2/openEHR/ introducing_openEHR.pdf, último acesso em outubro 2016.

[68] D. Lloyd, T. Beale, S. Heard, “openEHR Architecture: Architecture Overview”, 2008, disponível em http: //www.openehr.org/releases/1.0.2/architecture/ overview.pdf, último acesso em outubro 2016.

[69] K. Mu-Hsing, S. Tony, W.K Andre, M.B. Elizabeth, K. G. Daniel, “Health big data analytics: current perspectives, challenges and potential solutions”, Int. J. Big Data Intelligence, pp.114-126, 2014.

[70] T. Beale, S. Heard, The openEHR Archetype Model: Archetype Definition Language ADL 2, 2007, disponível em http://www.openehr.org/releases/1.0.2/ architecture/am/adl2.pdf, último acesso em Dezembro 2017.

[71] S. R. Seelam, P. Dettori, P. Westerink, B. Bo Yang, “Polyglot Application Auto Scaling Service for Platform As A Service Cloud”, IEEE International Conference on Cloud Engineering, 2015.

[72] K. Kaur, R. Rani, “Managing Data in Healthcare Information Systems: Many Models, One Solution”, IEEE Computer Society, 2015.

[73] R. Sellami, Ss Bhiri, B. Defude, “Supporting multi data stores applications in cloud environments”, IEEE Transactions on Services Computing, 2015.

[74] Lihong Bao,"QoS-based Trust Computing Scheme for SLA Guarantee in Cloud Computing System", International Conference on Computing Intelligence and Information System (CIIS), pp.236 - 240, 2017.

[75] M. Ellison, R. Calinescu, “Richard Paige, Re-engineering the Database Layer of Legacy Applications for Scalable Cloud Deployment”. IEEE/ACM 7th International Conference on Utility and Cloud Computing, 2014.

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

[77] S. M. Jaybhaye; Vahida Z. Attar, "A review on scientific workflow scheduling in cloud computing", 2nd International Conference on Communication and Electronics Systems (ICCES), pp.218-223, 2017.

[78] SEI, “Serpent Overview,” SEI Technical Report CMU/SEI89-UG-2, Carnegie Mellon University Software Engineering Institute, August 1989.

[79] R. Kazman, M. klein, P. Clements, "ATAM: Method for Architecture Evaluation,Technical Report CMU/SEI-2000-TR-004 ESC-TR-2000-004, Carnegie Mellon University Software Engineering Institute, August 2000.

[80] C. Lee, “An evaluation Model for Application Development Frameworks for Web Applications”, Thesis, The Ohio State Universtity, 2012.

[81] R.Ujera, Sudha V. Ragavi, Chandrasegar Thirumalai, "Analyzing correlation coefficient using software metrics", International Conference on Trends in Electronics and Informatics (ICEI), pp.1151-1153, 2017.

[82] Parveen Kumar; Pradeep Tomar, "Design of dynamic metrics to measure component based software", International Conference on Computing, Communication and Automation (ICCCA), pp.753-757, 2017.

[83] Yusuf U. Mshelia; Simon T. Apeh; Olaye Edoghogho, " comparative assessment of software metrics tools", International Conference on Computing Networking and Informatics (ICCNI), pp.1-9, 2017.

[84] Microsoft, " Working with Code Metrics Data", Microsoft Developer Network, disponível em https://msdn.microsoft.com/en-us/library/bb385911.aspx, último acesso em Janeiro de 2018.

[85] Dyba, T., Dingsøyr, T, "Empirical studies of agile software development: a systematic review", Inf. Softw. Technol. 50(9-10), pp.833–859, 2008. doi: DOI:10.1016/j.infsof.2008.01.006.

[86] Cruzes D.S., Dyba T., " Research synthesis in software engineering: a tertiary study", Inf. Softw. Technol. 53(5), pp.440–455, 2011. doi: 10.1016/j.infsof.2011.01.004.

[87] C. Wohlin,P. Runeson, M. Host, M. C. Ohlsson, B. Regnell, A. Wesslén, "Experimentation in Software Engineering", Springer Heidelberg New York Dordrecht London, 2012. DOI 10.1007/978-3-642-29044-2.

[88] Fink, A, "The Survey Handbook", 2nd edn. SAGE, Thousand Oaks/London, 2003.

[89] J. Nielsen, "Usability inspection methods," in Conference companion on Human factors in computing systems - CHI '94, pp. 413–414, 1994.

[90] A. Seffah, M. Donyaee, R. B. Kline, and H. K. Padda, "Usability measurement and metrics: A consolidated model," Softw. Qual. J., pp. 159–178, 2006.

[91] A. M C. Araújo, V. C. Times, M. U. Silva, Carlos A.C. Bezerra, "A CASE Tool for Modeling Healthcare Applications with Archetypes and Analysis Patterns".The Eleventh International Conference on Software Engineering Advances. Rome,Italy: IARIA, p. 206-211, 2016.

[92] A. M. C. Araújo, V. C. Times, S. S. C. Branco, "A Conceptual Data Model for Health Information Systems", The 14th International Conference on Software Engineering Research and Practice. USA: CSREA Press, p. 236-242, 2016.

[93] K. Atalag, H. Y. Yang, E. Tempero, James R. Warren, "Evaluation of Software Maintainability with openEHR - a Comparison of Architectures", International Journal of Medical Informatics, pp. 849-859, 2014.

[94] G. Piho, J. Tepandi, D. Thompson, A. Woerner, M. Parman, "Business Archetypes and Archetype Patterns from the HL7 RIM and openEHR RM Perspectives: Towards

Interoperability and Evolution of Healthcare Models and Software Systems", The 2nd International Workshop on Meta-modelling for Healthcare Systems - MMHS15, pp.553–560, 2015.

[95] C. Chalopina, D. Lindnerb, S. Kropfa, K. Deneckea,"Archetype based patient data modeling to support treatment of pituitary adenomas", MEDINFO 2015: eHealth-enabled Health, pp.178-182, 2015.

[96] G. Piho, J. Tepandi, D. Thompson, T. Tammer, M Parman, V. Puusep, "Archetypes based meta-modeling towards evolutionary, dependable and interoperable healthcare information system", The 1st International Workshop on Meta-modelling for Healthcare Systems (MMHS), pp. 457-464, 2014.

[97] D. Garcia, C. M. C. Moroa, L. M M. Cintho, "Bridging the Gap between Clinical Practice Guidelines and Archetype-Based Electronic Health Records: A Novel Model Proposal", MEDINFO 2015: eHealth-enabled Health, pp.952-953, 2015.

[98] D. Bosca, D. Moner, J. Maldonado, M. Robles, "Combining Archetypes with Fast Health Interoperability Resources in Future-proof Health Information Systems", MEDINFO 2015: eHealth-enabled Health, pp.180-184, 2015.

[99] R. Pederson, G. Ellingsen, "Interoperable Archetypes With a Three Folded Terminology Governance", MEDINFO 2015: eHealth-enabled Health, pp.934-935, 2015.

[100] J. R. M. Nogueira, T. W. Cook, L. T. Cavalini, "Mapping a Nursing Terminology Subset to openEHR Archetypes: A Case Study of the International Classification for Nursing Practice",Methods Inf Med Journal, pp.271-275, 2015.

[101] J. L. C. Moraes, W. L. Souza, L. F. Pires, A. F. Prado,"A methodology based on openEHR archetypes and software agents for developing e-health applications reusing legacy systems", Computer Methods and Programs in Biomedicine, pp.267–287, 2016.

[102] S. Lee, E. Kim, A. K.Monsen, "Public health nurse perceptions of Omaha System data visualization", International Journal of Medical Informatics, pp.826-834, 2015.

[103] V. Dinu, P. Nadkarni. “Guidelines for the Effective Use of Entity-Attribute-Value Modeling for Biomedical Databases”, International Journal of Medical Informatics. pp. 769-779, 2007.

[104] Node + Path Persistence, disponível em https ://openehr.atlassian.net/ wiki/pages/viewpage.action?pageId=6553626, último acesso em novembro de 2016.

[105] L. Wang, L. Min, X. Lu, H. Duan, “Archetype relational mapping - a practical openEHR persistence solution”, BMC Medical Informatics and Decision Making, pp. 1-18, 2015.

[106] S. M. Freire, D. Teodoro, F. W. Kleiner, E. Sundvall, D. Karlsson, P. Lambrix, Plos One Journal, pp.1-20, 2015.

[107] L. Min , L. Wang , X. Lu, H. Duan, "Case Study: Applying OpenEHR Archetypes to a Clinical Data Repository in a Chinese Hospital", MEDINFO 2015: eHealth-enabled Health, pp.207-211, 2015.

[108] G. Kopanitsa, H. Veseli, V. Yampolsky, "Development Implementation and Evaluation of an Information Model for archetype based user responsive medical data visualization", Journal of Biomedical Informatics, pp.196-205, 2015.

[109] G. Kopanitsa, "Evaluation Study for an ISO 13606 Archetype Based Medical Data Visualization Method", J Med System, pp.1-15, 2015.

[110] R. S. Madariaga , A. Muñoz, R. L. Rubí, P. S.Balazote, A. L. Castro, O. Moreno, M. Pascual, "Examining database persistence of ISO/EN 13606 standardized electronic health record extracts: relational vs. NoSQL approaches", BMC Medical Informatics, pp.1-14, 2017.

[111] H. Sun, K. Depraetere, J. Roo, G. Mels, B. Vloed, M. Twagirumukiza, D. Colaert, "Semantic Processing of EHR data for Clinical Research", Journal of Biomedical Informatics, pp. 247-259, 2015.

[112] API Explorer - EHRScape. Disponível em: https://www.ehrscape.com, último acesso em Março de 2018.

[113] EHRServer - CaboLabs. Disponível em : https://cabolabs.com, último acesso em Março de 2018.

[114] S. Prasad, S. B. Avinash, “Application of Polyglot Persistence to Enhance Performance of the Energy Data Management Systems”, International Conference on Advances in Electronics, Computers and Communications - ICAECC, 2014.

[115] C. A. Rombaldo, S. N. A. Souza ; L. S. Souza, “Persistence framework for Object-Relational Database”, 7th Iberian Conference on Information Systems and Technologies, 2012.

[116] A. M. C. Araújo. “ArcheER: Um Modelo Conceitual de Dados Arquetipados para Sistemas de Informação em Saúde”. Dissertação de Mestrado. Universidade Federal de Pernambuco, 2012.

[117] A. M. C. Araújo, V. C. Times, M. U. Silva, “A Cloud Service for Graphical User Interfaces Generation and Electronic Health Record Storage”, Information Technology - New Generations. Advances in Intelligent Systems and Computing, vol 558. Springer, Cham. pp. 257-263, 2017. https://doi.org/10.1007/978-3-319-54978-1_36

[118] A. M. C. Araújo, V. C Times, M. U. Silva, “ A Tool for Generating Health Applications using Archetypes, IEEE Software Journal, DOI 10.1109/MS.2018.110162508, pp. 1-7, 2018.

[119] A. M. C. Araújo, V. C. Times, M. U. Silva, “PolyEHR: A Framework for Polyglot Persistence of the Electronic Health Record”, The 17th International Conference on Internet Computing and Internet of Things. USA: CSREA Press, p. 71-77, 2016.

[120] A. M. C. de Araújo, Valéria C. Times, Marcus U. Silva, Sistema de persistência poliglota para registro de dados de saúde. 2016, Brasil. **Patente**: Privilégio de Inovação. Número do registro: **BR10201601858**, título: "Sistema de persistência poliglota para registro de dados de saúde" , Instituição de registro: INPI - Instituto Nacional da Propriedade Industrial. Depósito: 13/08/2016.

[121] A. M. C. Araújo, V. C. Times, M. U. Silva, C. A. C. Bezerra, “Software Architecture Modeling for Legacy Health Information System Using Polyglot Persistence and Archetypes”, The Twelfth International Conferece on Software Engineering Advances, ICSEA 2017, pp. 122-127, 2017.

[122] A. M. C. Araújo, V. C. Times, M. U. Silva, “Um Serviço para Geração de Esquemas de Dados Utilizando Arquétipos”, XV Congresso Brasileiro de Informática em Saúde, Goiânia, pp. 385-394, 2016.

[123] A. M. C. Araújo, V. C. Times, M. U. Silva, “Dynamically Generationg NoSQL Data Schemas using Archetypes”, 16th International Conference on WWW/Internet, pp. 31-38, 2017.

[124] A. M. C. Araújo, V. C. Times, M. U. Silva, “Template4EHR: Building Dynamically GUIs for the Electronic Health Records Using Archetypes”. 16th IEEE International Conference on Computer and Information Technology, pp. 26-33, 2016. DOI: 10.1109/CIT.2016.43

[125] D. S. Alves, V. C. Times, A. M. C. Araújo, M. U. Silva, A. S. C. Filho, M. A. Novaes, “Usability Analysis of Archetyped Interfaces for the Electronic Health Record: a Comparative Study”. The Tenth International Conference on Advances in Computer-Human Interactions - ACHI 2017, pp. 169-175, 2017.

[126] A. M. C. Araújo, V. C. Times, M. U. Silva, “Building Health Mobile Applications Using Archetypes”. The Tenth International Conference on Advances in Computer-Human Interactions - ACHI 2017, pp. 47-52, 2017.