UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE CIÊNCIAS SOCIAIS APLICADAS
DEPARTAMENTO DE CIÊNCIAS CONTÁBEIS E ATUARIAIS
CURSO DE CIÊNCIAS ATUARIAIS

MATHEUS FELIX DE SOUZA

EASYPREV: Um software low-code para otimizar a gestão dos Regimes Próprios de Previdência Social dos municípios de pequeno porte em Pernambuco

Recife
2024

MATHEUS FELIX DE SOUZA

EASYPREV: Um software low-code para otimizar a gestão dos Regimes Próprios de Previdência Social dos municípios de pequeno porte em Pernambuco

Trabalho de Conclusão de Curso apresentado à Coordenação do Curso de Ciências Atuariais do Campus Recife da Universidade Federal de Pernambuco, na modalidade de monografia, como requisito parcial para obtenção do grau de bacharel em Ciências Atuariais.

Orientador: Wilton Bernardino da Silva

Recife

2024

Ficha de identificação da obra elaborada pelo autor,
através do programa de geração automática do SIB/UFPE

Souza, Matheus Felix de.
EASYPREV: Um software low-code para otimizar a gestão dos Regimes Próprios de Previdência Social dos municípios de pequeno porte em Pernambuco / Matheus Felix de Souza. - Recife, 2024.
65 p. : il.

Orientador(a): Wilton Bernardino da Silva
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal de Pernambuco, Centro de Ciências Sociais Aplicadas, Ciências Atuariais, 2024.

1. Previdência Social. 2. Gestão. 3. RPPS. 4. OutSystems. I. Silva, Wilton Bernardino da. (Orientação). II. Título.

500 CDD (22.ed.)

MATHEUS FELIX DE SOUZA

EASYPREV: Um software low-code para otimizar a gestão dos Regimes Próprios de Previdência Social dos municípios de pequeno porte em Pernambuco

Trabalho de Conclusão de Curso apresentado à Coordenação do Curso de Ciências Atuariais do Campus Recife da Universidade Federal de Pernambuco, na modalidade de monografia, como requisito parcial para obtenção do grau de bacharel em Ciências Atuariais.

Aprovada em: 15/03/2024

BANCA EXAMINADORA

Documento assinado digitalmente
gov.br
WILTON BERNARDINO DA SILVA
Data: 22/03/2024 18:54:35-0300
Verifique em https://validar.iti.gov.br

_____

Prof. Dr. WILTON BERNARDINO DA SILVA (Orientador)
Universidade Federal de Pernambuco

Documento assinado digitalmente
gov.br
RENATA GOMES ALCOFORADO
Data: 22/03/2024 19:08:55-0300
Verifique em https://validar.iti.gov.br

_____

Profª. Drª. RENATA GOMES ALCOFORADO
Universidade Federal de Pernambuco

Documento assinado digitalmente
gov.br
VITOR EMANUEL DE LYRA SANTOS NAVARRETE
Data: 23/03/2024 08:37:37-0300
Verifique em https://validar.iti.gov.br

_____

Prof. Dr. VITOR EMANUEL DE LYRA SANTOS NAVARRETE
Universidade Federal de Pernambuco

Dedico este trabalho ao meu pai, Afrânio, que tanto lutou e se sacrificou por toda a vida na esperança de me proporcionar a educação que ele nunca pôde ter e que, infelizmente, não teve tempo o suficiente de testemunhar esta conquista.

AGRADECIMENTOS

Agradeço primeiramente aos meus pais, Sueli Felix e Afrânio Custódio, que sempre trabalharam arduamente para me proporcionar uma educação de qualidade, pela confiança e apoio depositados em mim na jornada de conquista dos meus sonhos e objetivos.

À minha esposa, Adrian Veloso, por quase uma década de amor, companheirismo, e pelo incansável apoio e confiança no meu potencial, até quando nem mesmo eu confiava mais. Sua incrível capacidade de ver o melhor nas pessoas foi parte fundamental para que eu alcançasse esta meta, sendo um farol de esperança nos meus momentos mais sombrios.

Aos meus melhores amigos, Alan Fernandes e Caio Cartaxo, que são como irmãos para mim. Os momentos de conversa, risadas, diversão e todo o companheirismo, apesar da distância, foram de suma importância para que eu pudesse chegar até aqui.

Aos docentes do Departamento de Ciências Contábeis e Atuarias, em especial ao Professor Filipe Costa, pelos ensinamentos e inspiração ao longo de toda a graduação.

Ao professor Wilton Bernardino, por aceitar ser meu orientador neste trabalho.

E, finalmente, a todos os colegas de curso com quem tive o prazer de um dia dividir sala da aula. Os momentos de descontração tornaram minha jornada acadêmica divertida, leve e, acima de tudo, memorável.

RESUMO

Ao redor do estado de Pernambuco, observa-se que o número de Regimes Próprios de Previdência Social (RPPS) instaurados para o financiamento dos benefícios futuros de seus funcionários públicos, representam aproximadamente 80% do número total dos municípios da unidade federativa, o que, consequentemente, configura que boa parte dos institutos tendem a ser de pequeno porte. Tais RPPS usualmente apresentam dificuldades de pleno funcionamento, muitas vezes tendo que recorrer a ferramentas mecanizadas e pouco otimizadas, como editores de planilhas, para o gerenciamento de suas bases de dados de servidores ativos e inativos. Este trabalho então apresenta a plataforma *OutSystems* e seu método *low-code* de programação, além de seus benefícios para atingir o objetivo deste trabalho, que é o desenvolvimento de uma aplicação *web* responsiva, chamada de *EasyPrev*, objetivando facilitar e automatizar o trabalho diário desses pequenos institutos no que tange ao gerenciamento de suas bases de dados, com o mínimo impacto negativo possível. O programa *EasyPrev* é então concebido, são detalhados todos os aspectos de seu desenvolvimento e suas funcionalidades, além de ser disponibilizado seu código-fonte em repositório aberto *online* para possíveis melhorias futuras.

Palavras-chave: Previdência Social; Gestão; RPPS; OutSystems.

ABSTRACT

Around the state of Pernambuco, it is observed that the number of Own Social Security Regimes (RPPS) established for the financing of future benefits of their employees, make up approximately 80% of the total number of municipalities in the federative unit which, consequently, means that a huge part of the institutes tends to be small. Such RPPS usually face difficulties in their operations, often having to rely on mechanized and poorly optimized tools, such as spreadsheet editors, for their active and inactive workers database management. This work then presents the OutSystems platform and its low-code programming method, in addition to its benefits to achieve the objective of this work, which is the development of a responsive web application, called EasyPrev, aiming to facilitate and automate the daily work of these small institutes in terms of managing their databases, with the least possible negative impact. The EasyPrev program is then conceived, all aspects of its development and functionalities are detailed, and its code is made available on an open online repository for potential future improvements.

Keywords: Social Security; Management, RPPS; OutSystems.

LISTA DE FIGURAS

SUMÁRIO

1. INTRODUÇÃO

Previsto no Artigo 40 da Constituição Federal de 1988, o Regime Próprio de Previdência Social (RPPS), de caráter contributivo e solidário, é destinado aos servidores públicos efetivos da União, Estados, Municípios e Distrito Federal, assegurando-lhes benefício previdenciário através de critérios que garantam a sustentabilidade e o Equilíbrio Atuarial e financeiro (BRASIL, 1988). Sendo assim, esses funcionários públicos têm seu regime previdenciário separado dos trabalhadores da iniciativa privada, com cada ente federativo dentre os citados acima podendo ter seu próprio fundo previdenciário para garantir os benefícios futuros de seus servidores, administrado de forma direta por meio de fundação, ou indireta através de autarquia.

Registram-se aproximadamente dois mil RPPS espalhados pelo Brasil (CORRÊA; QUEIROZ; RIBEIRO, 2014), dos quais cento e quarenta e nove são de municípios do estado de Pernambuco e um deles sendo do próprio governo da unidade federativa, resultando num valor de 80,5% das entidades municipais do estado possuindo um Regime Próprio de Previdência Social para seus servidores públicos (MINISTÉRIO DA PREVIDÊNCIA SOCIAL, 2022). Portanto, boa parte desses municípios podem ser considerados de pequeno porte, consequentemente concentrando os menores números de participantes (CORRÊA; QUEIROZ; RIBEIRO, 2014).

Por muitas vezes, a autarquia responsável pela administração previdenciária (nos casos de administração indireta) ou a prefeitura (se a administração for direta) desses municípios pequenos não conseguem desempenhar uma gestão satisfatória do dia a dia do fundo, seja pela falta de pessoal devidamente qualificado para tal, seja por não terem à disposição instrumentos adequados, pelos custos elevados de uma boa consultoria técnica terceirizada, e diversas outras razões. Isto acarreta um trabalho muito mecanizado, repetitivo e pouco otimizado como, por exemplo, se utilizando do Excel ou outro programa similar de manipulação de planilhas para efetuar o gerenciamento de sua folha salarial e/ou o controle dos dados das populações de servidores que estão na ativa, de aposentados e de pensionistas. Logo, seria de muita valia para um instituto nessa realidade ter à sua disposição um *software* que seja de funcionamento simples e acessível, para que se suscite o seu processo de "transformação digital" de maneira organizada, visando reduzir a mecanização dos serviços diários sem perder de vista a manutenção da segurança dos dados de seus segurados.

Isto posto, o presente trabalho se propõe a criar um *software*, de rápido desenvolvimento e de baixo custo, com o intuito de prestar auxílio ao trabalho dos profissionais dos Regimes

Próprios de Previdência Social dos municípios de pequeno porte. Assim, pode-se propiciar um processo de digitalização e automação, possibilitando que esses institutos possam abandonar o uso de maneiras rudimentares de gestão das suas bases de dados de ativos, aposentados e pensionistas.

Para alcançar tal objetivo, a criação desta aplicação, elaborada e programada pelo próprio autor, foi realizada através da *OutSystems*, uma plataforma de desenvolvimento *low-code* de aplicações *web* com linguagem própria e um ambiente visual de programação, que apresenta aos desenvolvedores opções integradas dedicadas à criação de software, facilitando a vida dos programadores (GOLOVIN, 2017). Desta forma, este tipo de tecnologia possibilita um avanço significativo na concepção de aplicações, proporcionando o método mais acelerado de desenvolvimento já feito e, provavelmente, o que propicia os menores custos (WASZKOWSKI, 2019) se comparado com linguagens de programação mais tradicionais.

A estrutura deste trabalho consiste neste capítulo introdutório, para trazer luz à realidade que se pode apresentar no dia a dia de trabalho de institutos de previdência dos municípios de menor porte, situação essa que serve de motivação para a elaboração deste projeto.

O capítulo seguinte traz o referencial teórico que embasa este trabalho, com um breve resumo da história da Previdência no Brasil, em especial a formação e estruturação atual dos RPPS. Seguidamente, são trazidas definições e discussões acadêmicas acerca do *low-code*, com ênfase na maneira como ele se insere no mercado de desenvolvimento de *software* e o vem revolucionando. Então, é trazido breve histórico, além de informações sobre a organização, arquitetura e funcionamento da plataforma de desenvolvimento *OutSystems*, concluído com contextualização de como a sua escolha assegura a realização dos objetivos do projeto, e em bem menos tempo do que linguagens de programação convencionais.

A seguir, é apresentado capítulo explicitando a metodologia do desenvolvimento da aplicação resultante deste trabalho, com detalhes acerca das suas funcionalidades propostas e como ela é estruturada, seguida de apresentação detalhada dos seus resultados.

E, finalmente, o trabalho se encerra com uma seção de considerações finais, onde discorre-se sobre os resultados do *EasyPrev* e seus possíveis próximos passos, apontando caminhos para melhorias de infraestrutura e novas funcionalidades do aplicativo, visando a escalabilidade e melhor distribuição do *software*.

2. REFERENCIAL TEÓRICO

Este capítulo é dedicado a apresentar o referencial teórico que baseia este trabalho, passando desde conceitos básicos e histórico da previdência brasileira e dos Regimes Próprios de Previdência Social, sobre do que se trata e como funciona a tecnologia por trás da metodologia *low-code*, finalizando com exposição e análise aprofundada do que é a plataforma *OutSystems* e de como ela se estrutura e funciona.

2.1. A PREVIDÊNCIA NO BRASIL E O CENÁRIO DOS RPPS

Destacado como um direito social fundamental pela nossa Carta Magna (GONÇALVES; MATOS; NOGUEIRA, 2022), a previdência brasileira é um dos três pilares fundamentais do conceito de Seguridade Social e uma das políticas públicas de maior importância para a proteção e amparo da nossa sociedade (GUEDES, 2017). Ela teve como seu primeiro marco jurídico a aprovação do Decreto Legislativo nº 4.682 de 1923, mais conhecida como Lei Eloy Chaves, que tratava das Caixas de Aposentadorias e Pensões (CAPs) dos profissionais das empresas ferroviárias. Ao longo dos anos de 1930, com a população brasileira cada vez mais presente nas cidades e com o fortalecimento do movimento sindical, aos poucos foram surgindo os chamados Institutos de Aposentadorias e Pensões (IAPs) (MINISTÉRIO DA PREVIDÊNCIA SOCIAL, 2017).

Com a decretação da Lei n° 3.807 em 1960, a Lei Orgânica de Previdência Social (LOPS), promoveu-se uma unificação dos IAPs existentes até então, estabelecendo o Instituto Nacional de Previdência Social (INPS) (MINISTÉRIO DA PREVIDÊNCIA SOCIAL, 2017). Contudo, esses sistemas vigentes até então contemplavam apenas os servidores estatutários, numa época em que parte do funcionalismo público ainda podia ser contrata sem prestação de concurso público. Para estes casos, o pagamento dos benefícios de aposentadoria ficava a cargo do Tesouro Nacional e sem o recolhimento de encargos, ficando sob a responsabilidade dos IAPs a arrecadação de contribuições para o financiamento de pensões, auxílios e assistência à saúde dos empregados concursados, chamados de “benefícios de família” (GONÇALVES; MATOS; NOGUEIRA, 2022).

Após a estruturação da Seguridade Social dada na Constituição de 1988, temos que a Previdência se divide em três pilares básicos, os quais são explicitados na Figura 1.

Figura 1 - Ramificação da estrutura previdenciária brasileira

Fonte: Gushiken *et al*. (2002, p. 29)

Neste cenário, temos o Regime Geral de Previdência Social (RGPS), de caráter obrigatório e contributivo, responsável pelos benefícios dos trabalhadores formais da iniciativa privada, administrado pelo Instituto Nacional do Seguro Social (INSS). Também temos o Regime de Previdência Complementar (RPC), de natureza facultativa e destinado a suprir a necessidade de complementação de renda do trabalhador inativo, dividido entre as entidades abertas (bancos e seguradoras) com benefícios designados para qualquer cidadão interessado em aderir, e as entidades fechadas (Fundos de Pensão) voltadas para os funcionários de uma determinada empresa. E, por fim, o Regime Próprio de Previdência Social (RPPS), no qual é focado este estudo, com funcionamento similar ao RGPS, porém voltado para os servidores públicos (GUSHIKEN *et al*., 2002).

Com o propósito de administrar, operacionalizar e garantir a manutenção e pagamento dos benefícios aos seus participantes (DAMASCENO; CARVALHO, 2021), os RPPS, estruturando-se como uma autarquia autônoma ou órgão vinculado ao ente federado, onde ele deve ser único, caracterizam-se como um Fundo de Pensão sujeito “à orientação, à supervisão, ao controle e à fiscalização do Ministério da Previdência e Assistência Social” (GUSHIKEN *et al*., 2002, p. 31). O RPPS deve passar por uma fase de acumulação de capital proveniente das contribuições dos setores públicos empregadores e de seus respectivos servidores, para que

esses possam perceber seus benefícios instituídos pelo RPPS no momento de satisfação das condições para tal (BOGONI; FERNANDES, 2011).

Assegurado constitucionalmente os seus serviços aos empegados públicos efetivados seja na União, Estados, Municípios ou Distrito Federal, como bem alude Briguet (2020), os RPPS têm que o financiamento das aposentadorias dar-se-á através de modelo de capitalização. Neste regime de financiamento, os RPPS devem investir os aportes financeiros ao longo do tempo para serem capazes de honrar os compromissos com seus servidores no longo prazo. Sendo assim, conforme elucidado por Gushiken *et al*. (2002), o envolvimento de um Atuário profissional e habilitado é de suma importância nessa garantia, tanto na avaliação inicial e definição de metas aderentes à realidade daquele fundo, quanto nos balanços periódicos e revisões caso sejam identificadas necessidades, para que seja assegurado seu Equilíbrio Atuarial. Equilíbrio do qual, de consoante definição elaborada por Ibrahim (2011, p. 174):

> ... diz respeito à estabilização de massa, isto é, ao controle e prevenção de variações graves no perfil da clientela, como, por exemplo, grandes variações no universo de segurados ou amplas reduções de remuneração, as quais trazem desequilíbrio ao sistema inicialmente projetado. É um equilíbrio financeiro de longo prazo.

Assim sendo, conforme o exposto até aqui, o desafio para que um RPPS consiga com sucesso estimar premissas atuariais e demográficas aderentes à sua população, enquanto gerencia o investimento dos seus aportes financeiros de forma a observar o Equilíbrio Atuarial do fundo, é imenso. Além disso, existem as dificuldades que um município interiorano e de menor porte podem encontrar, principalmente em termos de pouca infraestrutura, poucos servidores dedicados ao trabalho no fundo, muitas vezes não devidamente capacitados para tais funções e/ou as lideranças sendo nomeadas apenas por representar um laço político de confiança da gestão municipal, prática relativamente comum em municípios do interior onde, geralmente, a fiscalização do cumprimento dos requisitos para tais cargos é deficitária (SOUZA, 2020). Observa-se também a falta de um Atuário habilitado para performar as estimações necessárias, ou até mesmo por vezes o instituto possui completo desconhecimento da existência desse profissional tão crucial para o seu funcionamento. Com todos estes pontos, o desafio de garantir sua sustentabilidade e a garantia do futuro dos seus segurados pode ser descomunal.

2.2. O QUE É *LOW-CODE*?

Atravessar o processo de transformação digital é um grande desafio para muitas empresas. Conforme apontado por Bock e Frank (2021), o maior dos obstáculos para se obter êxito nesta empreitada é a falta de desenvolvedores profissionais, além do constante problema

de que muitos projetos de desenvolvimento de *software*, por conta de baixa eficiência, acabam sofrendo ou, até mesmo, falham completamente. Portanto, há muito têm-se buscado alternativas mirando na maior agilidade do processo de desenvolvimento, com uma das soluções encontradas nas últimas duas décadas sendo a Engenharia Orientada a Modelos (*Model-Driven Engineering* – MDE) (DI RUSCIO *et al*., 2022).

O termo *low-code* foi cunhado e veiculado pela primeira vez numa publicação da empresa *Forrester*, o definindo como "plataformas que permitem entrega rápida de aplicações com o mínimo de programação manual, configuração rápida e implantação, para sistemas de engajamento" (RICHARDSON; RYMER, 2014, p. 2), com esta definição evoluindo posteriormente para (RYMER, 2017 *apud*. DI RUSCIO *et al*., 2022, p. 438, tradução própria):

> produtos e/ou serviços em nuvem para desenvolvimento de aplicações que empregam técnicas visuais e declarativas em vez de programação e estão disponíveis aos clientes com baixo - ou nenhum - custo em dinheiro e tempo de treinamento para começar, com custos aumentando proporcionalmente ao valor comercial das plataformas.

Tisi *et al*. (2019) definem as plataformas de *low-code* como um modelo de Plataforma como Serviço (*Platform-as-a-Service* – PaaS), uma vez que boa parte delas ofertam seus serviços via computação em nuvem (*Cloud Computing* – CC); enquanto Di Sipio, Di Ruscio e Nguyen (2020) salientam que os pontos chave para definir as plataformas *low-code* é a flexibilidade, além da facilidade de desenvolvimento e de manutenção.

Nota-se que as definições apresentadas acima foram estabelecidas por uma empresa de pesquisa de mercado e que presta assessoria sobre impacto tecnológico a outras companhias. No caso dos profissionais da indústria de desenvolvimento de *software*, não há ainda unanimidade sobre uma definição consistente acerca do assunto mas, apesar disso, eles costumam descrever este tipo de plataforma através das ferramentas de "arrastar e soltar" (ou *drag-and-drop*) mostrando que o entendimento geral passa por essas plataformas possuírem interface gráfica para que os usuários arrastem e soltem objetos, fazendo de uso de pouca ou, em certos casos, nenhuma digitação de linhas de código (LUO *et al*., 2021). A maior parte das plataformas de desenvolvimento *low-code* fazem uso de elementos predefinidos nas ferramentas de arrastar e soltar, contudo já existem empresas que fazem uso de Inteligência Artificial (IA) com a finalidade de tentar prever o que o utilizador deseja performar (WOO, 2020).

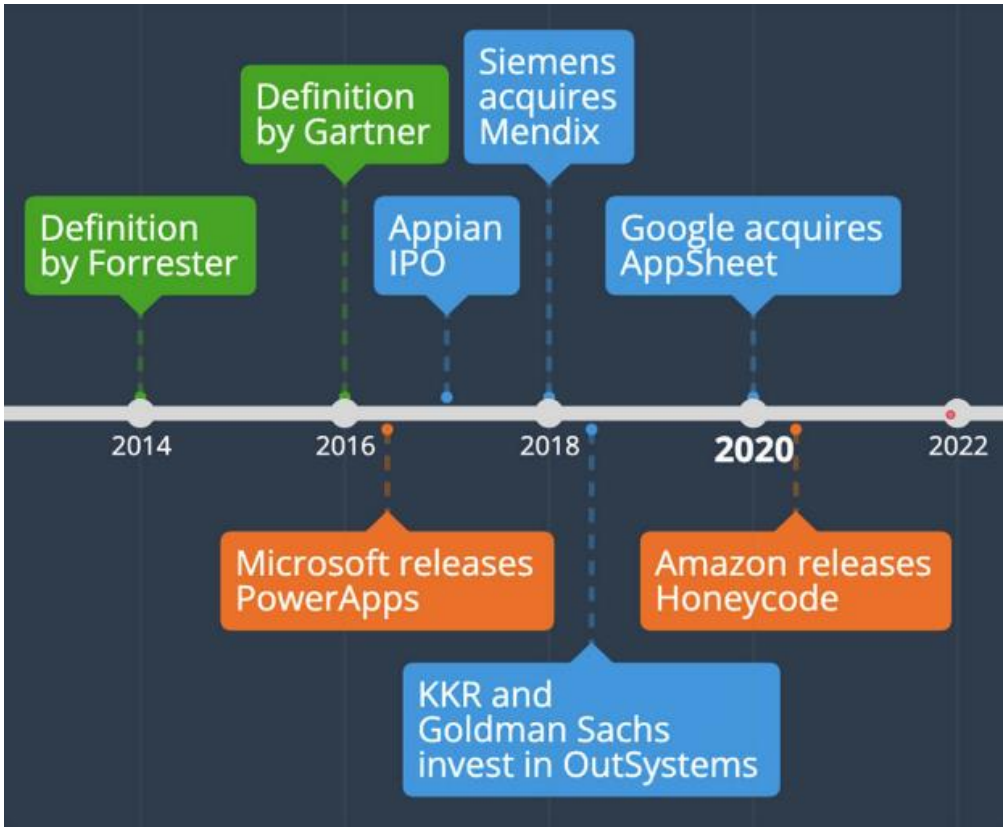
Alguns autores, como é o caso de Cabot (2020), identificam o *low-code* como um subconjunto das metodologias orientadas a modelos, assim como também o faz os autores Bucaioni, Cicchetti e Ciccozzi (2022), que definem o *low-code* como uma coleção de métodos

inserida no contexto de uma metodologia mais ampla, esta muitas das vezes caracterizada como Engenharia Orientada a Modelos. Justamente por isto, conclui-se que não há evidências suficientes para afirmar que as soluções das plataformas *low-code* sejam inovações radicais (BOCK; FRANK, 2021).

Waszkowski (2019), por exemplo, distingue o *low-code* como um conjunto de ferramentas, baseadas no design das aplicações sendo efetuada via interface gráfica para o usuário em vez de técnicas de código bruto convencional, tanto para programadores quanto para não programadores, garantindo rápida geração e entrega de aplicações com um esforço mínimo em termos de escrever linhas de código, configurações de ambientes, treinamento de pessoal, entre outros. Al Alamin *et al.* (2021) constatam que a agilidade e flexibilidade que esse tipo de plataforma de desenvolvimento fornece aos seus usuários acarreta numa maior capacidade de resposta às demandas de mercado, manutenibilidade facilitada e menos trabalho despendido no processo de lançamento de uma aplicação.

Sahay *et al.* (2020) ainda pontuam que, em linhas gerais, essas plataformas são ferramentas bem-vindas em organizações que enfrentem limitações orçamentárias ou de recursos de TI, visto que elas proporcionam que essas companhias sejam capazes de entregar *softwares* bem estruturados em um curto período. A Figura 2 demonstra uma breve linha do tempo com acontecimentos importantes para a história do *low-code*.

Figura 2 - Grandes eventos na história do *low-code*



Fonte: Di Ruscio *et al*. (2022, p. 2)

Conforme observado pelos autores Bock e Frank (2021), o *low-code* distingue-se de metodologias mais tradicionais de desenvolvimento de *software* pois permitem que, em um único ambiente, estejam integradas diversas ferramentas de design de sistemas diferentes e já bem conhecidas que, em outras metodologias mais clássicas, o programador precisaria lidar com uma gama considerável de ferramentas para os mais diferentes propósitos como, por exemplo, modelagem e ligação do banco de dados, editores de interface, entre outras. Com essa integração, é eliminada a necessidade de alternar entre diferentes sistemas, assim possibilitando ganhos em produtividade primariamente por conta da redução dos esforços empregados pelo programador em tarefas rotineiras durante a implementação de projetos de *software* que tenham complexidade de baixa a moderada.

Tais plataformas também conseguem viabilizar que usuários finais de aplicações em empresas, inclusive pessoas com mediano ou baixo conhecimento de determinadas ferramentas e/ou linguagens de programação, consigam performar certos desenvolvimentos. Não programadores capacitados para tal, também denominados como “cidadãos desenvolvedores”,

geralmente são usuários avançados, um funcionário comum ou até mesmo um desenvolvedor envolvido em determinado departamento de negócios (MCKENDRICK, 2017). Não obstante, Sahay *et al.* (2020) sinalizam que essas plataformas ainda assim exigem algum nível de conhecimento sobre desenvolvimento de *software* e, como algumas linguagens *low-code* possuem menos interfaces interativas, ou são menos intuitivas de usar, ou até mesmo menos documentação disponível do que outras, a depender da plataforma escolhida para ser adotada, isto pode ser um fator limitante para os chamados "cidadãos desenvolvedores".

Como mencionado anteriormente, existe a carência de uma definição formal para o *low-code* na literatura especializada. Todavia, os autores Pinho, Aguiar e Amaral (2023, p. 12, tradução própria) se propuseram a formular uma, concluindo o seguinte:

> Desenvolvimento *low-code* é um conjunto de abordagens, tecnologias e ferramentas que permitem o rápido desenvolvimento de aplicações através de técnicas que reduzem a quantidade de código escrito. Essas abordagens podem tornar possível que os usuários finais desenvolvedores programem *software* e usem técnicas e ferramentas muitas vezes incluindo, mas não se limitando a, engenharia orientada a modelos, linguagens de domínio específico e mecanismos de arrastar e soltar.

Qualquer tecnologia traz na bagagem certas limitações e desafios, e com o *low-code* isto não é diferente. Na falta de desenvolvedores profissionais, as aplicações concebidas se utilizando de uma plataforma *low-code* podem apresentar sérias dificuldades tanto para serem atualizadas por conta de novos requisitos técnicos a serem implementados, como também para serem integradas nos outros sistemas da organização em questão (WOO, 2020). E justamente a integração e consumo de conteúdo de sistemas externo à aplicação são os aspectos do desenvolvimento em *low-code* que mais se mostram desafiadores para os programadores, de acordo com o que é observado em fóruns *online* sobre programação, como o *Stack Overflow* (AL ALAMIN *et al*., 2021).

Mediante minuciosa análise de diversas plataformas de desenvolvimento *low-code* distintas, Sahay *et al.* (2020) realizaram o que eles denominaram como "taxonomia" dessas ferramentas, listando seus pontos em comum no que diz respeito às suas capacidades e o que podem ofertar para o desenvolvimento de uma aplicação da seguinte forma:

- Interface gráfica para o programador;
- Suporte à interoperabilidade com serviços e bases de dados externas;
- Suporte à segurança;
- Desenvolvimento colaborativo;
- Suporte a reutilização de código;
- Suporte a escalabilidade;

- Mecanismos de especificação de lógica e regras de negócio;
- Mecanismos para construção de aplicações; e
- Suporte de implantação das aplicações.

Ainda em sua análise, Sahay *et al.* (2020) enumeram as etapas típicas que são percorridas durante o desenvolvimento de uma aplicação, quando da utilização de uma plataforma *low-code*, da seguinte forma:

1. *Modelagem dos dados* – geralmente é uma das primeiras etapas a serem cumpridas, onde os programadores fazem uso das ferramentas visuais disponíveis para criar as entidades e seus respectivos atributos, todo o modelo relacional entre elas, dependências etc.;
2. *Definição da interface de usuário* – configuração e estruturação das páginas da aplicação, validações de acesso de acordo com as atribuições e permissões dos utilizadores finais, formulários e tudo o mais que será renderizado na experiência de usuário; e é aqui que as funcionalidades visuais como "*drag-and-drop*" desempenham papel ainda mais fundamental na aceleração do trabalho do programador;
3. *Especificação de regras de negócio* – gerenciamento do fluxo de trabalho e operações a serem realizadas em *Back-end* ao longo das páginas da aplicação, ou seja, onde o desenvolvedor define o comportamento de botões, links, formulários e demais componentes da interface de usuário, além de como eles irão interagir entre si;
4. *Integração com serviços externos e de terceiros* – criação de integração com *APIs* para satisfazer eventual necessidade de consumo de dados externos à aplicação a ser desenvolvida, conforme a documentação da plataforma *low-code* utilizada orienta; e
5. *Implantação da aplicação* – processo de efetivar o código e, basicamente, colocar a aplicação para funcionar, o que na maioria das plataformas de desenvolvimento *low-code* poder ser realizado com apenas alguns cliques.

Além de uma mudança cultural no desenvolvimento de *software* e sua entrega, essa metodologia também fomenta uma transformação no que diz respeito ao gerenciamento dos recursos tecnológicos e dos processos da corporação. Desta forma, aspectos como uma cultura que premie especialização, tomadas de decisão acerca de projetos centralizados hierarquicamente e métricas do êxito das entregas que somente cargos mais técnicos conseguem

entender, podem dar lugar para uma cultura que condecore o aprendizado em conjunto com a experimentação de técnicas e soluções inovadoras, equipes auto-organizadas e indicadores quantitativos que melhor reflitam os resultados do negócio, respectivamente (RICHARDSON; RYMER, 2014).

Ademais, previsões dão conta de que o mercado das plataformas de desenvolvimento *low-code* pode fechar o ano de 2024 gerando cerca de 32 bilhões de dólares ao redor do mundo. Espera-se também que, até 2026, pelo menos 80% da base de utilizadores destas plataformas sejam de fora de departamentos formais de TI, número este que já ultrapassava os 60% em 2021 (GARTNER, 2022).

Tendo em vista todo o conteúdo abordado nesta seção, é notável o quão poderoso é o *low-code* e como ele está em franca expansão no mercado de desenvolvimento, com um colossal potencial de trazer economia de tempo de programação por conta de suas ferramentas automatizadas que exoneram o programador de boa parte do trabalho mecânico de se dominar uma linguagem de programação e escrever milhares de linhas de código. Além de, consequentemente, propiciar economia de custos a um projeto, uma vez que menos tempo da força de trabalho deve ser empregado para a realização de tarefas similares, se comparado com a utilização de metodologias de programação mais convencionais. Diante disto, para a realização deste projeto foi feita a escolha pela plataforma de desenvolvimento *low-code* *OutSystems*, que será abordada em detalhes na seção a seguir.

2.3. A PLATAFORMA *OUTSYSTEMS*

2.3.1. Breve História

Considerada uma das seis empresas “unicórnio” de Portugal (AGUIAR, 2022), termo empregue para caracterizar uma *start-up* que tenha seu valor de mercado avaliado em pelo menos um bilhão de dólares (MERRIAM-WEBSTER, [s.d.]), a *OutSystems* é uma das maiores plataformas *low-code* para desenvolvimento de *software*, tendo sido reconhecida por sete vezes consecutivas como a líder deste segmento por pesquisas de mercado (GARTNER, 2021 *apud* OUTSYSTEMS, 2023a)

Fundada pelos portugueses Rui Pereira e Paulo Rosado, a companhia surgiu no ano de 2001 (AGUIAR, 2022) como pioneira no mercado de *low-code* e com o propósito de revolucionar o desenvolvimento e entrega de *software* no ambiente empresarial (OUTSYSTEMS, 2023b). Desde sua concepção, foi percebida pelos fundadores a necessidade

de que a empresa crescesse mundialmente, e isto se refletiu inclusive na sua política interna de comunicação e no seu *website*, com o Inglês sendo adotado em ambos como idioma padrão em vez de sua língua materna (AGUIAR, 2022).

Após mais de 20 anos de atuação lhe rendendo diversos prêmios, a *OutSystems* se orgulha de ser uma das líderes de seu segmento de mercado, possibilitando o desenvolvimento mais acelerado de soluções atrelado à redução de custos e escalabilidade garantida, "com escritórios ao redor do mundo, mais de 700.000 membros da comunidade, mais de 400 parceiros, e milhares de clientes ativos em 87 países e através de 22 indústrias" (OUTSYSTEMS, 2023b, tradução própria).

2.3.2. Características Gerais

Modernizar sistemas e digitalizar processos é um objetivo em comum de companhias ao redor do mundo atualmente. Independente de que segmento de mercado seja a atuação da empresa, a chamada "transformação digital" é de suma importância e praticamente mandatória na busca pelo sucesso e sustentabilidade dos seus negócios a longo prazo. Por conseguinte, com a aceleração de concepção e a entrega de soluções facilitada, a *OutSystems* se mostra como uma ferramenta extremamente poderosa na perseguição deste objetivo, ao mesmo tempo que "empodera os desenvolvedores com um ambiente único e integrado de desenvolvimento que cobre todo o ciclo de vida do desenvolvimento (desenvolvimento, garantia de qualidade, implantação, monitoramento e gerenciamento)" (AZOFF, 2018, p. 8, tradução própria).

Golovin (2017, p. 11, tradução própria) define a plataforma como:

> *OutSystems* é uma complexa plataforma de desenvolvimento *low-code* baseada em nuvem projetada com performance, escalabilidade e alta disponibilidade em mente. Uma configuração padrão de Servidor de Aplicação *Web* é complementada por vários serviços e repositórios extras necessários para gerar, construir, empacotar e publicar aplicações.

De foram complementar, Pereira (2022) salienta que a *OutSystems* é capaz de aprimorar e acelerar o desenvolvimento e entrega de aplicações *web* e *mobile*, ao mesmo tempo que garante ao utilizador que ele extrairá ao máximo o que há de melhor no mercado em termos das melhores práticas e padrões de segurança e de performance.

Inclusive, caso os futuros usuários finais de uma aplicação desenvolvida através da *OutSystems* possuam algum tipo de necessidade especial ou deficiência, a plataforma possibilita que seus desenvolvedores consigam projetar suas aplicações ao mesmo tempo que atingem com razoável facilidade os níveis A e AA de conformidade em acessibilidade definidos na versão

mais recentes das Diretrizes de Acessibilidade para Conteúdo *Web* (*Web Content Accessibility Guidelines 2.0*) (GOLOVIN, 2017).

A *OutSystems* dispõe em seu website uma vasta gama de informações, passando desde documentação técnica detalhada até *boot camps*, demonstração de aplicações para exemplificar algumas funcionalidades, cursos gratuitos, jornadas de conhecimento focadas em certas áreas (*Front-end*, *Web* e *Mobile* são alguns exemplos), além das especializações e certificações disponíveis, às quais o profissional pode se submeter aos respectivos exames e contribuir com novas habilidades em seu currículo (OUTSYSTEMS, 2023c). Por conta disso, apesar de possuir uma metodologia e dinâmica de desenvolvimento bem diferentes do que é comumente visto, a plataforma oportuniza não só que profissionais acostumados a desenvolver com linguagem de programação mais tradicionais consigam se adaptar ao desenvolvimento através do *low-code*, mas também que outras pessoas, mesmo que elas venham de ambientes profissionais muito distintos das Tecnologias da Informação, consigam aprender e, eventualmente, trabalhar na área de desenvolvimento de *software*.

Segundo a própria *OutSystems* (2023d), em cerca de uma semana de treinamento com as ferramentas acima exemplificadas, um profissional já pode se tornar produtivo desenvolvendo através da plataforma. E, caso ele já possua alguma familiaridade com outras tecnologias, a transição para trabalhar com *OutSystems* é perfeitamente natural, abstraindo as complexidades de linguagens como *HTML5*, *CSS3* e *JavaScript* para o usuário, enquanto lida com elas nos bastidores. Isso permite que o desenvolvedor se preocupe exclusivamente com a construção das funcionalidades de seu aplicativo sem ter que se preocupar com detalhes técnicos do código ou dominar pormenores de uma linguagem de programação específica.

Considerando os pontos explicitados até aqui, e conforme observado nos resultados obtidos por Martins *et al*. (2020), a depender do tempo despendido em treinamento, podemos considerar a curva de aprendizado em *OutSystems* de rápido crescimento e consolidação do conhecimento, viabilizando a produção de resultados com uma maior qualidade e em menos tempo, se comparado às metodologias tradicionais, visto que “a *OutSystems* usa várias ferramentas já são conhecidas, então a curva de aprendizado se foca mais em aprender como usar todas as funcionalidades que a plataforma oferece” (MARTINS *et al*., 2020, p. 401). Além do exposto aqui, a interface de usuário ser de fácil usabilidade contribui para um entendimento acelerado das situações e, no fim, gera uma vantagem clara sobre o uso de algum método mais tradicional de desenvolvimento de *software*.

Martins *et al*. (2020) apontam ainda que a menor necessidade de escrever código manualmente dos desenvolvedores é fator preponderante para a maior efetividade e facilidade que os autores encontraram na realização de seu trabalho. Além do que, dado o tempo que foi estipulado como prazo limite no começo do projeto dos autores, eles observaram que não seria possível atingirem seus objetivos caso o desenvolvimento da aplicação tivesse ocorrido por meios tradicionais.

Não meramente em corporações e indústrias de múltiplos segmentos, ou empresas de Tecnologia da Informação que prestam serviço de engenharia de *software*, mas a plataforma *OutSystems* também vem sendo adotada no âmbito de universidades. Assim como destacado por Golovin (2017), instituições renomadas dos Estados Unidos como, por exemplo, a *The University of Georgia*, a *Kent State University* e a *Georgia Tech* vêm desenvolvendo aplicações através da *OutSystems* para auxiliar o trabalho de seus pesquisadores. Inclusive contribuindo diretamente para o aprimoramento de abordagens pedagógicas para com os seus alunos, uma vez que a forma simples e visual que o *low-code* da plataforma opera serve de grande atrativo para os estudantes, principalmente aqueles que não possuem alguma experiência prévia com programação. O autor continua: "O processo de Rápido Desenvolvimento de Aplicações é muito importante para tais instituições pois elas usualmente têm que entregar protótipos em um período de tempo muito limitado, e a *OutSystems* lida com isso de forma notável" (GOLOVIN, 2017, p. 14, tradução própria).

Conforme observado nos resultados de estudo de caso acerca da adoção de plataformas *low-code* da sua autoria, Käss, Strahringer e Westner (2017) enquadraram a *OutSystems* nos arquétipos "Democratizador do Desenvolvimento de Aplicações", caracterizada pela ampliação da base de desenvolvedores para aplicações menos sofisticadas e com efeitos positivos na eficiência de custos e velocidade como pontos chave, e "Realizadores de Sinergia", no qual, além dos altos valores entregues em termos de melhorias na eficiência, mostra também alta compatibilidade com sistemas e/ou licenças já existentes do trabalho de uma empresa. Azoff (2018) indica ainda que a *OutSystems* vem mostrando iniciativa em incluir cada vez mais funcionalidades e ferramentas com Inteligência Artificial (IA) incorporadas, buscando deixar o trabalho dos seus utilizadores cada vez mais produtivo. O autor também destaca que parte da força que a plataforma mostra perante a concorrência no mercado de desenvolvimento *low-code* vem de sua capacidade de "endereçar a necessidade de soluções simples que mascaram tecnologias sofisticadas, como a IA" (AZOFF, 2018, p. 30, tradução própria), em outras palavras, o poder que a plataforma demonstra de conseguir abstrair aspectos complexos de

tecnologias e/ou linguagens, lidando em essas complexidades “por trás das cortinas” enquanto entrega aos utilizadores apetrechos bem mais amigáveis e fáceis de usar. Este cenário se ocasiona tanto ao longo do ciclo de vida do desenvolvimento do *software*, como também viabilizando instrumentos automatizados para sua entrega.

E esse nível de abstração das complexidades para os utilizadores se estende também para a segurança das aplicações. Em se tratando da transformação digital de uma companhia é de crucial importância que sejam observados altíssimos níveis de segurança para proteger seus dados e suas operações dos riscos digitais. Tendo isto em vista, a *OutSystems* trabalha continuamente com as melhores práticas do mercado para garantir que tudo o que é desenvolvido em sua plataforma possua excelência nos níveis de segurança, desde as bases da infraestrutura da aplicação, passando pela proteção das bases de dados, fortificação das conexões com servidores contra ataques cibernéticos externos e garantindo a privacidade (OUTSYSTEMS, 2023e). Tudo isto dispondo de contínua vigilância proativa e automatizada visando a proteção das aplicações de ponta a ponta do seu ciclo de vida com medidas de segurança das mais robustas (OUTSYSTEMS, 2023f).

Ademais, a plataforma faz uso de um time interno dedicado para investigar e responder rapidamente a incidentes de vazamento de dados e/ou propriedade intelectual, possuindo diversas certificações relevantes de segurança, como CSA STAR, ISO 9001, dentre muitas outras. Inclusive, está em conformidade com o Regulamento Geral de Proteção de Dados (*General Data Protection Regulation* – GDPR) (OUTSYSTEMS, 2023g) do parlamento da União Europeia, que dispõe as regras de administração e processamento de dados pessoais. A Figura 3 apresenta exemplos de algumas das certificações possuídas pela plataforma.

Figura 3 - Algumas das certificações de segurança da plataforma

Fonte: *OutSystems* (2023g)

Pelo fato de operar sob o modelo de negócio de Plataforma como Serviço (*Platform-as-a-Service* – PaaS), como boa parte das plataformas de desenvolvimento *low-code* (TISI *et al*., 2019), as corporações que desejam fazer uso de sua tecnologia para conceber suas aplicações precisam avaliar os planos de subscrição disponíveis e escolher o que melhor atende suas demandas (OUTSYSTEMS, 2023h), com escalabilidade garantida para os casos de novas necessidades futuras exigirem algum plano mais completo.

Apesar disso, a *OutSystems* oferece uma versão gratuita chamada de *Personal Environment* (Ambiente Pessoal), que é um pequeno ambiente para desenvolvimento com funcionamento totalmente em nuvem e relativamente simples de se configurar, mas que apresenta diversas limitações de funcionamento e capacidades em relação às versões pagas como, por exemplo, quantidade de armazenamento disponível restrito, tendo apenas um ambiente disponível, dentre outras (PEREIRA, 2022). Portanto, dessa forma a plataforma encoraja que seus usuários, ou até mesmo potenciais novos utilizadores, tenham a possibilidade de conhecer a plataforma de maneira gratuita, de estudar para exames de certificação, testar soluções, construir pequenas aplicações sem necessariamente possuir um contexto produtivo e, finalmente, avaliar se a solução *low-code* oferecida por ela faz sentido com o trabalho pretendido para que valha a pena progredir para o serviço pago ou até mesmo evoluir uma assinatura existente, como bem destaca Pereira (2022).

No contexto de trabalho de uma assinatura empresarial, o cliente dispõe do que é chamado de *Factory* (ou Fábrica), que pode ser hospedada em nuvem, seja nos componentes

da própria *OutSystems*, num serviço de nuvem privado (*Amazon Web Services* ou *Microsoft Azure*, por exemplo), ou até mesmo em servidores físicos próprios da empresa (modalidade chamada de *On-premises*) (PEREIRA, 2022). Diferentemente do ambiente pessoal gratuito, uma Fábrica devidamente configurada conta com vários ambientes, habitualmente possuindo pelo menos três ambientes: *DEV*, ou Desenvolvimento, camada mais inferior onde melhorias, correções e novas funcionalidades são desenvolvidas; *QA*, ou *Quality Assurance* (garantia de qualidade), dependendo da empresa também podendo ser encontrado como *CERT*, ou Certificação, uma segunda camada para onde versões estáveis do código da aplicação em Desenvolvimento são subidas para fins de testes pelo time de controle de qualidade; e *PROD*, ou Produção, última camada para onde sobe a versão mais recente e estável do *software*, no sentido de que tudo está devidamente implementado, testado e o mais livre de *bugs* possível, e é por meio do ambiente de Produção que os usuários finais daquele programa o utilizarão.

Os ambientes possuem os mesmos sistemas, bases de dados e funcionalidades, funcionando de maneira independente entre si, sendo possível fazer *Deploy* (ou, mais corriqueiramente, Subida), que nada mais é do que uma cópia do código de uma ou mais aplicações entre ambiente. A depender das necessidades do cliente e do tipo de assinatura, essa quantidade de ambientes pode ser diferente.

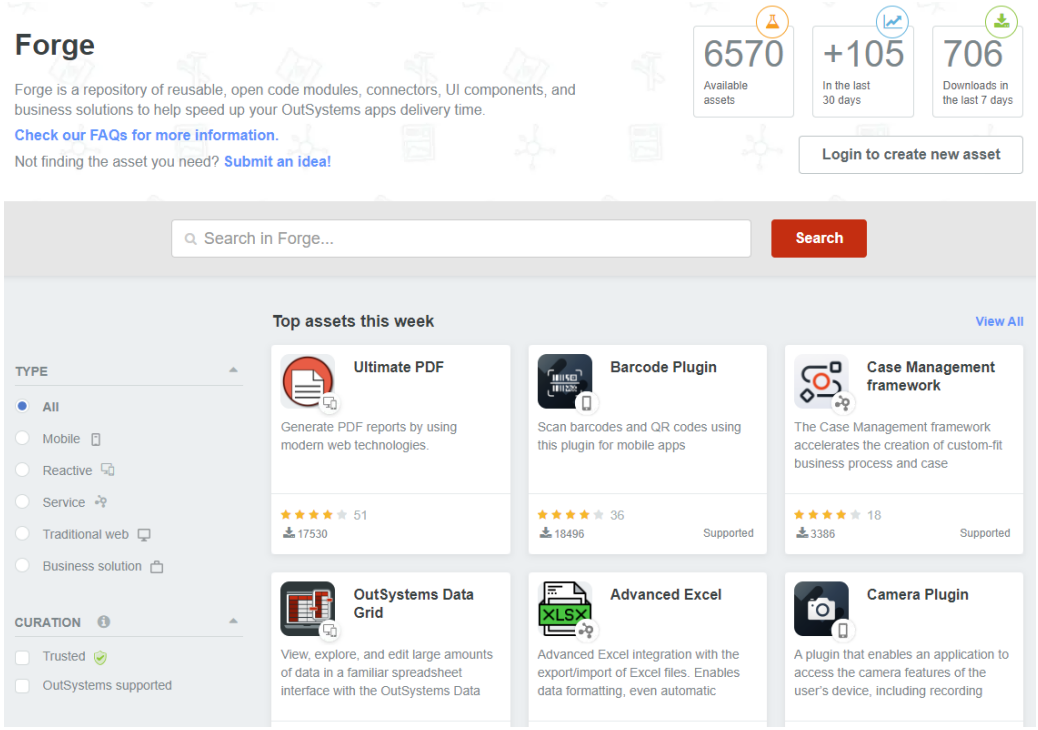
2.3.3. Ecossistema e Arquitetura da Plataforma

Visando unificar esforços na construção e cultivação de um ecossistema de sucesso, a *OutSystems* utiliza um sistema de parcerias com empresas de desenvolvimento de *software* ao redor do mundo e que fazem uso da plataforma em seus projetos, assegurando treinamento das equipes através dos cursos e exames de certificação com as habilidades necessárias para cada etapa de desenvolvimento, buscando aceleração de projetos e focando me ajudar seus parceiros no fomento à inovação e entregas de valor para seus respectivos clientes (OUTSYSTEMS, 2023i). Nessa estrutura, as empresas que aderem ao programa recebem um selo de “Parceira” da *OutSystems* e assim mostram ao mundo que possuem profissionais capacitados para entregar projetos utilizando sua tecnologia, servindo de cartão de visitas para a prospecção de clientes. Inclusive, no *website* da plataforma fica disponibilizada uma página dedicada a listar os seus mais de 400 parceiros (OUTSYSTEMS, 2023j), onde é possível consultar em que posição aquela companhia se encontra em meio ao ranking de parceiros formado por avaliações dos clientes, uma breve biografia, os países onde aquela empresa presta seus serviços, quantos

profissionais certificados ela possui por cada competência na plataforma, dentre outras informações.

E esse senso de comunidade entre a plataforma, empresas e desenvolvedores é sólida em tal nível que o fórum oficial da *OutSystems* em seu *website* é um espaço que reúne milhares de discussões (OUTSYSTEMS, 2023k) postadas por utilizadores de todas as partes do globo, independentemente de seu nível de experiência, possibilitando a partilha de conhecimento, de dúvidas, soluções para problemas, coleta de feedback, ideias, melhores práticas, entre outros (PEREIRA, 2022). Concomitantemente, existe também a *Forge* (OUTSYSTEMS, 2023l), funcionando como um repositório de soluções e *plugins* previamente construídos, postados tanto pela própria *OutSystems* quanto pela comunidade de desenvolvedores. Nela, um programador pode pesquisar por alguma ferramenta ou mecanismo específico para o funcionamento do seu projeto, que pode já ter sido desenvolvido anteriormente por alguma outra equipe em alguma parte do mundo e que foi disponibilizado na *Forge*. Assim, como bem pontua Pereira (2022, p. 73, tradução própria), “podemos evitar construir algo que precisamos do zero, em vez disso reutilizamos algo que alguém já fez, consequentemente economizando tempo e esforço”, contribuindo ativamente para o compartilhamento de conhecimento entre a comunidade. A Figura 4 ilustra a página principal da *Forge*.

Figura 4 - Página principal do repositório *Forge*



Fonte: *OutSystems* (2023l)

Consoante tudo o que foi exposto até aqui, é notável que programar fazendo uso de uma linguagem *low-code* por si só já traz muitos benefícios no que diz respeito à velocidade de desenvolvimento. Para além disto, a *OutSystems* ainda dispõe de mecanismos de dicas movidos por IA ao longo da construção de ações e fluxos lógicos, apresentando sugestões e prováveis próximos passos para o código, de acordo com o que está a ser construído pelo programador. Há, ainda, a possibilidade de dar *prompts* em texto ao construir buscas na base de dados, assim o utilizador não necessariamente precisa elaborar uma *query* manualmente. Estas são algumas das ferramentas disponibilizadas, que visam catalisar o trabalho dos seus usuários, delegando tarefas tediosas e repetitivas para serem performadas pelos modelos de inteligência artificial (OUTSYSTEMS, 2023m) e, gradualmente, a plataforma vem se empenhando para sobrescritar as barreiras no que diz respeito à incorporação de IAs no contexto do desenvolvimento de aplicações (OUTSYSTEMS, 2023n).

Para além das ferramentas supracitadas no tocante ao desenvolvimento, a plataforma ainda oferta o *AI Mentor*, que se caracteriza como um sistema que, através de inteligência artificial, identifica e aponta aos utilizadores partes do código e padrões ao longo da aplicação que possivelmente não estão em conformidade com as melhores práticas de programação e que

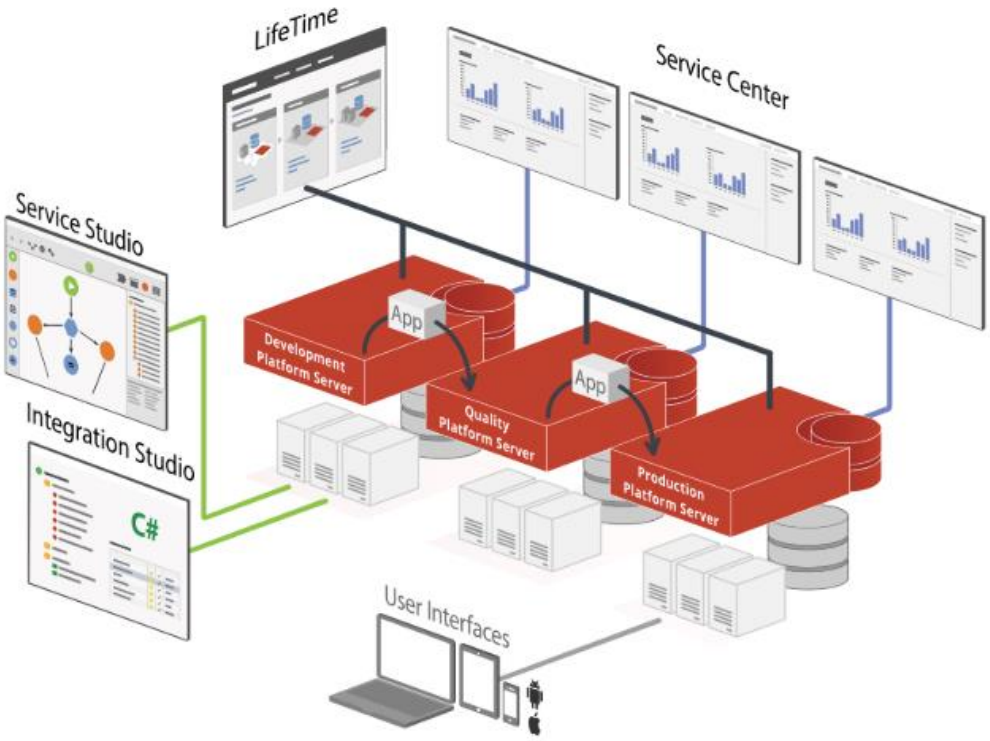
podem melhorados, seja no âmbito da arquitetura do *software*, da segurança, performance, escalabilidade, manutenibilidade do código, dentre outros (OUTSYSTEMS, 2023o).

Agora destrinchando as minúcias do funcionamento da plataforma, ela divide-se, essencialmente, em duas camadas (ROMÃO, 2021):

1. *Development Environment* – aqui se concentram dois produtos complementares, que ficam instalados no computador dos programadores: o ***Service Studio***, ambiente de desenvolvimento integrado (IDE) efetivamente utilizado para programar e construir as aplicações; e o ***Integration Studio***, programa utilizado para construir integrações entre os *softwares* desenvolvidos em *OutSystems* e sistemas ou componentes de terceiros.
2. *Platform Server* – é a parte central, o núcleo da *OutSystems* (MARTINS *et al*., 2020), onde se encontram os serviços e sistemas automatizados da plataforma para gerar, compilar, otimizar, gerenciar, rodar, monitorar e publicar o código efetivo após realizado o processo de publicação das aplicações no *Service Studio*, sejam elas *web* ou, até mesmo, aplicativos nativos para os sistemas *iOS* ou *Android*, e o torna disponível para ser executado nos servidores.

Um esquema elucidando a arquitetura da plataforma *OutSystems* pode ser conferido na Figura 5.

Figura 5 - Ecossistema *OutSystems*



Fonte: Spets (2022, p. 14)

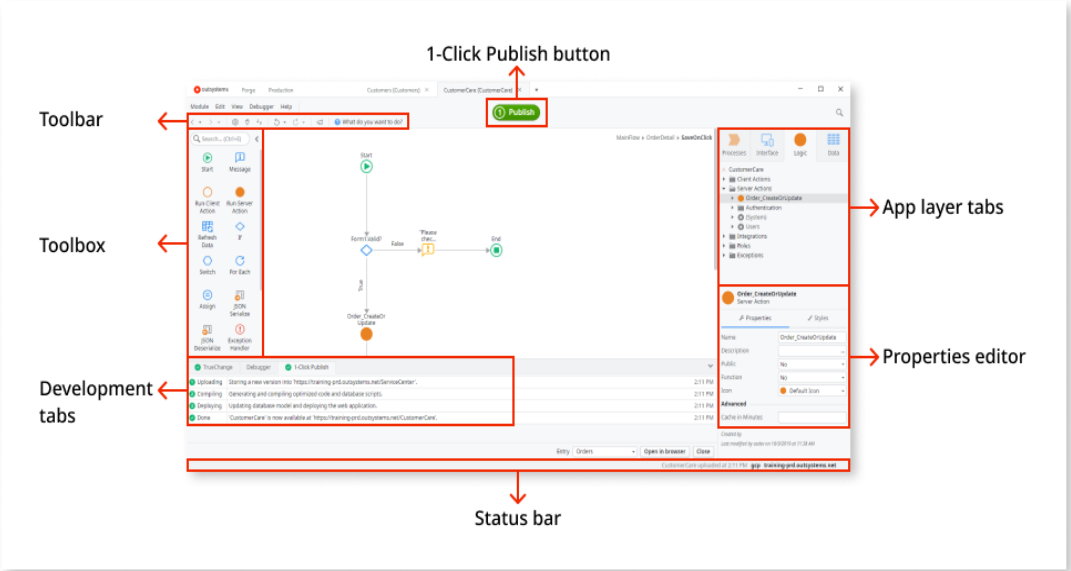
2.3.4. *Service Studio* e *Integration Studio*

Analisando de maneira mais aprofundada, o *Service Studio* é a interface principal que detém as ferramentas e componentes necessários para o programador na criação e construção de todas as partes das aplicações (OUTSYSTEMS, 2023p), passando desde a modelagem da base de dados, todos os fluxos lógicos, validações, *timers*, integrações com serviços *web*, processos síncronos e assíncronos, configurações de segurança, interface de usuário etc. (ROMÃO, 2021). E é por meio dele que o *software* objetivo deste projeto foi construído, desfrutando da simplicidade e intuitividade que a programação *low-code* promove por intermédio de seus aceleradores, estruturando telas e toda a camada de *Front-end* com ferramentas visuais de arrastar e soltar (ou *drag-and-drop*) e demais mecanismos já exemplificados anteriormente, propiciando que o desenvolvedor possa abstrair parte das complexidades envolvidas para focar sua atenção nas regras de negócio da aplicação.

Além disso, como bem evidencia Pereira (2022), o programador ainda dispõe, no *Service Studio*, do auxílio do *TrueChange*, um poderoso mecanismo que constantemente monitora o código e disponibiliza alertas ou mensagens de erro instantaneamente caso seja

encontrado algo que não esteja em conformidade e com o potencial de prejudicar o funcionamento do aplicativo, além de impedir que o código seja publicado para o *Platform Server* com erros. Em seguida, é apresentada a Figura 6, exemplificando como a interface do *Service Studio* e alguns de seus elementos se apresentam ao utilizador quando da criação de um fluxo lógico.

Figura 6 - Exemplo de interface do *Service Studio*



Fonte: OutSystems (2023q)

Também ilustrado na figura acima, o *1-Click Publish Button* se mostra disponível na parte superior central da janela do *Service Studio* sempre que o *TrueChange* não está a detectar inconformidades com o código, e é um dos responsáveis por traz de parte dos automatismos da *OutSystems*. Quando acionado o botão, é iniciado o processo de publicação do código da aplicação no respectivo ambiente, no qual o *Service Studio* envia e salva uma nova versão do código no *Platform Server*, onde são gerados, otimizados e compilados todos os códigos *HTML*, *CSS*, *JavaScript*, *.NET* e *SQL*, além de efetuar possíveis atualizações na base de dados e, finalmente, tornar disponível para rodar no servidor esta nova versão do *software*. Além de propiciar que o desenvolvedor não precise dominar sintaxe e todas as especificidades que advém de programar com as linguagens listadas acima, como muito bem examina Pereira (2022) em sua obra, esta funcionalidade de publicação, chamada de *hot deployment*, consegue manter a aplicação perfeitamente disponível, minimizando o impacto nos utilizadores finais

durante todo o processo, “publicando os arquivos necessários em um diretório virtual, e quando a operação é completada, o diretório anterior é completamente substituído pelo novo” (PEREIRA, 2022, p. 55, tradução própria).

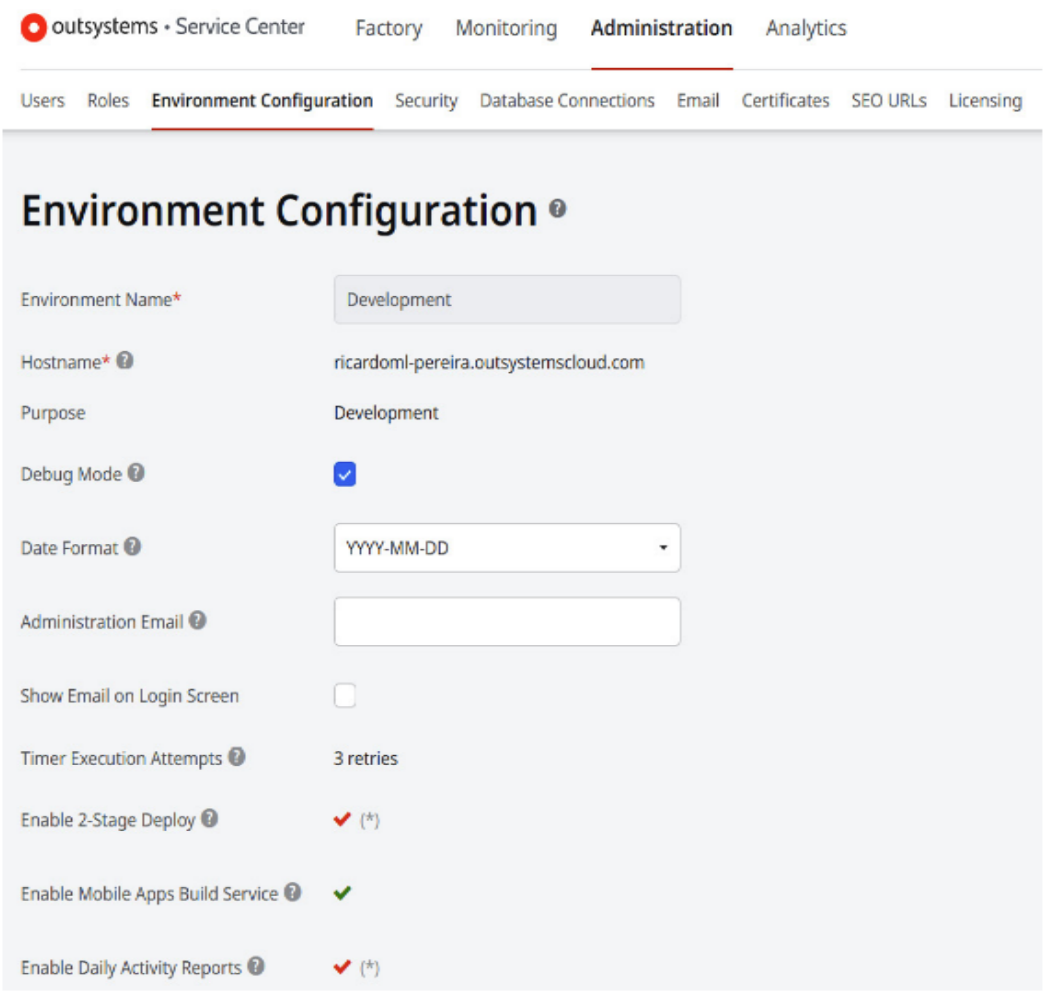
Ainda alinhado com o processo de desenvolvimento de um aplicativo, o *Integration Studio* serve aos utilizadores da plataforma como uma ferramenta para criação de extensões, conexão com bases de dados externas, sistemas próprios ou de terceiros, consumir bibliotecas desenvolvidas sob outras linguagens de programação, dentre outras funções (ROMÃO, 2021). Como de praxe em *OutSystems*, o *Integration Studio* também dispõe de aceleradores para tornar o trabalho mais agradável e produtivo, além de disponibilizar suporte nativo para integrar código programado nas linguagens *C#* e *.NET* (PEREIRA, 2022).

2.3.5. *Service Center* e *Lifetime*

O *Service Center* consiste em um console *web* que permite os utilizadores gerenciarem diferentes aspectos e configurações de um ambiente, inspecionar o funcionamento de serviços (PEREIRA, 2022), além do monitoramento de registros de erros, parâmetros e relatórios acerca da performance das aplicações do ambiente.

É disponibilizada uma instância do *Service Center* para cada ambiente da fábrica, propiciando assim um monitoramento detalhado de cada aplicativo nas diferentes etapas do ciclo de vida do seu desenvolvimento. Nele é possível alterar configurações do ambiente, que vão desde o formato de data padrão que será adotado ao longo de todos os programas nele instalados, parâmetros de segurança, licenças e certificados, opções de *URLs* mais intuitivos e redirecionamentos para otimização de ferramentas de busca (SEO – *Search Engine Optimization*), atribuição de conexão com bases de dados externas, dentre outras. É possível também acessar um histórico de versões publicadas das aplicações, dependências entre módulos da mesma e/ou de outros aplicativos, consultar registros de *logs*, configurar parâmetros de segurança etc. (PEREIRA, 2022). Em seguida, a Figura 7 exemplifica as opções disponibilizadas na aba de Administração e de configuração de ambiente.

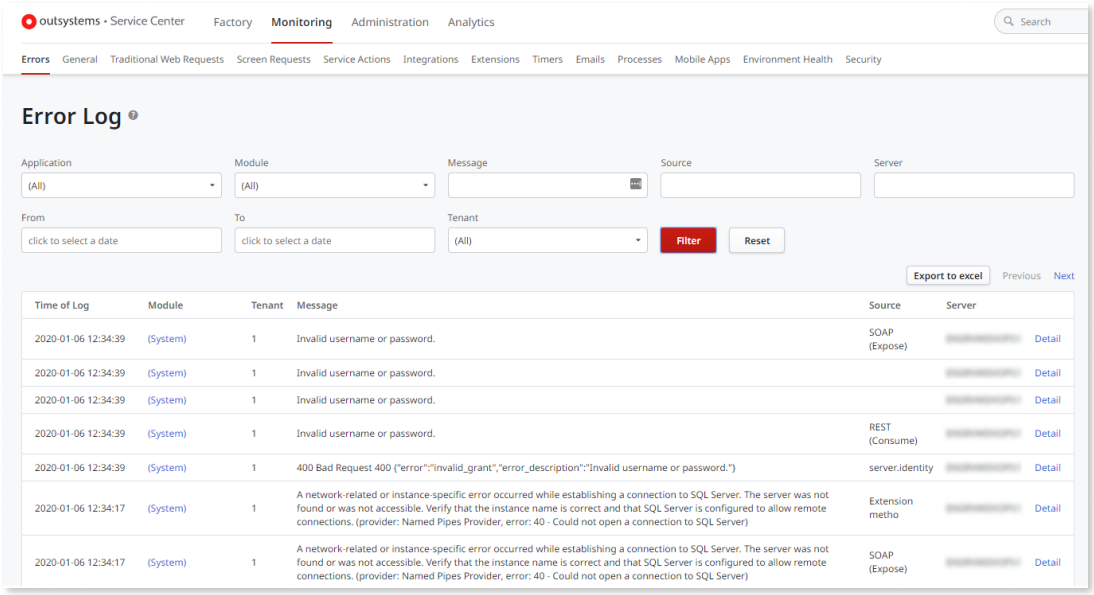
Figura 7 - Exemplo de tela de configurações básicas de ambiente no *Service Center*



Fonte: Pereira (2022, p. 113)

Outra ferramenta importante disponibilizada através do *Service Center* é a aba de monitoramento do ambiente, onde os desenvolvedores podem coletar informações sobre erros, lentidão em consultas à base de dados ou em extensões, performance da renderização de telas, visualização detalhada sobre a execução de *Timers* e de demais processos assíncronos, disponibilidade de integrações à serviços externos e seu desempenho etc. (OUTSYSTEMS, 2024a). Inclusive, tendo a possibilidade de construir *logs* personalizados para os casos de necessidade de monitoramento de aspectos ou eventos específicos do funcionamento da aplicação (PEREIRA, 2022). A seguir, a Figura 8 apresenta um exemplo da tela de gerenciamento dos logs de erros do ambiente.

Figura 8 - Exemplo de monitoramento de logs no *Service Center*

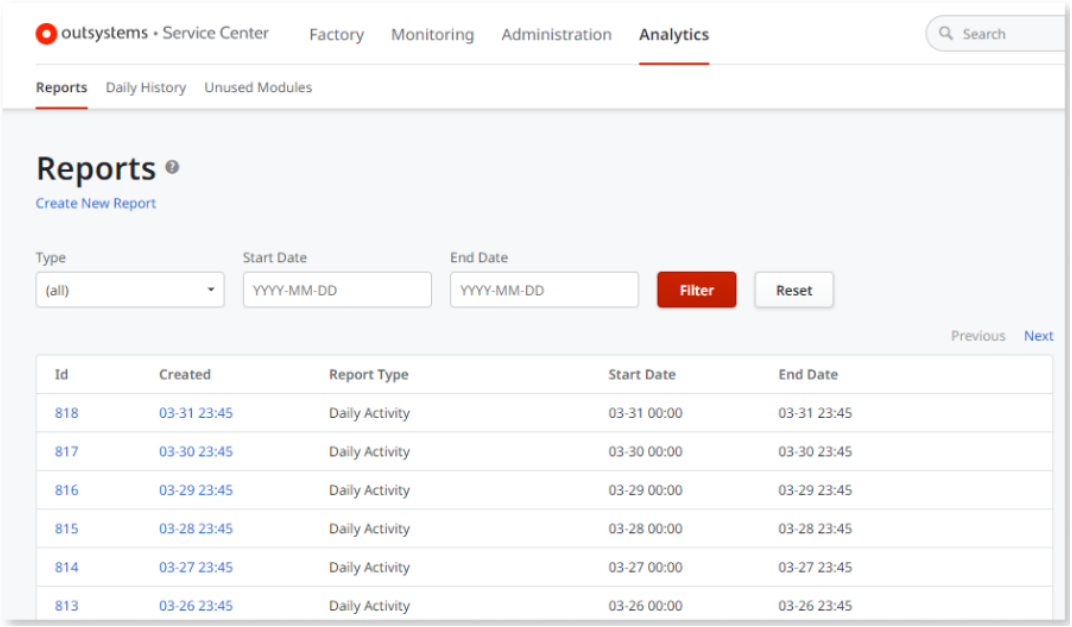


Fonte: *OutSystems* (2024a)

Com as informações registradas nesses logs o programador consegue identificar especificamente quais os pontos da aplicação que estão apresentando erros ou problemas de performance, facilitando assim a resolução de bugs e melhorias de performance, contribuindo para uma melhor experiência de seus cliente e usuários finais.

Mediante a aba *Analytics* do *Service Center*, fica disponível ferramenta completa para a criação de relatórios e colheita de informações sobre o comportamento das aplicações do ambiente por um período em particular, como: histórico de atividades e acessos pelos usuários finais, performance de extensões e integrações externas, de renderização de telas, de consultas às bases de dados, os erros com maior incidência, entre outros (PEREIRA, 2022). Tal ferramenta se apresenta conforme explicitado na Figura 9.

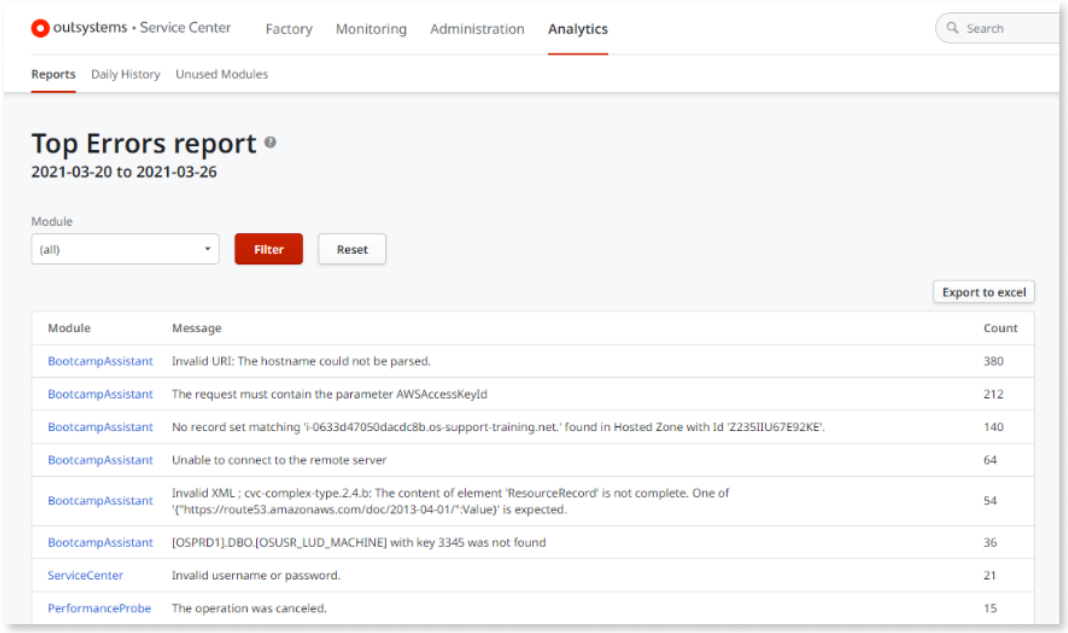
Figura 9 - Exemplo de relatórios no *Service Center*



Fonte: *OutSystems* (2024a)

Dentre os relatórios, um dos mais úteis disponíveis é o *Top Errors*, que apresenta uma listagem agrupada por módulo e por tipo de ocorrência e fornece um valor total referente à quantidade de incidências de um determinado erro. Através desta lista, o desenvolvedor consegue definir prioridades na correção de bugs de acordo com seu impacto, percebendo as raízes do problema e contribuindo com o processo de conserto. Um exemplo desse relatório pode ser averiguado na Figura 10.

Figura 10 – Exemplo de tela do relatório *Top Errors*

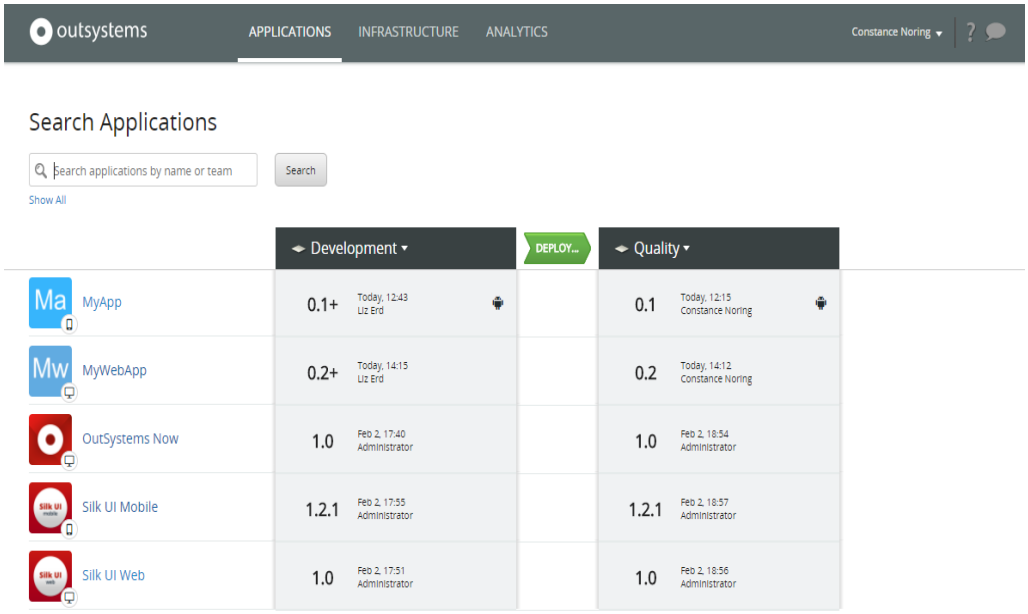


Fonte: *OutSystems* (2024a)

Complementarmente ao *Service Center*, a *OutSystems* estende sua gama de console *web* para o *LifeTime*, centralizando o controle e gerenciamento da infraestrutura da Fábrica como um todo e, consequentemente, todos os seus ambientes (OUTSYSTEMS, 2024b). Por meio do *LifeTime*, é possível configurar parâmetros de segurança, realizar a administração dos utilizadores das equipes de desenvolvimento e suas respectivas atribuições, entre outras configurações.

É com o ele que se faz possível conferir e modificar especificidades de cada ambiente, checar versões das aplicações ou se existem módulos com dependências quebradas ou desatualizadas, inclusive efetuar o *deploy* delas entre diferentes ambientes e inspecionar seu funcionamento após finalizado o processo (PEREIRA, 2022). A Figura 11 apresenta um exemplo de tela de preparação de *deploy* de aplicações, neste caso, a partir de um ambiente de Desenvolvimento para um de Garantia de Qualidade.

Figura 11 - Exemplo de tela do *LifeTime* com preparação de subida de aplicações



Fonte: *OutSystems* (2024c)

Conforme destacado por Pereira (2022), a plataforma é pensada para que o acesso e monitoramento de informações relevantes dos ambientes de uma fábrica sejam realizados de maneira fácil, e o *LifeTime* nos apresenta um exemplo perfeito disso, conferindo aos desenvolvedores controle total e de forma intuitiva sobre todo o ciclo de vida das aplicações.

2.3.6. Tipos de Aplicações

Ao se usar a plataforma *OutSystems*, temos à disposição três tipos diferentes de aplicações *web* que podemos desenvolver: *Traditional*, *Reactive* e *Mobile*. Cada uma delas possuindo suas particularidades a nível de processos, telas, comportamento da interface de usuário, capacidades e de seu funcionamento em geral.

No que diz respeito às aplicações *Traditional Web*, como o próprio nome sugere, são aplicativos *web* que têm seu funcionamento consoante o padrão que programas dessa natureza tradicionalmente possuem: operações sobre a base de dados, renderização de telas e toda a lógica a cada clique ou ação do usuário final em tela é performada exclusivamente no servidor da aplicação (ROMÃO, 2021).

Por sua vez, os aplicativos desenvolvidos em *Reactive*, um padrão de desenvolvimento de aplicações mais recente que, além de ser mais o utilizado para satisfazer a necessidade de

exposição de uma alto volume de dados, como aponta Romão (2021), são constituídos com o propósito de trazer uma interface responsiva de tal maneira que a disposição dos elementos em tela seja capaz de se adaptar a depender do tipo e do tamanho da tela do dispositivo que o esteja reproduzindo, proporcionando uma experiência de usuário para seus clientes devidamente ajustada de acordo com seu dispositivo.

Já os programas em *Mobile* seguem praticamente o mesmo paradigma dos produzidos em *Reactive*, o que inclusive propicia que muitos componentes e padrões desenvolvidos possam ser perfeitamente reaproveitados entre esses tipos de aplicação. Contudo, observa-se no *Mobile* como principais diferenças a possibilidade de utilização de recursos do dispositivo para aproveitamento em suas funcionalidades (câmera e leitor de impressão digital são alguns exemplos), e o fato dos aplicativos apresentarem a capacidade de guardar dados *offline* no dispositivo cliente, além de mecanismos de sincronia desses dados com as tabelas correspondentes no servidor que rodam de maneira assíncrona (ROMÃO, 2021). Dessa forma, permite-se que o programa possa ser usado mesmo na falta de acesso constante à internet, com menos limitações nas suas capacidades.

Existem algumas diferenças significativas quando comparamos *Traditional* e *Reactive* em termos de funcionamento e de elementos disponíveis para uso durante o desenvolvimento da aplicação no *Service Studio*. Em *Traditional*, toda e qualquer ação e fluxo lógico é necessariamente rodado no servidor, o que não acontece em *Reactive*. Por conta de suas capacidades de disparar eventos e executar ações cuja lógica é performada inteiramente utilizando processamento do dispositivo do usuário final ao invés de sempre necessitar que o servidor faça isso, um programa em *Reactive* não tem necessidade de funcionalidades para ordenar no fluxo lógico de uma ação que partes da tela sejam atualizadas, por exemplo. Sendo assim, no instante que são detectadas atualizações nos dados e/ou na lógica da tela, a renderização desses locais específicos é automaticamente performada, assim facilitando o processo de construção da interface de usuário, promovendo algumas preocupações a menos para os desenvolvedores (ROMÃO, 2021).

Outra diferença notável entre as duas tecnologias é que em *Traditional* existe a figura do *Preparation*, que é o primeiro fluxo lógico que é executado ao se redirecionar para uma tela. Nesta ação, o programador posiciona toda a lógica para validação de acesso àquela tela, buscas às bases de dados, tratamento desses dados retornados caso seja necessário, dentre outros. Terminada a execução completa do *Preparation*, todas as partes visuais da tela são, então, renderizadas. Já no caso de uma aplicação em *Reactive*, o desenvolvedor exerce maior poder

sobre o fluxo de carregamento de uma página, em virtude da possibilidade de serem realizadas mais de uma ação do lado do servidor e que são executadas de maneira totalmente assíncrona. Consequentemente, a tela não fica em dependência do término dessa lógica, pois todo o seu *layout* é logo renderizado para o usuário final e, conforme essas consultas ao servidor vão sendo finalizadas, apenas as partes dependentes dessas informações na página são atualizadas e renderizadas novamente.

Conforme o exposto, aplicações em *Reactive* contribuem amplamente para uma experiência de usuário mais suave e agradável, visto que o tempo despendido entre uma ação realizada pelo utilizador em tela e uma resposta visual de que está sendo performado o carregamento da próxima etapa fica notoriamente menor.

Neste trabalho, portanto, optou-se pelo desenvolvimento do *EasyPrev*, utilizando o Ambiente Pessoal gratuito possuído pelo autor na *OutSystems*, sob a tecnologia de aplicação *Reactive*. Não apenas esta metodologia traz inúmeras vantagens em relação ao *Traditional Web*, dada sua versatilidade para ser executada com qualidade e velocidade, como também não foi identificada necessidade de o aplicativo ser exclusivamente *Mobile*, visto que aplicações *Reactive* são plenamente capazes de adaptarem-se a qualquer tipo de dispositivo que esteja sendo utilizado pelo usuário final, quando em *Mobile* o *EasyPrev* estaria limitado apenas a dispositivos móveis.

3. METODOLOGIA

Ao longo deste capítulo serão expostos detalhes acerca dos objetivos e de todo o desenvolvimento da aplicação, passando desde como sua arquitetura foi pensada até detalhes de suas funcionalidades e seu resultado.

3.1 ARQUITETURA DA APLICAÇÃO

Pensando em um maior ordenamento e na garantia de uma melhor escalabilidade do projeto, a arquitetura deste projeto foi toda pensada fazendo uso das melhores práticas disponíveis pela *OutSystems* em sua extensa documentação, de modo a cumprir padrões criteriosos de organização em camadas e subcamadas que se comunicam entre si. Sendo assim, o *EasyPrev* divide-se em quatro aplicações, onde tanto elas quantos seus respectivos módulos seguem padronizações e convenções de nomenclatura.

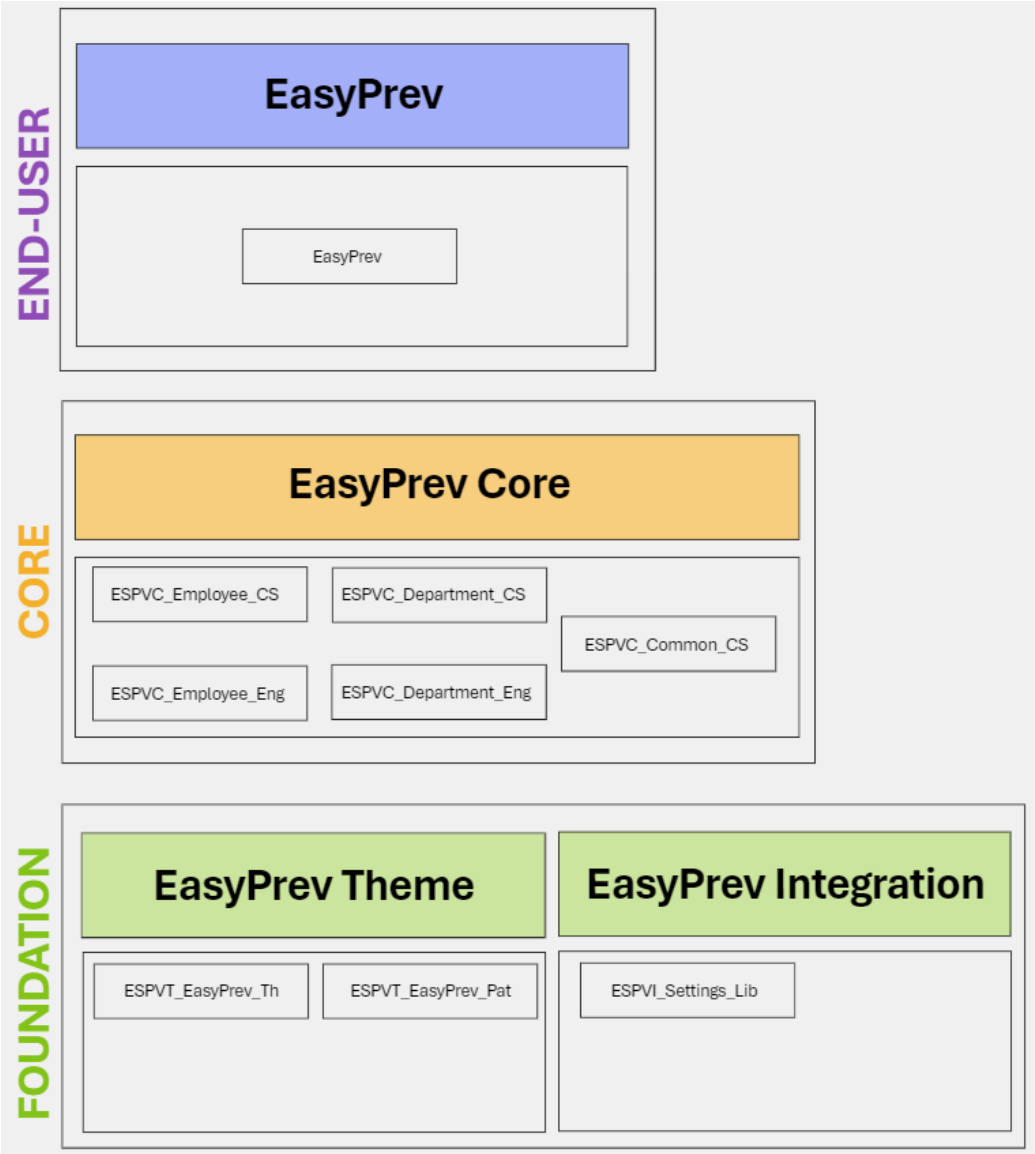
Começando pela camada mais basilar, comumente denominada como *Foundation*, ou Fundação, esta porção da aplicação fica responsável por abrigar componentes e requisitos classificados como não funcionais, ou seja, que não se relacionam diretamente com as funcionalidades específicas oferecidas pelo *software*, como temas, estilos, bibliotecas ou outros serviços externos, de modo que possam ser utilizados para estender a aplicabilidade das funções do programa. Neste sentido, foram construídas as aplicações *EasyPrev Theme* e *EasyPrev Integration*, onde a primeira acomoda os módulos responsáveis pelo tema e todos os padrões visuais reutilizáveis, e a segunda contém suporte para integrações e manipulação de configurações globais do aplicativo.

A camada logo acima, chamada de *Core*, encarrega-se e encapsular regras de negócio específicos e reutilizáveis do conceito da aplicação, além da possibilidade de gerar exposição de dados e serviços para o exterior do *software*. Para este fim, foi criada a aplicação *EasyPrev Core*, abarcando módulos para tratar da modelagem da base de dados e as ações básicas capazes de criar e eliminar registros nas tabelas, além dos módulos destinados à orquestração e lógica de negócio mais refinada.

Diferentemente das explanadas acima que se direcionam, em linhas gerais, a serviços reutilizáveis, esta última e mais acima camada se destina ao *Frontend* e às interações com o usuário final do programa, lhe provendo as funcionalidades devidas. Para tal finalidade, foi criada a aplicação *EasyPrev*, abrigando um único e homônimo módulo suprido de todas as telas,

capacidades e ferramentas que o *software* será capaz de prover. A Figura 12 mostra em detalhes as diferentes camadas da arquitetura do *EasyPrev*, assim como seus respectivos módulos.

Figura 12 - Arquitetura do *EasyPrev*



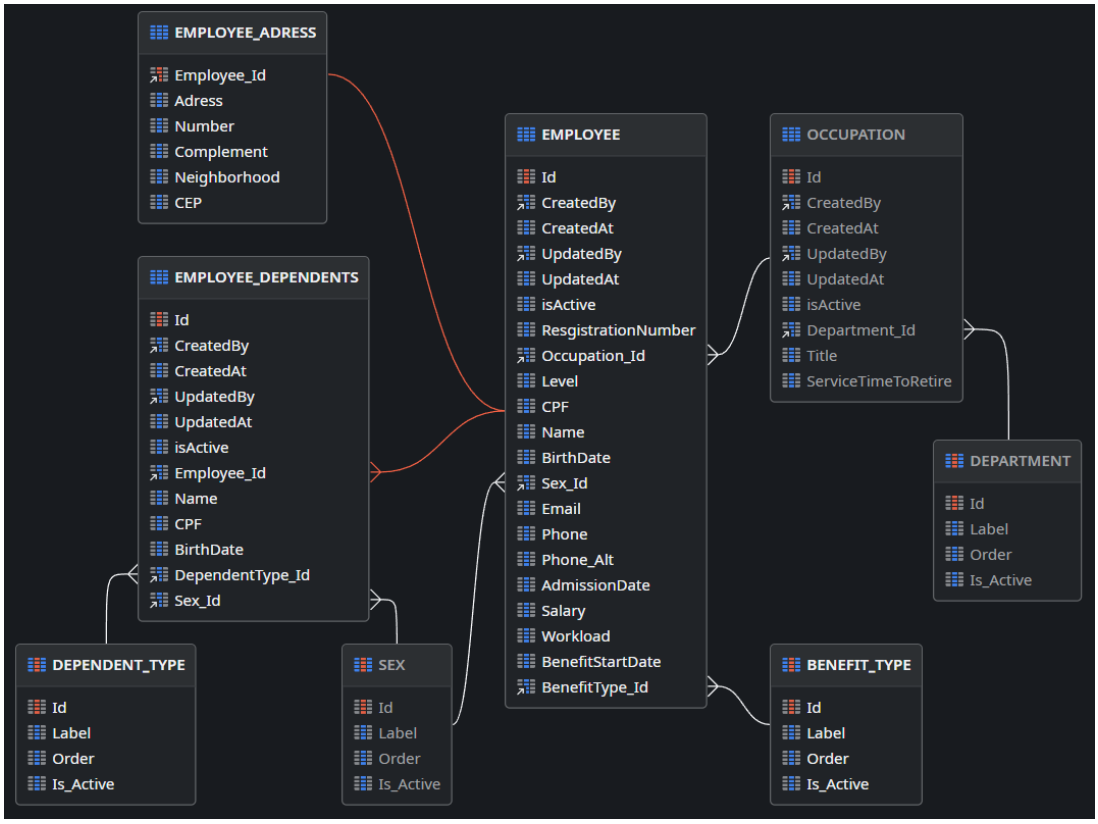
Fonte: Elaboração própria

3.2 MODELAGEM DO BANCO DE DADOS

Já em termos da base de dados da aplicação, a modelagem foi designada de forma que as tabelas que fazem sentido entre si, assim como suas respectivas ações de criação e eliminação, permanecessem agrupadas no mesmo módulo. Consequentemente, as tabelas relacionadas com cargos e suas secretarias municipais de lotação foram abrigadas no

ESPVC_Department_CS, as informações pessoais de ativos e beneficiários, assim como de seus dependentes, foram destinadas ao *ESPVC_Employee_CS*, enquanto o módulo *ESPVC_Common_CS* ficou encarregado de embarcar a tabela responsável pela designação do gênero da pessoa cadastrada, assim como para futuras tabelas que tenham abrangências mais comum e geral com o restante da aplicação. Assim como no caso dos módulos, aqui também são observadas convenções de nomenclatura para as tabelas e seus atributos. Na Figura 13 a seguir é apresentado um diagrama detalhado, elaborado dentro do próprio *Service Studio*, com todas as tabelas criadas para atender ao escopo inicial da aplicação, assim como todo o modelo relacional entre elas.

Figura 13 - Diagrama relacional do banco de dados da aplicação



Fonte: Elaboração própria

3.3 IMPORTAÇÃO DE DADOS VIA EXCEL

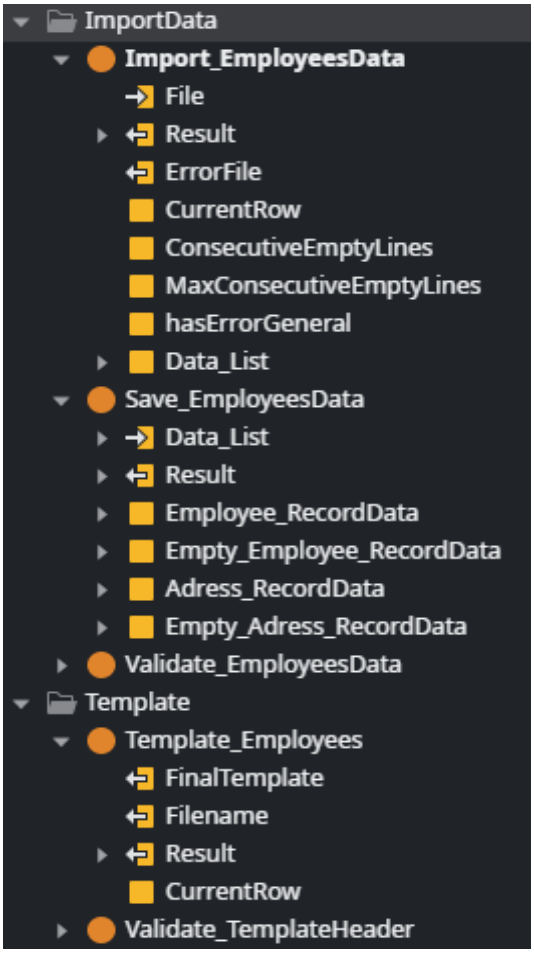
Dado o propósito de trazer facilidade e mais produtividade para o cotidiano de trabalho de um RPPS pequeno e com pouca estrutura, considerável parte dos esforços de construção desta aplicação foram orientados ao desenvolvimento de funcionalidade de importação de

dados através de arquivo Excel. Com tal dispositivo espera-se uma melhor curva de aprendizagem no que tange ao começo do uso do *software* por um RPPS de pequeno porte, em virtude de se tratar de um tipo de arquivo que já é costumeiramente utilizado para gerenciamento de bases de dados por institutos dentro destas características. Assim é promovida a utilização plena do programa de forma muito mais rápida, haja vista que a inserção dos dados dessa maneira se sucede em uma fração do tempo que uma equipe levaria para realizar o cadastro dos servidores um a um.

Primeiramente, foi construída manualmente duas planilhas simples, uma para os dados referentes aos cargos públicos do município e a outra para os dados dos servidores e beneficiários, para que sirvam de *template* para a incorporação desses dados na aplicação por seus usuários finais. Em ambas está presente um cabeçalho indicando os dados aceitos pelo programa, juntamente com o formato específico que os dados devem ser inseridos na planilha para que seja possível efetuar identificar os dados corretamente (por exemplo: CPF, datas, código postal etc.). Ademais, as células responsáveis pela inserção de dados que não são de digitação livre, como Cargo, Gênero ou Secretaria, já são disponibilizadas com listagem dentro das respectivas células com as opções possíveis de escolha. Por fim, ambos os arquivos foram incorporados à aplicação, estando presentes nos módulos do Core destinados à orquestração de regras de negócio: *ESPVC_Department_Eng* e *ESPVC_Employee_Eng*, respectivamente.

Para atingir esta finalidade, foi feito uso, na implementação de todo o fluxo lógico desta funcionalidade, da extensão *Advanced_Excel*, disponível abertamente no repositório *Forge*, que disponibiliza dezenas de ações e funções para leitura, gravação e manipulação de arquivos de Excel dentro das aplicações *OutSystems*. Para cada um dos *templates* de inserção de dados foram desenvolvidas cinco ações distintas, todas elas empregando-se de diversas funções da extensão acima. Na Figura 14 é possível visualizar resumidamente as ações que foram construídas para lidar com a importação especificamente dos funcionários do município, conforme elas ficaram configuradas na árvore lógica do módulo *ESPVC_Employee_Eng*.

Figura 14 - Ações de orquestração da importação dos servidores públicos via Excel



Fonte: Elaboração própria

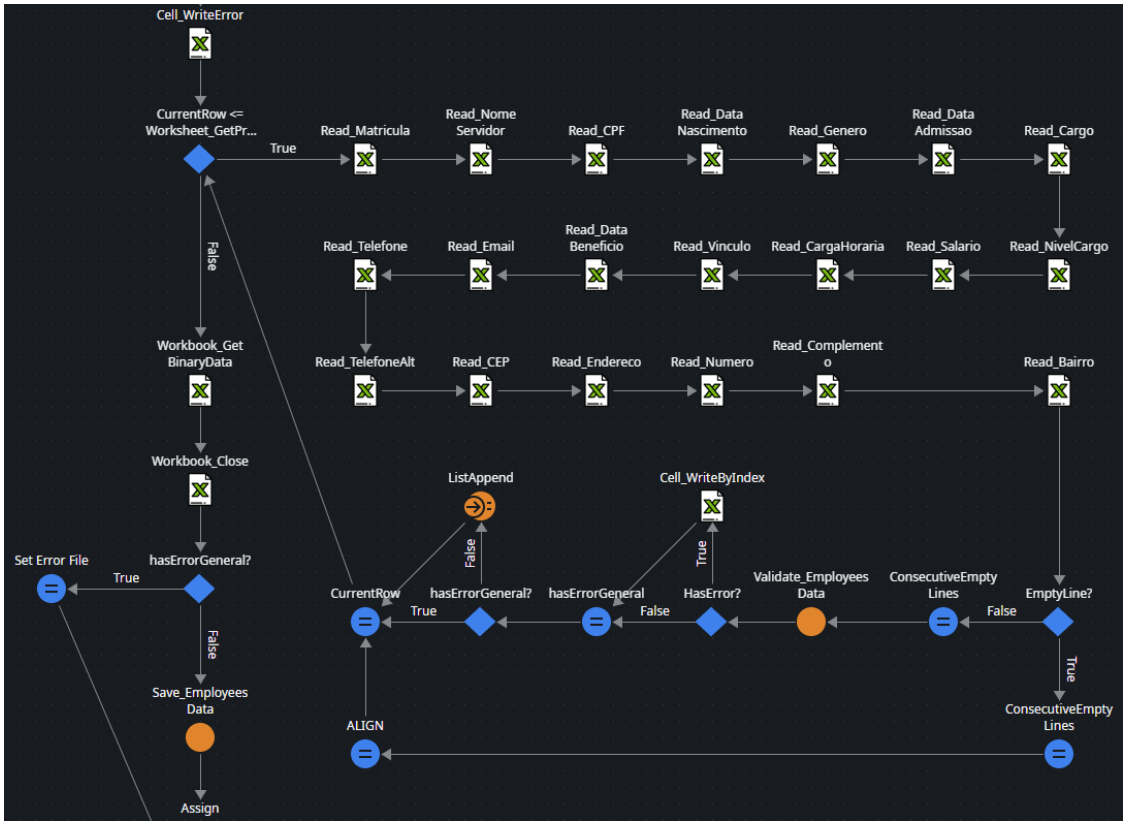
A primeira ação deste fluxo, *Template_Employees*, no caso dos servidores, fica encarregada de preparar o *template* para que o usuário da aplicação possa baixá-lo. No momento em que é solicitado ao programa o *download* do arquivo Excel para preenchimento, é corrida uma ação que realiza a população da lista de opções nas células que são necessárias (por exemplo, a lista de cargos já cadastrados para associar a novos servidores). Feito isso, é realizado o *download* do arquivo para o dispositivo do usuário, onde ele poderá efetuar o preenchimento das informações.

As demais ações são empregues no restante do fluxo de importação. Imediatamente após o utilizador carregar o ficheiro preenchido na aplicação, é executada ação que lê cada uma das células do cabeçalho do arquivo para confrontar com o que está presente no *template* na base de dados, assim se certificando não somente se o arquivo inserido no programa possui o formato Excel esperado, mas também que ele corresponde ao *template* esperado e não uma planilha

qualquer. Em caso de falha desta validação, o trâmite não é prosseguido e uma mensagem de erro é disparada visualmente para alertar o usuário do problema.

Validado o tipo de arquivo esperado, temos uma nova ação que entra em cena, com um ciclo lógico para efetuar leitura de cada linha da planilha que possua dados. Em seguida, estes dados são enviados para outra ação que encapsula diversas validações de cada um dos dados inseridos nas células, de modo a avaliar se eles estão no formato esperado, e para evitar conflitos ao inseri-los na base de dados. No caso de serem encontrados erros nos dados, a ação adiciona uma nova coluna ao fim do cabeçalho e efetua automaticamente o encerramento do fluxo de importação e o download de um novo arquivo, onde cada uma das linhas que possuem inconformidades pode ser identificada através das mensagens de erro, disponíveis na célula correspondente a esta nova coluna, tratadas cuidadosamente de forma que o utilizador possa saber exatamente o que foi inserido de errado e como corrigir para tentar novamente. A Figura 15 exemplifica uma parte do extenso fluxo lógico de leitura e validação dos dados inseridos no *template* de importação de servidores.

Figura 15 - Parte do fluxo lógico de importação de Servidores via Excel



Fonte: Elaboração própria

Por fim, após performadas todas as validações de conformidade dos dados inseridos em cada uma das linhas preenchidas do arquivo, é executada a quinta e última ação deste fluxo. Nela são organizados e salvos os dados nas tabelas correspondentes, e é finalizada com a emissão de mensagem de sucesso para o usuário, demonstrando que o fluxo correu até o fim sem problemas e os dados foram guardados.

3.4 DESENVOLVIMENTO DA EXPERIÊNCIA DE USUÁRIO

Toda a programação desempenhada no *Frontend* da aplicação foi realizada objetivando construir um programa direto, intuitivo e de fácil utilização, independentemente do nível de experiência dos usuários finais com aplicações *web*. Com isto em mente, o número de telas disponíveis na versão final foi planejado de forma que as funcionalidades do aplicativo ficassem independentes para manter uma visualização livre de poluição visual proveniente de muita informação amontoada, entretanto tentando ao máximo que essa diretriz não gerasse ao usuário uma sensação de aplicação vazia.

Isto posto, a interface de usuário ficou dividida em três blocos, cada um deles com um *link* fixo na barra do topo de todas as páginas, propiciando fácil acesso a qualquer uma delas a todo momento. O primeiro, constituído pela página inicial do programa, é a primeira tela com a qual o utilizador depara-se após efetuado o login. Ela funciona como um *Dashboard* de rápida visualização da situação dos servidores públicos do instituto, onde são apresentados gráficos de acordo com os dados registrados na plataforma.

O segundo bloco tem a incumbência de proporcionar a listagem, cadastro e edição dos cargos municipais e em que secretaria eles são lotados, e a todo momento oferecendo *links* para fácil navegação por toda a aplicação. Uma das telas fica encarregada de apresentar a lista completa desses cargos apresentando seu nome, a que secretaria ele está subordinado, e qual o tempo mínimo de serviço naquele cargo para que o trabalhador seja elegível à aposentadoria. A outra é designada para exibição de detalhes e edição de registro de cargo, seja de um existente ou adicionando um novo, providenciando um formulário ao utilizador inserir os dados como quiser e, ao salvar, são corridas validações para garantir que os dados preenchidos não possuem inconsistências.

De forma análoga ao descrito acima para os cargos, o terceiro bloco traz uma tela para visualização de listagem e outra para criação e edição de registros, mas no caso dos servidores

públicos. Todavia, a disposição dos elementos ao longo da interface do usuário difere ligeiramente do que o mencionado anterior, principalmente em se tratando do formulário para inserção e edição de dados, em razão de a tabela dos trabalhadores abranger uma gama bem maior de dados do que o observado no caso dos cargos, além de ser disponibilizada a capacidade de cadastrar dependentes para cada servidor público.

Para objetivar o funcionamento simples e intuitivo da aplicação, foi-se utilizado ao máximo as ferramentas e automatismos proporcionados pela *OutSystems* e, principalmente, todo o ganho em velocidade de processos assegurados pelo desenvolvimento em *Reactive*, resultando em um programa responsivo e com viabilidade de ser usufruído por meio de qualquer dispositivo, desde computadores até tablets e celulares. E tudo isso em conjunto com o uso de ferramentas disponibilizadas na *Forge* por outros desenvolvedores ao redor do mundo. Além da supracitada extensão *Advanced_Excel*, também foram incorporados ao aplicativo os componentes:

- *Input Mask Reactive* – Traz funções para serem aplicadas em campos de entrada de texto livre, aplicando a chamada "máscara" nos dados inseridos, para que assumam uma determinada formatação de maneira automática. Foi configurado para utilização nos campos de CPF, telefone e CEP, tornando mais conveniente o preenchimento do formulário por parte do usuário pois não precisa se preocupar em inserir traços ou pontos enquanto digita;
- *BR_Consult* – Dentre as inúmeras utilidades ofertadas por este componente, foi feito uso na aplicação de uma função encarregada de realizar buscas mediantes CEP. Desta forma, assim que o usuário preenche o campo do código postal no formulário, imediatamente os campos relacionados com o nome da rua e do bairro são preenchidos de acordo com os resultados apresentados pela função. Além disso, foram aplicadas validações para avisar imediatamente ao utilizador caso o valor inserido se trate de um CEP inválido;
- *Check CPF Number* – Esta extensão oferece uma valiosa função para verificação de CPF. Através dela, foi aplicado automatismo no campo referente a este dado para que ele seja verificado e, em se tratando de uma entrada inválida, o utilizador é alertado imediatamente;
- *SweetAlert* – Este se utiliza de códigos em *JavaScript* para gerar *popups* interativos com mensagens de alerta ou de confirmação, componente amplamente utilizado através de toda a aplicação. Com ele, foi possível municiar

> o aplicativo com comunicação simples e lúdica ao usuário, buscando gerar uma consciência situacional e transparência em todos os processos.

Para fins de demonstração dos resultados obtidos com este projeto, a aplicação foi alimentada com dados parte reais, parte fictícios. Foi tomado como exemplo para este estudo o município do Paulista, localizado na região metropolitana do Recife, onde, através de seu Portal da Transparência, fora obtida uma pequena amostra dos servidores públicos ativos no ente e utilizados para municiar o *software* os nomes de suas secretarias, os cargos, data de ingresso no funcionalismo público municipal e carga horária. Dados sensíveis dos trabalhadores, como CPF, endereço, e-mail e telefone, assim como os demais fundamentais para o funcionamento do sistema são fictícios e foram gerados aleatoriamente.

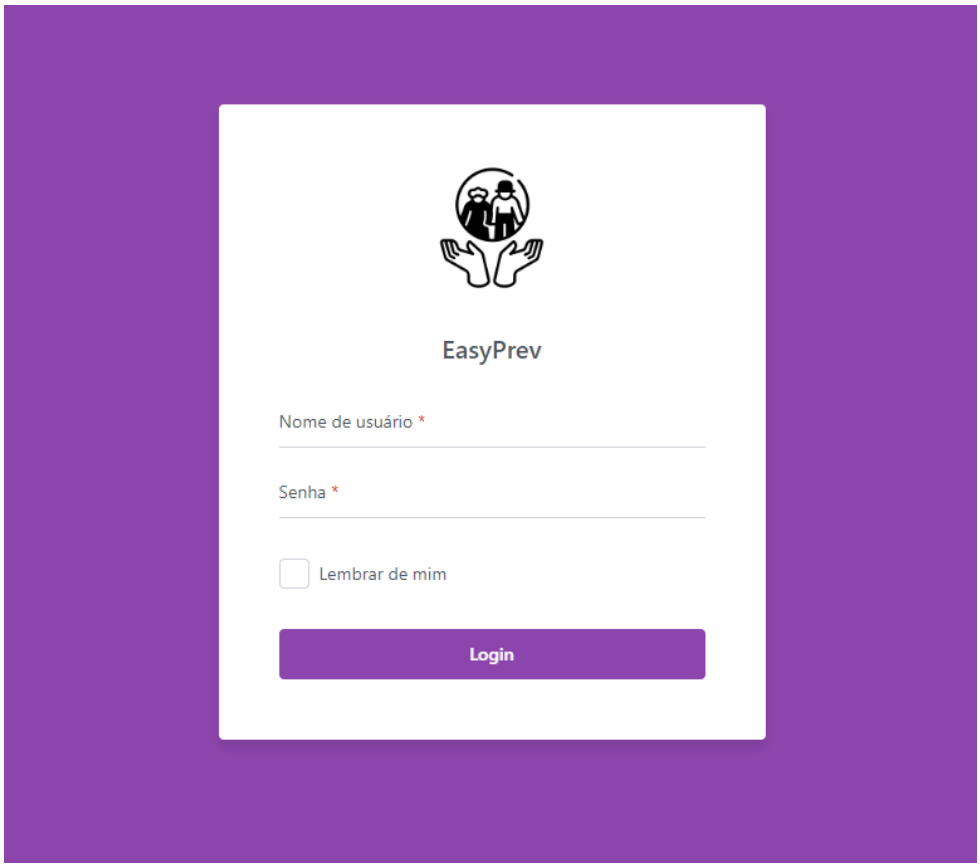
4. RESULTADOS

Neste capítulo serão apresentados os resultados obtidos neste trabalho após a finalização do desenvolvimento da aplicação *EasyPrev*. E, logo a seguir, são apresentadas informações acerca da disponibilidade deste *software* e como seu código-fonte pode ser obtido.

4.1 O *EASYPREV*

A aplicação pode ser acessada por qualquer navegador com acesso à internet, através de seu endereço *web*[1]. O primeiro contato do usuário é com a tela de login, disposta de maneira simples e direta, com os campos para inserção de suas credenciais, o nome da aplicação e sua logo, que foi obtida de maneira gratuita no *site Iconfinder*. Um exemplo de como a tela de login se apresenta pode ser observado na Figura 16 a seguir.

Figura 16 - Tela de login do *EasyPrev*

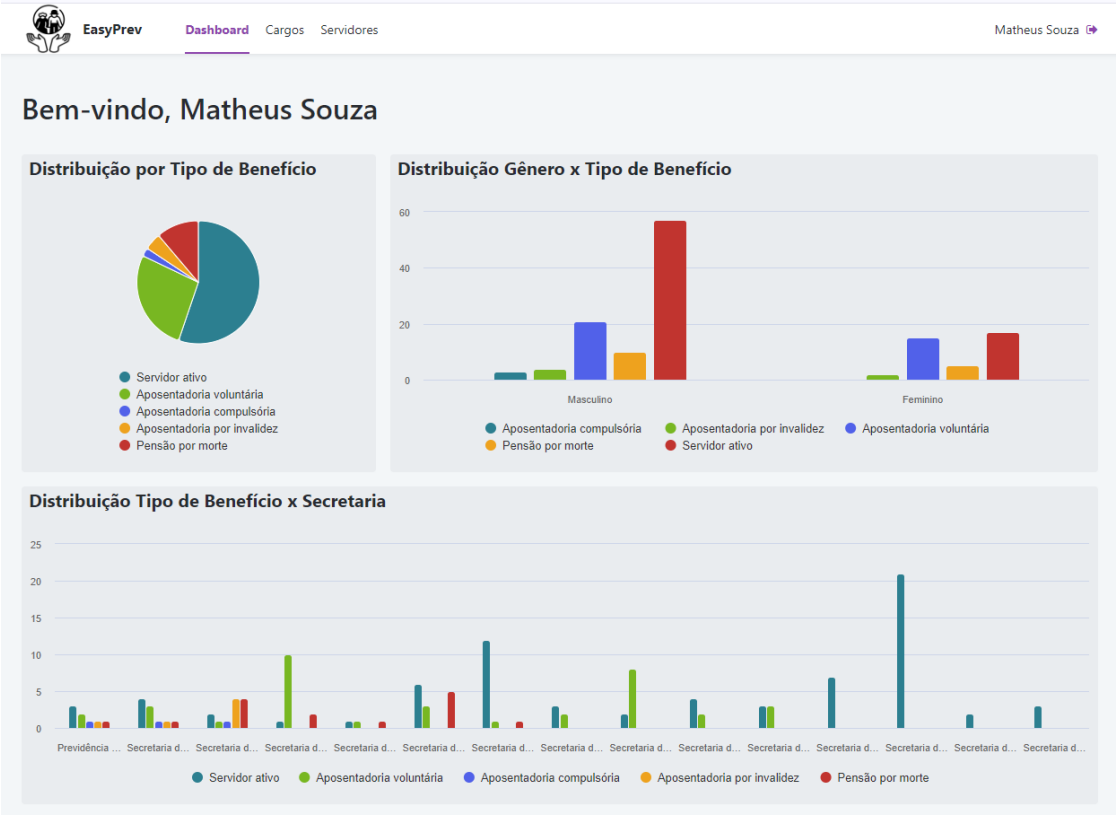


Fonte: Elaboração própria

[1] Acessível através do endereço eletrônico: <https://personal-xtk2tim6.outsystemscloud.com/EasyPrev>.

No momento em que é concedido acesso à aplicação, a tela reproduz uma transição e apresenta ao utilizador o *Dashboard*, página principal do *EasyPrev*. Aqui é montado painel para recepcionar o usuário e apresentar alguns gráficos ao utilizador que evidenciam, de forma suscinta, interativa e organizada a situação atual da base de dados do instituto. E isto pode ser averiguado na Figura 17.

Figura 17 – *Dashboard* do *EasyPrev*



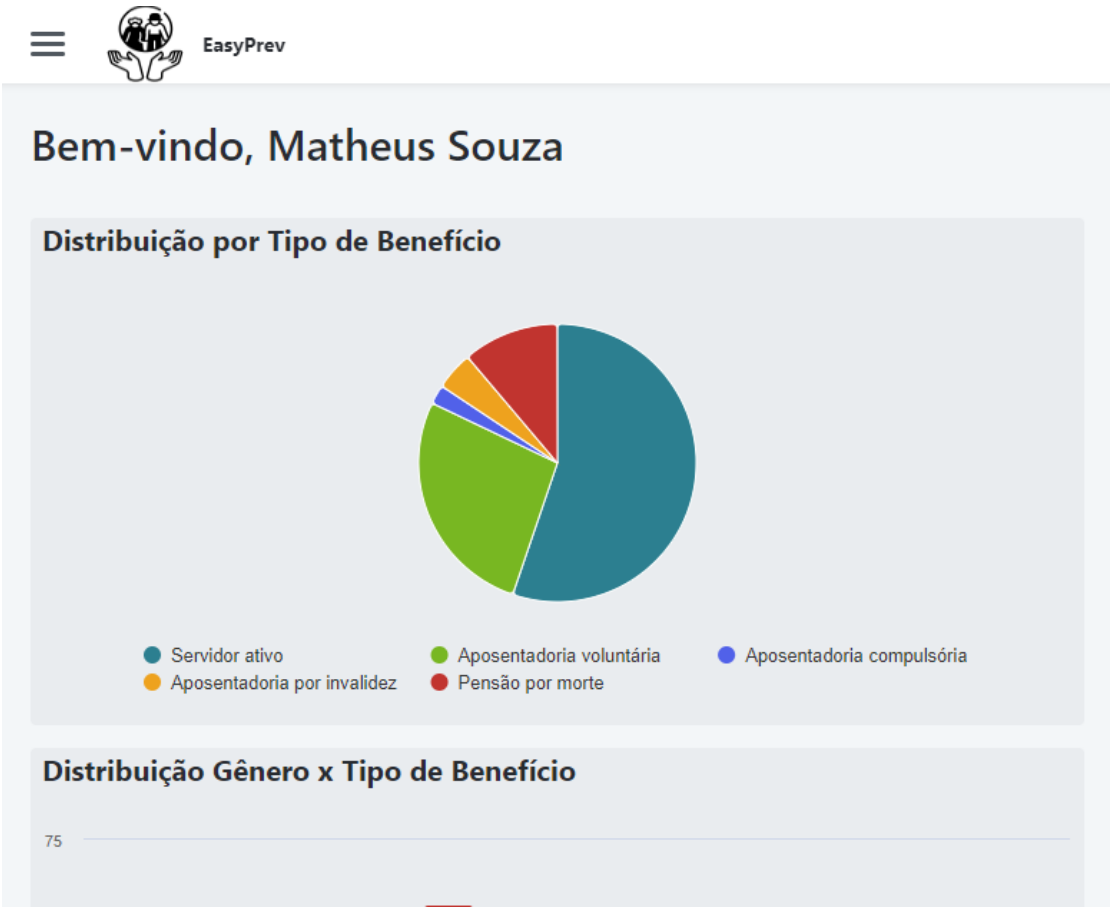
Fonte: Elaboração própria

O primeiro gráfico, de setores, chamado “Distribuição por Tipo de Benefício”, mostra o número de funcionários públicos cadastrados no programa de acordo com o tipo de benefício, ou seja, Servidor Ativo, Aposentadoria Voluntária, Aposentadoria Compulsória, Aposentadoria por Invalidez ou Pensão por Morte. O segundo, “Distribuição Gênero x Tipo de Benefício”, é um gráfico de colunas mostrando a quantidade de homens e mulheres também de acordo com seu tipo de benefício. Já o terceiro e último gráfico fica encarregado de mostrar a “Distribuição Tipo de Benefício x Secretaria” na página inicial e, também configurado na visualização de

colunas, evidencia a quantidade de servidores separados de acordo com o tipo de benefício ao qual eles estão vinculados para cada um dos departamentos ou secretarias que esses trabalhadores estejam lotados.

Além do dinamismo na renderização das telas e de carregamento de dados de forma assíncrona, outro grande benefício trazido para a aplicação por ter sido desenvolvida sob a metodologia *Reactive* foi a responsividade da interface de usuário, que é capaz de automaticamente se adaptar ao tamanho de tela disponível e sem necessidade de recarregamentos, independente de qual seja o dispositivo usado. Na Figura 18 temos um exemplo do *EasyPrev* sendo executado no computador em uma janela de navegador propositalmente bem reduzida e, instantaneamente, os elementos em tela são reorganizados e dispostos de maneira a aproveitar o espaço disponível, sem prejudicar a experiência do usuário. Inclusive, com o menu principal superior sendo prontamente transformado no chamado “menu sanduíche”, que abre uma barra lateral mostrando os links do menu principal, assumindo um comportamento similar ao experienciado ao se executar um aplicativo para dispositivo móvel.

Figura 18 - Exemplo de responsividade do *Dashboard* sendo executado em tela reduzida



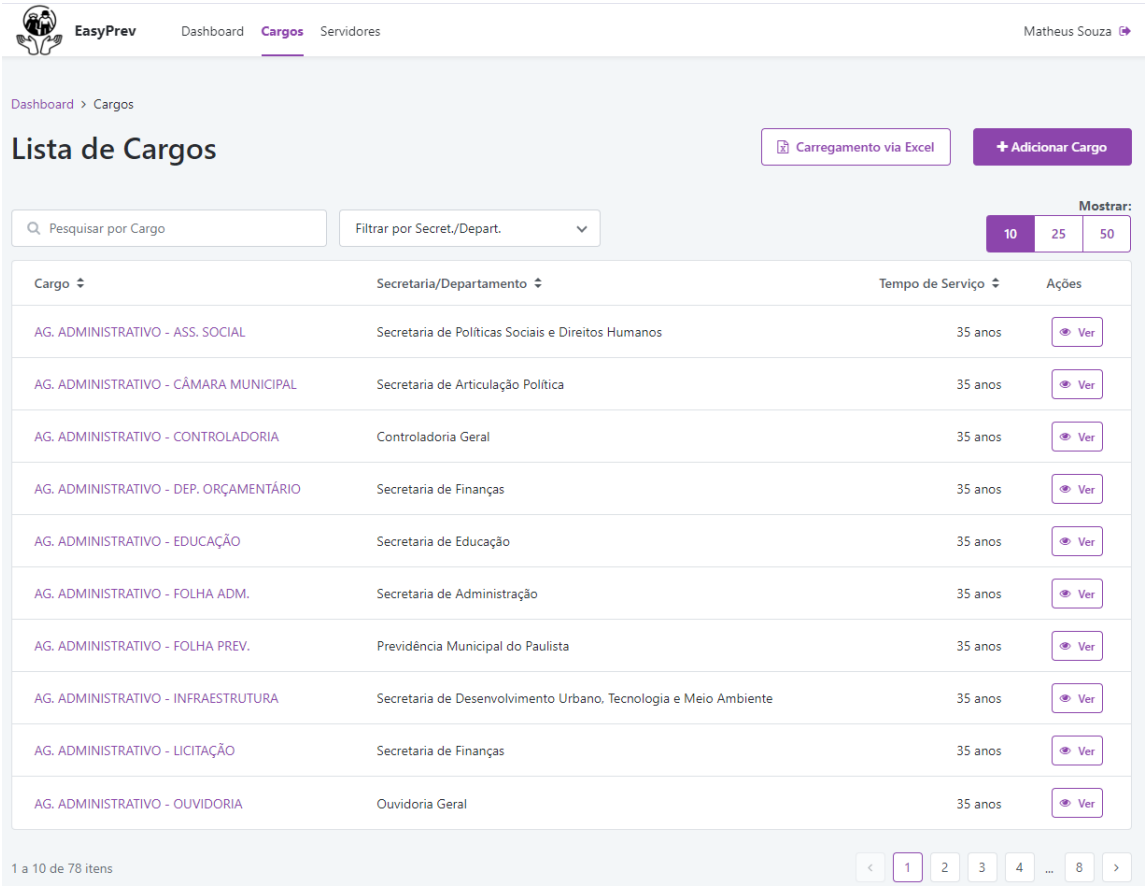
Fonte: Elaboração própria

Ao acessar a página Cargos, o utilizador depara-se com listagem de todos os cargos cadastrados no programa. A lista traz por padrão 10 resultados por página, podendo ser alterado para 25 ou 50, conforme preferência do usuário, graças aos botões disponibilizados no canto superior direito da listagem que permite esta alteração e atualiza os resultados instantaneamente.

Também localizados logo acima da lista, estão dispostos campos para filtragem automática dos registros a partir do momento que qualquer um deles sofra alguma alteração, atualizando a visualização da listagem de forma versátil para que sejam exibidos especificamente os resultados que atendam às necessidades do utilizador a acessar. Um destes campos permite entrada de texto livre para pesquisar sobre o nome do cargo, enquanto o outro traz um menu listando as secretarias do ente municipal para serem mostrados apenas os cargos com aquela lotação. Além do mais, o utilizador em sessão pode alterar a ordenação de como os

registros são apresentados na lista apenas clicando no cabeçalho de alguma coluna para ordenar por aquele atributo. É possível contemplar a estrutura da página de cargos na Figura 19.

Figura 19 - Página de listagem de Cargos

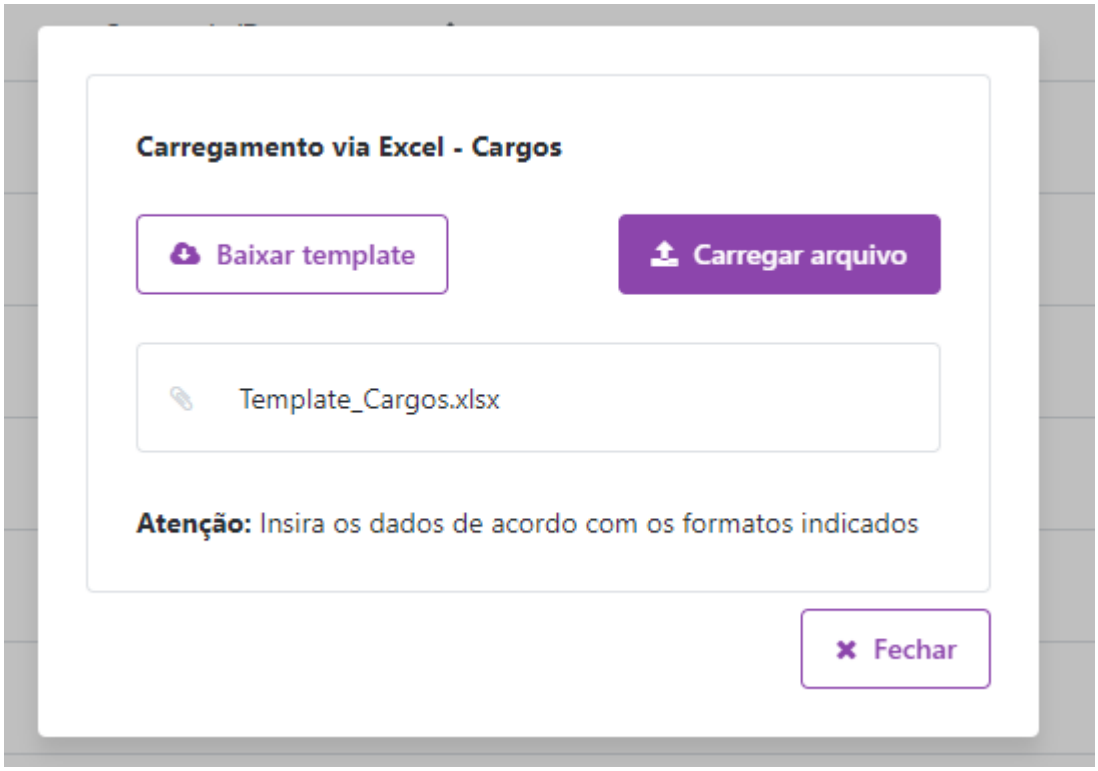


Fonte: Elaboração própria

Disponível apenas para os utilizadores da aplicação que possuam um perfil de administrador, por se tratar de uma funcionalidade sensível e complexa, temos o botão “Carregamento via Excel”. Mediante seu acionamento, é aberta janela *popup* onde o usuário em sessão pode baixar o *template* Excel para ser preenchido com os dados a importar, junto ao mecanismo de carregamento do arquivo de seu dispositivo e o botão “Carregar arquivo”, que efetuará a leitura e gravação dos dados, conforme fluxo lógico já elucidado em detalhes anteriormente. Enquanto o carregamento é performado, o botão “Carregar arquivo” fica indisponível e com um ícone girando, demonstrando visualmente para o utilizador que a aplicação está fazendo seu trabalho nos bastidores e para evitar que, na incerteza se está

funcionando ou não, haja novos cliques. Detalhes de como se apresenta este *popup* de importação de dados podem ser visualizados na Figura 20.

Figura 20 - *Popup* de importação de dados via Excel



Fonte: Elaboração própria

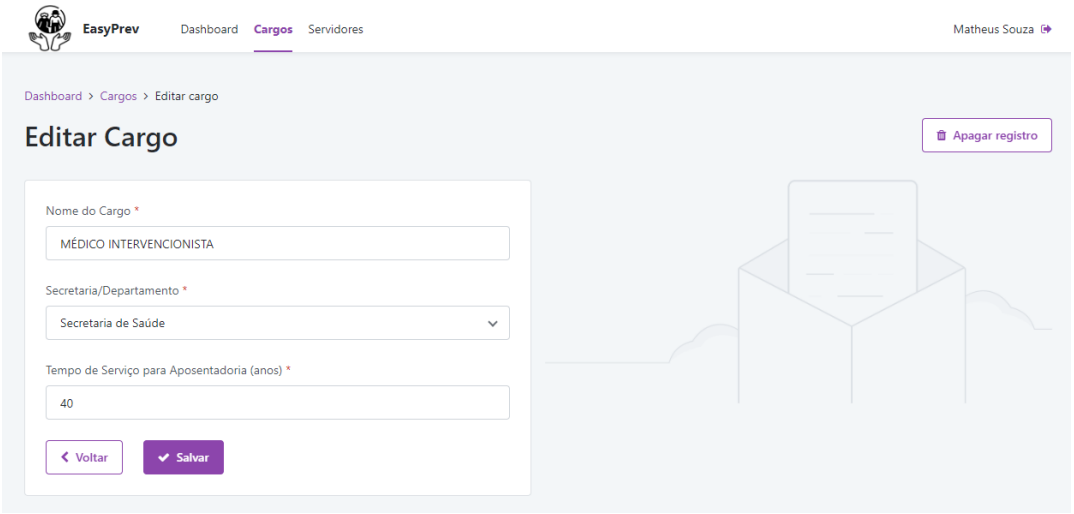
Em conjunto com a tela da listagem, foi desenvolvida uma página de detalhes para os cargos, onde os respectivos dados podem ser acessados individualmente e com a possibilidade de gravar alterações em base de dados caso seja necessário. Foi implementada uma transição em que a página atual desliza da direita para a esquerda ao aceder a esta página, e uma transição inversa para quando voltamos dela para a listagem ou ao gravar os detalhes do cargo que está a ser consultado, para demonstrar de forma lúdica e visual para o usuário final o fluxo de trabalho entre ambas as telas no que tange aos cargos.

Essa página pode ser acessada ao clicar no nome do cargo na listagem, ou no botão “Ver”, entrando assim no modo edição do cargo em questão, ou então clicando no botão “Adicionar Cargo” presente na tela de listagem ao nível do título. Neste último caso, o usuário é redirecionado para a mesma tela de detalhes de cargo e com a mesma transição supracitada, mas, dessa vez, com o formulário em branco para preenchimento e criação de um novo cargo

na base de dados. A partir do momento que o usuário tentar salvar os dados, assim como pode-se perceber na funcionalidade de importação de dados por Excel, o botão de salvar fica indisponível como um indicativo visual para o utilizador de que a ação está correndo, são percorridas validações dos dados que foram inseridos nos campos para evitar erros e, caso alguma inconformidade seja detectada, a lógica de gravação dos dados é interrompida e os erros são explicitados em vermelho na tela, com mensagens claras para que o utilizador tenha plena sabedoria do que está errado e corrija antes de tentar prosseguir. Também é disponibilizado botão “Apagar registro” apenas para usuários administradores, assim como é feito para a importação de dados, por se tratar de uma funcionalidade sensível e para evitar perda de dados por engano.

Caso o utilizador tente navegar para qualquer outra página enquanto está nesta página de detalhe, seja criando um registro ou durante a edição de algum existente, lhe é mostrado uma janela *popup* de confirmação, alertando que está a tentar sair da janela atual e que, ao confirmar o prosseguimento, todos os dados informados em tela e ainda não salvos serão perdidos. Um exemplo da tela de detalhes de um cargo pode ser examinado na Figura 21.

Figura 21 - Tela de criação/edição de Cargo



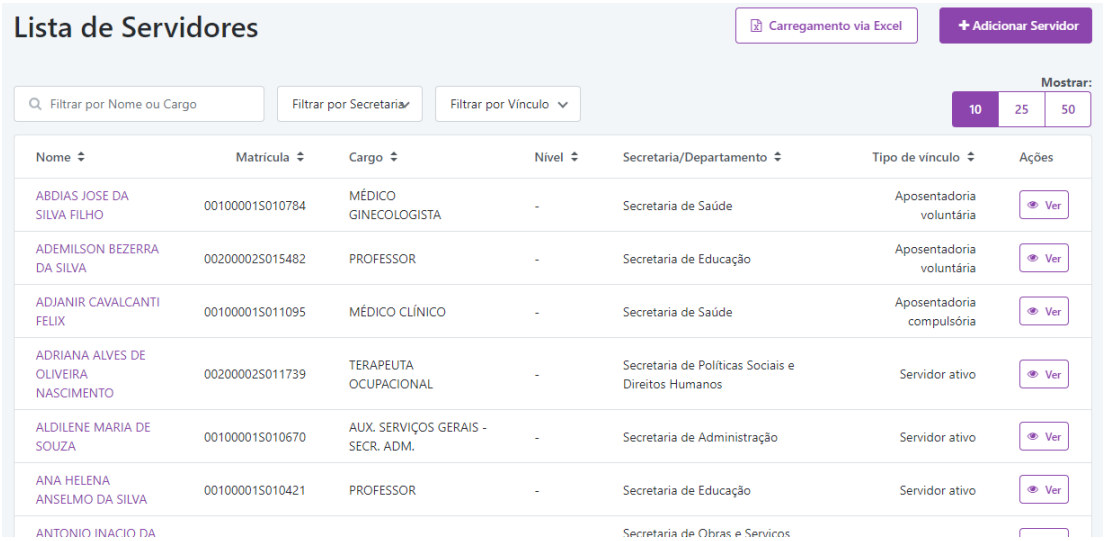
Fonte: Elaboração própria

As páginas destinadas às informações dos servidores públicos possuem funcionamento e dinâmicas análogas ao observado para as dos cargos: listagem com os principais detalhes, paginação com um número padrão de 10 resultados por página podendo ser alterado mediante ação do usuário, *link* e botão “Ver” dentro da listagem para visualizar e/ou editar detalhes de

um determinado registro, capacidade de mudar a ordenação da lista, campos de pesquisa automatizados, botão “Carregamento via Excel” disponível para perfil administrador e transição entre telas similar para demonstrar intuitivamente o fluxo de trabalho.

Como principais diferenças temos que os registros dos trabalhadores abarcam muito mais dados do que o observado no caso dos cargos, portanto a listagem possui mais campos, mas se concentrando em alguns mais importantes para evitar deixá-la bagunçada e confusa. No caso dos filtros, o campo de texto livre permite pesquisar pelo nome do servidor ou do cargo que ele ocupa, além de outros dois campos com lista para filtrar por secretaria de lotação do trabalhador ou por tipo de vínculo (Servidor Ativo, Aposentado etc.). O *popup* de importação via Excel possui o mesmo comportamento, com exceção para o *template* que é baixado e que ele espera receber preenchido de volta para leitura. Pode-se averiguar um exemplo de como se dispõe a tela da listagem de empregados na Figura 22.

Figura 22 - Tela de listagem de Servidores

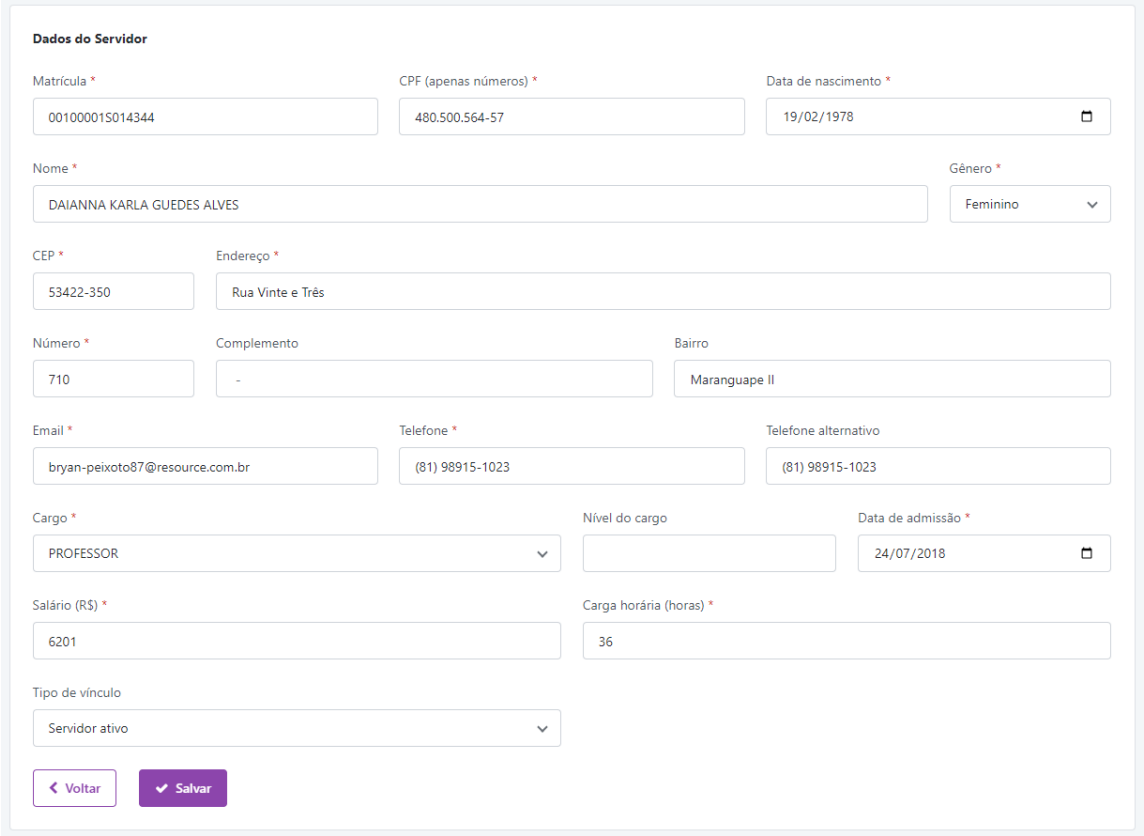


Fonte: Elaboração própria

Já na tela de detalhes dos servidores é onde vemos as maiores diferenças em relação à dos cargos. Por conta da quantidade de dados, o formulário aqui é bem maior e ocupa todo o espaço útil horizontal da tela. Ao aceder ao detalhe de um trabalhador, as entradas de dados ficam inicialmente indisponíveis, permitindo apenas a visualização dos dados e exigindo que o utilizador clique no botão “Editar” para que possa efetuar mudanças, ficando também disponível o botão “Apagar registro” caso o usuário em sessão seja um administrador. Iniciando

o modo de edição, o formulário se atualiza com os campos agora permitindo edições, e se tornando acessível o botão “Salvar” logo ao fim que, similarmente ao exposto previamente, se torna indisponível ao clique, performando validações dos dados inseridos, apontando os erros caso sejam encontrados, ou efetuando a gravação do registro caso contrário. Analogamente, mediante qualquer tentativa de navegação para outra tela enquanto esta se encontra no modo de edição, será disparada mensagem de confirmação, alertando que dados não salvos podem ser perdidos. O formulário completo dos dados dos servidores públicos pode ser apurado na Figura 23.

Figura 23 - Formulário de criação/edição de empregados



Fonte: Elaboração própria

Logo abaixo deste formulário, fica presente uma tabela listando os dependentes cadastrados atrelados àquele servidor público. A opção “Novo Dependente” só fica disponível para utilização após já ter sido criado o registro do trabalhador e quando o modo edição da tela está ativado. Acionando este botão, é apresentado um *popup* para o utilizador, para preenchimento dos dados pessoais referentes a esse novo dependente, assim como ele é aberto

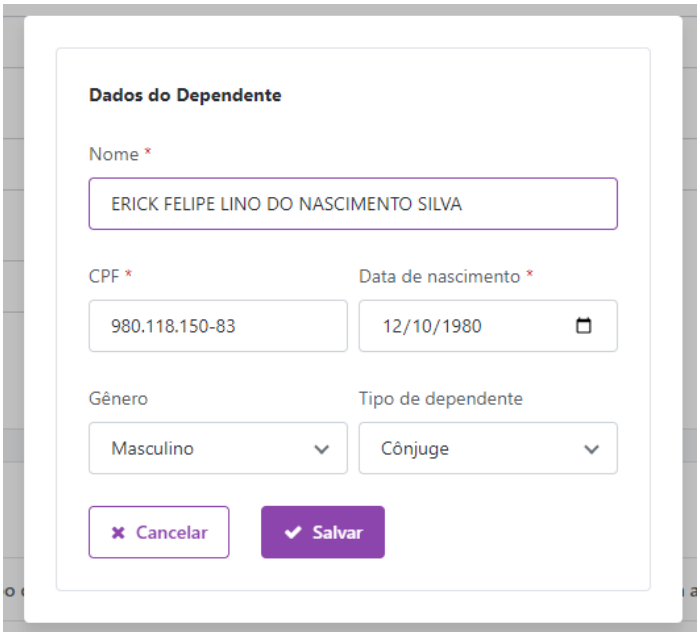
quando se clica no botão “Editar” na tabela, preenchendo este novo formulário com as informações existentes daquele dependente para que possam ser consultadas em detalhes e editadas. Bem como acontece em toda a aplicação, aqui o botão apagar igualmente só é mostrado para os usuários administradores. A Figura 24 exemplifica a visualização da lista de dependentes.

Figura 24 - Tabela de dependentes de um servidor

Fonte: Elaboração própria

Finalmente, também são realizadas validações dos dados introduzidos e com as devidas máscaras para aplicar automaticamente a formatação necessária. E, para além disso, aqui é validado também o tipo de dependente que está a ser adicionado ou editado, pois a aplicação não permite que um servidor público tenha mais do que um cônjuge vinculado a si. Na Figura 25, tal *popup* de criação e edição de dependentes pode ser examinado.

Figura 25 - *Popup* de criação e edição de dependentes



Fonte: Elaboração própria

4.2 DISPONIBILIDADE DA APLICAÇÃO

O *software* resultante deste projeto e seu código-fonte foram disponibilizados no repositório *GitHub*, podendo ser acessado através de seu endereço eletrônico[2]. Deixando o código aberto à comunidade, outros desenvolvedores podem criar interesse em testar a aplicação, podendo resultar na implementação de melhorias, novas funcionalidades e expansão de seus propósitos.

[2] Disponível em <https://github.com/Matheus-FSouza/easyprev>.

5. CONSIDERAÇÕES FINAIS

Neste trabalho, abordou-se o histórico de construção e formação do sistema previdenciário brasileiro, chegando até a forma como ele hoje é estruturado, visando mais detalhadamente o funcionamento dos Regimes Próprios de Previdência Social, e como a realidade de seu trabalho diário pode ser muito prejudicado quando se trata de um instituto ligado um município de pequeno porte, esses que, no caso do estado de Pernambuco, são maioria das entidades federativas que possuem esse tipo de regime. Na falta de acesso a profissionais devidamente qualificados, ou, até mesmo, na falta de um quadro de servidores dedicados ao instituto em quantidade adequada, o desafio para a conquista dos objetivos do fundo de manter seu Equilíbrio Atuarial e, no limite, cumprir com os benefícios futuros de seus segurados, torna-se ainda mais árduo.

Motivado a contribuir com a melhoria da qualidade do trabalho de institutos de previdência nesse tipo de cenário, foi apresentado projeto de desenvolvimento de um *software*, com o propósito de ser leve, de simples funcionamento e com boa curva de aprendizado, para que um RPPS pequeno possa abandonar outras ferramentas mais mecanizadas, como o Excel, para efetuar o gerenciamento de suas bases de dados de servidores ativos e inativos, com o mínimo de impacto possível.

Para atingir tal desígnio, foi apresentada a tecnologia *low-code*, e como ela pode contribuir para o desenvolvimento de programas com significativa redução de custos e de tempo, através de ferramentas de baixo código permitindo uma programação mais visual e interativa, diferentemente das linguagens mais tradicionais. Consequentemente, foi retratada em detalhes a plataforma *OutSystems*, já tradicional e consolidada no mercado de desenvolvimento de *software* por meio desse tipo de tecnologia, e como seu funcionamento e estruturação poderiam trazer benefícios notórios para o progresso satisfatório do projeto.

Então, foi proposto o *EasyPrev*, aplicação baseada em *web*, assim sendo, não exigindo a instalação de um programa no aparelho em que ele será utilizado, bastando apenas acessar seu endereço eletrônico através de qualquer navegador, em qualquer tipo de dispositivo.

Seguidamente, foram apresentadas em detalhes suas funcionalidades, de planejamento e *design* próprios do autor, para a aplicação e de que maneira foi idealizada a sua arquitetura, a estruturação de suas telas e experiência de usuário em geral, com uma maior ênfase no desenvolvimento dos instrumentos para importação de dados para a base de dados do programa através de arquivos Excel. Este mecanismo foi considerado como a funcionalidade mais

essencial do programa, contribuindo para que um RPPS que comece a utilizar o *EasyPrev* tenha o menor impacto inicial possível. Espera-se que a aplicação e suas funcionalidades consigam possibilitar que o trabalho diário dos colaboradores dos pequenos institutos de previdência municipais.

Vale salientar que o tempo despendido pelo autor, desde o início do planejamento da arquitetura e das funcionalidades da aplicação até o ponto de estar completamente desenvolvida e funcionando como se encontra, foi de pouco menos um mês. Portanto, demonstrou-se na prática os benefícios que o desenvolvimento de *software* fazendo uso de uma plataforma *low-code*, como é o caso da *OutSystems*, já que, caso o *EasyPrev* fosse concebido através tecnologias mais tradicionais de desenvolvimento de aplicativos, seria necessário ter mais programadores para desempenhar este trabalho e, definitivamente, bem mais tempo de projeto do que o que foi utilizado.

Finalmente, ficam como sugestões de trabalhos futuros para o *EasyPrev*, após adquiridas licenças de uso da plataforma para que a aplicação tenha uma Fábrica bem estruturada por trás, e para além da parte elementar de gestão de bases de dados já abordada, a expansão das funcionalidades ofertadas pelo aplicativo, principalmente no que tange à parte financeira de um RPPS, como: demonstrativo rápido do montante atual de benefícios a ser pago pelo fundo, ferramentas para previsão de evolução das despesas com benefícios programados com a identificação automática por parte da aplicação de servidores que estejam na iminência de atingir seu tempo mínimo de contribuição para adquirir elegibilidade de se aposentar, e possibilidade de geração de relatórios e demonstrativos para satisfazer requisições de auditorias.

REFERÊNCIAS

AGUIAR, J. N. **Portuguese Unicorns: Where else to go but the U.S. – A tale of two unicorns**. Dissertação—Lisboa: Universidade Católica Portuguesa, jun. 2022.

AL ALAMIN, M. A. *et al*. **An empirical study of developer discussions on low-code software development challenges**. Proceedings - 2021 IEEE/ACM 18th International Conference on Mining Software Repositories, MSR 2021. **Anais**...Institute of Electrical and Electronics Engineers Inc., 1 maio 2021.

AZOFF, M. Ovum Decision Matrix: Selecting an Enterprise Mobile Application Development Platform, 2018-19. **Ovum**, 13 abr. 2018.

BOCK, A. C.; FRANK, U. Low-Code Platform. **Business and Information Systems Engineering**, v. 63, n. 6, p. 733–740, 1 dez. 2021.

BOGONI, N. M.; FERNANDES, F. C. Gestão de risco nas atividades de investimento dos Regimes Próprios de Previdência Social (RPPS) dos municípios do Estado do Rio Grande do Sul. **Revista Eletrônica de Administração (Porto Alegre)**, v. 17, n. 1, p. 117–148, 1 abr. 2011.

BRASIL. **Constituição da República Federativa do Brasil**. [s.l: s.n.].

BRIGUET, M. R. C. Regimes Próprios: aspectos relevantes. **Associação Brasileira de Instituições de Previdência Estaduais e Municipais**, v. 14, p. 421–437, 2020.

BUCAIONI, A.; CICCHETTI, A.; CICCOZZI, F. Modelling in low-code development: a multi-vocal systematic review. **Software and Systems Modeling**, v. 21, n. 5, p. 1959–1981, 1 out. 2022.

CABOT, J. **Positioning of the low-code movement within the field of model-driven engineering**. Proceedings - 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, MODELS-C 2020 - Companion Proceedings. **Anais**...Association for Computing Machinery, Inc, 16 out. 2020.

CORRÊA, C. S.; QUEIROZ, B. L.; RIBEIRO, A. J. F. **Tamanho populacional e custeio previdenciário: como variações aleatórias afetam o risco de solvência de RPPS municipais**. [s.l: s.n.]. Disponível em: <www.cedeplar.ufmg.br>.

DAMASCENO, A. T.; CARVALHO, J. V. DE F. Assessment of the new investment limits for assets of Social Security Regimes for Public Servants established by Resolution CMN 3,922/2010. **Revista Brasileira de Gestão de Negócios**, v. 23, n. 4, p. 728–743, 2021.

DI SIPIO, C.; DI RUSCIO, D.; NGUYEN, P. T. **Democratizing the development of recommender systems by means of low-code platforms**. Proceedings - 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, MODELS-C 2020 - Companion Proceedings. **Anais**...Association for Computing Machinery, Inc, 16 out. 2020.

DI RUSCIO, D. *et al*. Low-code development and model-driven engineering: Two sides of the same coin? **Software and Systems Modeling**, v. 21, n. 2, p. 437–446, 1 abr. 2022.

GARTNER. **Gartner Forecasts Worldwide Low-Code Development Technologies Market to Grow 20% in 2023**. Disponível em: <https://www.gartner.com/en/newsroom/press-releases/2022-12-13-gartner-forecasts-

worldwide-low-code-development-technologies-market-to-grow-20-percent-in-2023>. Acesso em: 14 dez. 2023.

GOLOVIN, D. **OutSystems as a Rapid Application Development Platform for Mobile and Web Applications**. [s.l.] Lahti University of Applied Sciences, 2017.

GONÇALVES, L. A.; MATOS, M.; NOGUEIRA, N. G. **Os 100 Anos da Previdência Social**. Brasília: Ministério da Previdência Social, 2022.

GUEDES, L. P. O princípio do equilíbrio econômico e financeiro no Regime Geral de Previdência Social. **Revista da AGU**, v. 15, n. 03, p. 241–262, 7 abr. 2017.

GUSHIKEN, L. *et al*. Regime Próprio de Previdência dos Servidores: Como Implementar? Uma Visão Prática E Teórica. **Coleção Previdência Social**, v. 17, 2002.

IBRAHIM, F. Z. **A Previdência Social no estado contemporâneo: Fundamentos, financiamento e regulação**. Rio de Janeiro: Universidade do Estado do Rio de Janeiro, 2011.

KÄSS, S.; STRAHRINGER, S.; WESTNER, M. **A Multiple Mini Case Study on the Adoption of Low Code Development Platforms in Work Systems**. , 2017.

LUO, Y. *et al*. **Characteristics and challenges of low-code development: The practitioners perspective**. International Symposium on Empirical Software Engineering and Measurement. **Anais**...IEEE Computer Society, 11 out. 2021.

MARTINS, R. *et al*. An overview on how to develop a low-code application using OutSystems. **2020 International Conference on Smart Technologies in Computing, Electrical and Electronics (ICSTCEE)**, p. 395–401, 2020.

MCKENDRICK, J. **The Rise Of The Empowered Citizen Developer: 2017 Low-Code Adoption Survey**. [s.l: s.n.]. Disponível em: <www.unisphereresearch.com.>.

MERRIAM-WEBSTER. **Unicorn Definition & Meaning**. Disponível em: <https://www.merriam-webster.com/dictionary/unicorn>. Acesso em: 24 dez. 2023.

MINISTÉRIO DA PREVIDÊNCIA SOCIAL. **Breve histórico — Instituto Nacional do Seguro Social - INSS**. Disponível em: <https://www.gov.br/inss/pt-br/acesso-a-informacao/institucional/breve-historico>. Acesso em: 19 fev. 2024.

MINISTÉRIO DA PREVIDÊNCIA SOCIAL. **Painel Estatístico da Previdência - Regimes Próprios de Previdência Social - Regime Previdenciário dos Entes**. Disponível em: <https://www.gov.br/previdencia/pt-br/assuntos/estatisticas-da-previdencia/painel-estatistico-da-previdencia/regimes-proprios-de-previdencia-social-1/regime-previdenciario-dos-entes-federativos>. Acesso em: 13 dez. 2023.

OUTSYSTEMS. **Gartner Magic Quadrant for Low-Code Application Platforms 2023** . Disponível em: <https://www.outsystems.com/1/low-code-application-platforms-gartner-/>. Acesso em: 20 dez. 2023a.

OUTSYSTEMS. **About OutSystems: Powering Innovation with Low-Code**. Disponível em: <https://www.outsystems.com/company/>. Acesso em: 20 dez. 2023b.

OUTSYSTEMS. **OutSystems Training & Certifications**. Disponível em: <https://www.outsystems.com/evaluation-guide/getting-started-with-outsystems/training/>. Acesso em: 20 dez. 2023c.

OUTSYSTEMS. **Learning Curve in the OutSystems Platform**. Disponível em: <https://www.outsystems.com/evaluation-guide/getting-started-with-outsystems/learning-curve/>. Acesso em: 20 dez. 2023d.

OUTSYSTEMS. **Security Overview**. Disponível em: <https://www.outsystems.com/evaluation-guide/security/>. Acesso em: 20 dez. 2023e.

OUTSYSTEMS. **Security for Application Development**. Disponível em: <https://www.outsystems.com/evaluation-guide/security/applications/>. Acesso em: 20 dez. 2023f.

OUTSYSTEMS. **Security Certifications, Governance and Compliance**. Disponível em: <https://www.outsystems.com/evaluation-guide/security/certifications/>. Acesso em: 20 dez. 2023g.

OUTSYSTEMS. **OutSystems pricing: Low code platform pricing**. Disponível em: <https://www.outsystems.com/pricing-and-editions/>. Acesso em: 20 dez. 2023h.

OUTSYSTEMS. **OutSystems Partner Program**. Disponível em: <https://www.outsystems.com/partners/>. Acesso em: 20 dez. 2023i.

OUTSYSTEMS. **Find your Partner**. Disponível em: <https://www.outsystems.com/partners/list/>. Acesso em: 20 dez. 2023j.

OUTSYSTEMS. **Forum & Community Discussions**. Disponível em: <https://www.outsystems.com/forums/>. Acesso em: 20 dez. 2023k.

OUTSYSTEMS. **Search Forge: assets**. Disponível em: <https://www.outsystems.com/forge/>. Acesso em: 20 dez. 2023l.

OUTSYSTEMS. **The Power of OutSystems AI**. Disponível em: <https://www.outsystems.com/evaluation-guide/ai/outsystems-ai/>. Acesso em: 20 dez. 2023m.

OUTSYSTEMS. **AI-Infusion in Applications**. Disponível em: <https://www.outsystems.com/evaluation-guide/ai/infusion/>. Acesso em: 20 dez. 2023n.

OUTSYSTEMS. **AI for App Development and Management**. Disponível em: <https://www.outsystems.com/evaluation-guide/ai/app-development/>. Acesso em: 20 dez. 2023o.

OUTSYSTEMS. **OutSystems Platform Architecture**. Disponível em: <https://www.outsystems.com/evaluation-guide/architecture/>. Acesso em: 20 dez. 2023p.

OUTSYSTEMS. **Service Studio Overview - OutSystems 11 Documentation**. Disponível em: <https://success.outsystems.com/documentation/11/getting_started/service_studio_overview/>. Acesso em: 20 dez. 2023q.

OUTSYSTEMS. **Monitoring OutSystems applications - OutSystems Best Practices**. Disponível em: <https://success.outsystems.com/documentation/best_practices/performance_and_monitoring/monitoring_outsystems_applications/>. Acesso em: 11 fev. 2024a.

OUTSYSTEMS. **Managing the applications lifecycle - OutSystems 11 Documentation**. Disponível em:

<https://success.outsystems.com/documentation/11/managing_the_applications_lifecycle/>. Acesso em: 11 fev. 2024b.

OUTSYSTEMS. **Tag a Version - OutSystems 11 Documentation**. Disponível em: <https://success.outsystems.com/documentation/11/managing_the_applications_lifecycle/deploy_applications/tag_a_version/>. Acesso em: 11 fev. 2024c.

PEREIRA, R. **Rapid Application Development with OutSystems**. Birmingham, UK: Packt Publishing Ltd., 2022.

PINHO, D.; AGUIAR, A.; AMARAL, V. What about the usability in low-code platforms? A systematic literature review. **Journal of Computer Languages**, v. 74, 1 jan. 2023.

RICHARDSON, C.; RYMER, J. R. **New Development Platforms Emerge For Customer-Facing Applications**. [s.l: s.n.]. Disponível em: <www.forrester.com>.

ROMÃO, D. M. G. **Migration from Legacy to Reactive Applications in OutSystems**. [s.l: s.n.].

SAHAY, A. *et al*. **Supporting the understanding and comparison of low-code development platforms**. Proceedings - 46th Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2020. **Anais**...Institute of Electrical and Electronics Engineers Inc., 1 ago. 2020.

SOUZA, F. DE O. **Regime Próprio de Previdência Social (RPPS): vantagens e desvantagens da gestão pelos fundos de previdência municipais**. Recife: Universidade Federal Rural de Pernambuco, 11 maio 2020.

SPETS, S. **Application Security Verification Standard Compliance Analysis of a Low Code Development Platform**. [s.l: s.n.].

TISI, M. *et al*. **Lowcomote: Training the Next Generation of Experts in Scalable Low-Code Engineering Platforms**. [s.l: s.n.]. Disponível em: <https://hal.science/hal-02363416>.

WASZKOWSKI, R. **Low-code platform for automating business processes in manufacturing**. IFAC-PapersOnLine. **Anais**...Elsevier B.V., 2019.

WOO, M. The Rise of No/Low Code Software Development—No Experience Needed? **Engineering**, v. 6, n. 9, p. 960–961, set. 2020.