Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Arquitetura Reativa Cognitiva Baseada em Microsserviços e Micro-Frontends para Melhorar a Experiência do Usuário em Aplicações Bancárias por Meio de Interfaces Adaptativas

Michael Rodrigues da Silva

Dissertação apresentada como requisito parcial para qualificação do
Mestrado Profissional em Computação Aplicada

Orientadora
Prof.a Dr.a Leticia Lopes Leite

Brasília
2023

Ficha catalográfica elaborada automaticamente,
com os dados fornecidos pelo(a) autor(a)

Silva, Michael
SS586a Arquitetura Reativa Cognitiva Baseada em Microsserviços e Micro-Frontends para Melhorar a Experiência do Usuário em Aplicações Bancárias por Meio de Interfaces Adaptativas / Michael Silva; orientador Leticia Leite. -- Brasília, 2023.
130 p.

Dissertação(Mestrado Profissional em Computação Aplicada) -- Universidade de Brasília, 2023.

1. interface adaptativa. 2. aprendizado por reforço. 3. experiência do usuário. 4. micro frontends. 5. arquitetura de software. I. Leite, Leticia, orient. II. Título.

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Arquitetura Reativa Cognitiva Baseada em Microsserviços e Micro-Frontends para Melhorar a Experiência do Usuário em Aplicações Bancárias por Meio de Interfaces Adaptativas

Michael Rodrigues da Silva

Dissertação apresentada como requisito parcial para qualificação do
Mestrado Profissional em Computação Aplicada

Prof.a Dr.a Leticia Lopes Leite (Orientadora)
CIC/UnB

Prof. Dr. Edison Ishikawa
CIC/UnB

Dr.a Márcia Cristina Moraes
Colorado State University

Prof. Dr. Gladston Luiz da Silva
Coordenador do Programa de Pós-graduação em Computação Aplicada

Brasília, 1 de Agosto de 2023

Dedicatória

Primeiramente, agradeço a Deus, por todas as conquistas e pela fortaleza que tem sido na minha vida.

À minha querida esposa Pâmela Silva, por todo amor, compreensão e apoio, ao longo dessa trajetória.

À minha mãe, avó, avô e pai (Meirivan) por todo amor, carinho e dedicação.

Ao meu bebê amado, que ainda está na barriga da mamãe, mas já é a maior bênção que Deus concedeu na nossa vida.

Aos meus queridos familiares, amigos e irmãos, que se esforçaram para realizar o teste empírico e validar a hipótese apresentada.

Desejo que este trabalho seja uma gota de motivação na vida da minha família para que se dediquem à educação.

Por fim, dedico também à ciência. É um privilégio viver em um mundo com tantas mentes brilhantes.

Agradecimentos

Agradeço sempre a Deus que me ajudou até aqui (Marcos 6,41-42). Esses últimos anos foram um período de muitas dificuldades no âmbito pessoal e profissional. Vi, mais uma vez, a fidelidade d'Ele no cumprimento de Suas promessas.

Agradeço à Prof.a Dr.a Letícia Lopes Leite, minha orientadora, pelo suporte essencial durante este trabalho. Aos professores e colegas do Mestrado Profissional em Computação, por todo o conhecimento compartilhado. Aos professores que compõem a minha banca de avaliação, Prof. Dr. Edison Ishikawa e Prof.a Dr.a Marcia Moraes.

Aos colegas da Amazon, pelo constante auxílio e compreensão durante o mestrado.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES), por meio do Acesso ao Portal de Periódicos.

Resumo

O avanço das tecnologias de comunicação potencializou o uso da internet, afetando a vida da sociedade e impactando os negócios. O segmento bancário tem acompanhado essa evolução; e oferecido serviços essenciais pela internet para milhões de clientes. Nesse cenário, observa-se que a Experiência do Usuário (UX) é um fator relevante na percepção dos clientes das instituições, em relação à qualidade dos serviços, e influência na escolha por um banco ou outro. Como a arquitetura do software é um elemento que afeta a usabilidade, é essencial garantir que critérios de usabilidade e experiência do usuário sejam suportados no planejamento dessa arquitetura. Assim, este trabalho propõe uma arquitetura para sistemas bancários via web, que permite melhorar a experiência dos usuários, ao utilizar conceitos de interfaces adaptativas e sistemas reativos que se integram com serviços e bases de dados heterogêneas. Utilizou-se o modelo de *micro-frontends* para renderizar o *frontend*, pois este gera bases de código menores, mais coesas e sustentáveis, o que possibilita trabalhar com equipes autônomas, dispersas e tecnologias diferentes. A arquitetura proposta utiliza o algoritmo de Aprendizado por Reforço (AR), somado ao algoritmo *Monte Carlo Tree Search* e à técnica de Aprendizado Profundo (AP) para a criação de aplicações adaptativas. Por fim, para viabilizar tanto o algoritmo quanto o *frontend*, um sistema *backend* robusto foi criado, de modo a extrair, armazenar e processar os dados dos usuários, de modo a enviar *feedbacks* para o *frontend*, em tempo real. Diante disso, um Mínimo Produto Viável (MVP) foi desenvolvido com o intuito de demonstrar e validar os conceitos estudados. Em relação ao *backend*, foi adotada a arquitetura de microsserviços, por possibilitar que as aplicações sejam formadas por serviços pequenos, coesos e independentes. Foi realizado um experimento empírico, por intermédio de uma aplicação implementada com base na abordagem sugerida nesta pesquisa. Os resultados revelaram que a arquitetura alcançou os objetivos estabelecidos e o método de adaptação demonstrou superar tanto uma política não adaptativa quanto uma política baseada em frequência para uma página web que oferece serviços bancários.

Palavras-chave: interface adaptativa; aprendizado por reforço; experiência do usuário; *micro-frontends*; arquitetura de software.

Abstract

The advancement of communication and computing technologies has boosted the use of the internet, which affects the life of society and impacts business. The banking segment has followed this evolution; and offered essential services over the Internet to millions of customers. In this scenario, it is observed that the user experience is a relevant factor in the perception of the institutions' customers, in relation to the quality of the services, and influences the choice for one bank or another. As software architecture is an element that affects usability, it is essential to ensure that usability and user experience criteria are supported when planning this architecture. Thus, this work proposes an architecture for web banking systems, which allows improving the user experience, by using concepts of adaptive interfaces and reactive systems that integrate with services and heterogeneous databases. The micro-frontends model was used to render the frontend, as it generates smaller, more cohesive and sustainable code bases, which makes it possible to work with autonomous, dispersed teams and different technologies. The proposed architecture uses the Reinforcement Learning algorithm, in addition to the Monte Carlo Tree Search algorithm and the Deep Learning technique to create adaptive applications. Finally, to make both the algorithm and the frontend viable, a robust backend system was created in order to extract, store and process user data in order to send real-time feedback to the frontend. Therefore, an Minimum Viable Product was developed in order to demonstrate and validate the studied concepts. Regarding the backend, the microservices architecture was adopted, as it allows applications to be formed by small, cohesive and independent services. An empirical experiment was carried out, through an application implemented based on the approach suggested in this research. The results revealed that the architecture achieved the established objectives and the adaptation method was shown to overcome both a non-adaptive policy and a frequency-based policy for a web page that offers banking services.

Keywords: adaptive user interface, reinforcement learning, user experience, micro-frontends, software architecture.

Sumário

Lista de Figuras

Lista de Quadros

Lista de Abreviaturas e Siglas

AP Aprendizado Profundo.

APIs *Application Programming Interfaces*.

AR Aprendizado por Reforço.

ARBM Aprendizado por Reforço Baseado em Modelo.

AWS *Amazon Web Services*.

CDK *Cloud Development Kit*.

CE Critérios de Exclusão.

CI Critérios de Inclusão.

CSB Critérios de Seleção das Bases.

CSL Critérios de Seleção de Base.

CSS Folhas de estilo em cascata.

DRNN Modelo de Recomendação Sequencial Profundo.

DSE Decisão Sequencial Estocástico.

EDA Arquitetura Orientada a Eventos.

ES Engenharia de Software.

FOSS *Free and Open Source Software*.

HTML *Hypertext Markup Language*.

IA Inteligência Artificial.

IDE Ambientes de Desenvolvimento Integrados.

IHC Interação Humano-Computador.

IU Interface do Usuário.

IUA Interface do Usuário Adaptativa.

LSTM Camada de Memória de Longo Prazo.

MCTS Busca por Árvore de Monte Carlo.

ML Aprendizado de Máquina.

MVP Mínimo Produto Viável.

NoSQL Bancos de Dados Não Relacionais.

PDM Processo de Decisão de Markov.

QP Questão de Pesquisa.

QS Questões Secundárias.

RBUIS *Role-Based UI Simplification.*

RN Rede Neural.

RNP Redes Neurais Profundas.

RSL Revisão Sistemática de Literatura.

SOA Arquitetura Orientada a Serviços.

StArt *State of the Art through Systematic Review.*

UCT Limites Superiores de Confiança Aplicados a Árvores.

UnB Universidade de Brasília.

UX Experiência do Usuário.

Capítulo 1

Introdução

Nos últimos anos, os avanços nas tecnologias de comunicação e computação potencializaram o uso da internet, afetaram a vida da sociedade e impactam nos negócios. Conforme o relatório da ONU [1], a população mundial, em 2021, era de 7,8 bilhões de pessoas e 63% delas, ou seja, 4,9 bilhões, estavam conectadas à internet. Atualmente, a maioria das empresas e organizações tem seus próprios sites ou aplicativos para atender às necessidades dos clientes e, com isso, qualquer falha nos serviços pode provocar problemas significativos na rotina dos usuários [2].

No Brasil, o setor bancário tem acompanhado esses avanços tecnológicos, ao oferecer serviços essenciais para milhões de clientes pela internet. Um exemplo disso foi indicado em pesquisa de 2020, realizada pela Deloitte [3], a qual mostrou que 66,3% das transações executadas em bancos brasileiros foram realizadas de forma *on-line*, representando um acréscimo de quase 15%, em relação a 2016, em que 9 em cada 10 contratações de crédito são realizadas em canais digitais. Outro dado importante é em relação à transação instantânea Pix, realizada por meio de *Application Programming Interfaces* (APIs), que utilizam aplicações web e mobile. Segundo o Banco Central do Brasil [4], somente no mês de maio de 2022, foram realizadas mais de 784 milhões de transações financeiras por intermédio do Pix. Esses dados mostram que as pessoas dependem, cada vez mais, dos canais digitais para realizar suas operações financeiras; e da disponibilidade desses serviços de forma ininterrupta. Dessa forma, falhas podem gerar grandes prejuízos financeiros, tanto para as instituições quanto para os usuários. Ademais, um relatório da BIS [5] apontou que US$ 2,49 bilhões foram investidos em bancos digitais no Brasil, em 2019. Na América Latina, a participação do investimento em bancos digitais e o número de acordos foram, respectivamente, de 1% e 4% do total global. Entre 2017 e 2019, o investimento em bancos digitais cresceu mais de 100%, enquanto o número de acordos aumentou 28%, demonstrando a força de crescimento dessas instituições [6].

Apesar do cenário descrito possibilitar novas oportunidades de negócio, há também

desafios importantes para o setor bancário. No Brasil, a concorrência entre os bancos tradicionais e os novos bancos, chamados digitais, está cada vez mais acirrada. Segundo o relatório da Akamai Technologies, encomendado à Cantarino Brasileiro [7], 31% dos brasileiros utilizaram bancos digitais, em 2021, em um total de 82 milhões de contas [8]. Além disso, em 2020, a UBS Evidence Lab [9] divulgou um levantamento que indica que a porcentagem de downloads relacionados a aplicativos dos novos bancos atingiu a marca de 52%, superando as instituições tradicionais.

Considerando que os serviços bancários são semelhantes, a Experiência do Usuário (UX) se torna um fator fundamental para diferenciar a qualidade entre os bancos; assim, a escolha de muitos clientes ocorre devido a esse quesito [10]. Nessa perspectiva, torna-se relevante a distinção entre dois conceitos: a UX e a usabilidade. Segundo Norman e Nielsen [11], a UX é o conjunto de sensações e atitudes que o usuário vivencia, ao interagir com produtos ou serviços. Já usabilidade é um atributo de qualidade para avaliar a facilidade de uso de uma interface, portanto, é um elemento que influencia significativamente a UX [12][13]. Dessa maneira, os critérios de usabilidade podem ser utilizados para atingir uma experiência positiva [14].

Nielsen e Landauer [15] entendem que o conceito de usabilidade se refere à facilidade de compreensão e de uso de algo, cujos atributos básicos são: a facilidade de aprendizado, a eficiência, a capacidade de memorização, o número de erros durante a navegação e a satisfação ao usar. Contudo, o baixo nível de usabilidade é o maior causador de falhas em softwares [16]. Nesse sentido, Fishbein e Ajzen [17] afirmam que a experiência positiva de um usuário em determinada situação no passado impacta a próxima decisão. Dabholkar e Sheng [18] consideram que uma UX adequada em plataformas *on-line* cria impressões positivas, aumenta a satisfação dos clientes e influência as intenções futuras, enquanto uma experiência ruim pode gerar efeitos opostos. Segundo Bhattacherjee [19], adquirir novos clientes pode custar cinco vezes mais do que reter os existentes. Por fim, Zhou *et al.* [20] afirmam haver uma relação positiva entre experiência e satisfação, pois uma boa experiência gera maior satisfação com compras *on-line*, e, consequentemente, aumenta o número de transações realizadas.

Assim, por mais de duas décadas, a comunidade de Engenharia de Software (ES) tem buscado diferentes linhas de pesquisa, de modo a incorporar práticas de usabilidade propostas pela comunidade da área de Interação Humano-Computador (IHC) no desenvolvimento, arquitetura e *design* de software [21]. Bass e John [22] afirmam que a arquitetura de software é o artefato mais antigo que afeta a usabilidade, além de ser o artefato mais difícil de mudar, posteriormente, no processo de desenvolvimento. Consequentemente, é essencial garantir que recursos importantes de usabilidade sejam suportados na arquitetura de software. As conexões entre usabilidade e arquitetura de software são benéficas

tanto para a qualidade do produto final quanto para o processo de desenvolvimento.

Diante do exposto, o objetivo deste trabalho é propor uma arquitetura para sistemas bancários via web, que permita melhorar a experiência dos usuários, com base em conceitos de interfaces adaptativas e sistemas reativos que se integram com serviços e bases de dados heterogêneas. Sistemas denominados reativos são flexíveis, desacoplados, escaláveis e, consequentemente, facilitam o desenvolvimento e as mudanças. Eles também são mais tolerantes a falhas e altamente responsivos, que proporcionam um efetivo feedback interativo [23]. O sistema desenvolvido com base nessa arquitetura deve ser desacoplado em duas partes: *backend* - aplicação executada no servidor; e *frontend* - interface com a qual o usuário interage, que possibilita que ambas as partes sejam desenvolvidas e mantidas de forma independente. Além disso, o *backend* é fundamentado na arquitetura de microsserviços; e o *frontend* proporciona uma Interface do Usuário Adaptativa (IUA), por meio da arquitetura de *micro-frontends*.

A arquitetura de microsserviços possibilita que uma aplicação seja formada por um conjunto de serviços pequenos, coesos, independentes e construídos em torno da capacidade de negócios [24]. Além disso, em geral, os serviços devem ter seus próprios repositórios de dados [25].

O estilo arquitetural *micro-frontends* aplica os conceitos de microsserviços no lado do navegador, e transforma aplicativos da web em um único aplicativo que combina vários pequenos componentes. Nesse caso, o componente pode ser executado, desenvolvido e implantado de forma independente e por equipes diferentes [26]. A utilização desse estilo arquitetural facilita o desenvolvimento de uma IUA, que possibilita a criação de diversas versões da mesma interface para satisfazer as diferentes necessidades dos usuários [27].

Os bancos tradicionais possuem fontes de dados de tipos, acessos e formatos distintos; portanto, a arquitetura descrita neste trabalho também apresenta uma proposta para o desafio de integração dos dados, de modo a permitir o acesso descentralizado sobre as fontes, sem preocupações com investimentos em soluções compatíveis ou com ferramentas de transformação.

O resultado da arquitetura proposta pretende beneficiar tanto o usuário final quanto os bancos, que poderão fornecer serviços customizados e escaláveis, mesmo em regime de alta demanda de desenvolvimento e utilização. Empresas ou instituições não bancárias que prestam serviços essenciais por meio da internet e possuem sistemas com características semelhantes às dos sistemas bancários, ou seja, suportam alta demanda de negócio e desenvolvimento com integração a fontes de dados diversas, também poderão considerar a arquitetura proposta.

1.1 Definição do Problema

A aplicação web do banco tradicional Beta[1] é utilizada por mais de 54 milhões de clientes, os quais se conectam para realizar consultas e transações bancárias. Ela é composta por sistemas que seguem a arquitetura monolítica. Tais sistemas contêm todas as funcionalidades em um único processo e na mesma base de código-fonte e compartilham recursos da mesma máquina, tais como: processamento, memória, bancos de dados e/ou sistema de arquivos. Essa categoria de arquitetura traz limitações, pois, à medida que os sistemas crescem, tendem a apresentar problemas diversos, por exemplo: dificuldades de entendimento do código, alto acoplamento entre módulos, aumento do tempo de implantação e dependência de tecnologia a longo prazo [28][29][30]. Além disso, a instituição conta com diretorias de negócios responsáveis por criar e manter produtos, que geram demandas para a área de tecnologia. Tal estrutura exige que os sistemas tenham uma arquitetura de software que suporte alta demanda, seja resiliente a falhas, tenha componentes desacoplados, possua um nível elevado de segurança, disponha de facilidade de manutenção e, ainda assim, possibilite alta produtividade de desenvolvimento, além de oferecer experiência de utilização positiva para os usuários.

Uma arquitetura bem elaborada é aquela que descreve, claramente, as interfaces dos componentes, facilita o desenvolvimento do sistema, promove o cumprimento dos requisitos funcionais e suporta o reuso em outros projetos [31]. O resultado de uma boa arquitetura é possibilitar um software de melhor qualidade e permitir o adequado planejamento de ações acerca do processo de desenvolvimento. Medvidovic e Taylor [32] afirmam que uma boa arquitetura oferece o potencial para realizar: integridade conceitual, controle intelectual, reutilização efetiva de conhecimento, experiência, projetos e código, comunicação eficaz do projeto e gerenciamento de um conjunto de sistemas variantes relacionados. Por outro lado, uma arquitetura inadequada impacta negativamente a qualidade do sistema, pois resulta em softwares com constante dívida técnica, visto que será alterado sem nenhum tipo de controle [33][34], prejudicando a usabilidade e a UX. Além disso, podem gerar efeitos colaterais prejudiciais em outras propriedades de qualidade, como desempenho e confiabilidade Garcia *et al.* [35].

Com base nesse cenário, este trabalho tem o intuito de propor uma arquitetura consistente para sistemas bancários via web, de modo a melhorar a experiência dos usuários, fundamentado em conceitos de sistemas reativos, que se integram com serviços e bases de dados heterogêneas. Nesse sentido, cada diretoria de negócio pode desenvolver sistemas, corrigir problemas, criar otimizações e realizar a implantação de suas aplicações, de forma independente e paralela. O desenvolvimento paralelo proporciona agilidade e qualidade

[1]Banco Beta é um nome fictício para descaracterizar a instituição ao qual a arquitetura proposta foi validada.

ao produto final que o usuário utiliza, além de acelerar o processo de inovação. Assim, a integração com as diversas fontes de dados, as quais apresentam tipologia, acessos, formatos e regras de segurança distintas é um desafio ao qual este trabalho propõe uma solução.

1.2 Justificativa

Em função da problemática apresentada, o banco Beta necessitava de uma arquitetura de sistema que possibilitasse o desenvolvimento paralelo de seus produtos, de forma célere e com qualidade, de modo a aumentar a sua competitividade, em um mercado em que em que a concorrência apresenta estruturas enxutas e produtos cada vez mais tecnológicos e inovadores.

Pode-se justificar a viabilidade técnica desta pesquisa com base em casos de sucesso vivenciados por outros pesquisadores [36][37][37][38][39][40]. Em especial, Fernandez Garcia *et al.* [41] relatam as lições aprendidas no processo de desenvolvimento de sistemas que utilizaram interfaces de *mashup*[2] por *micro-frontends*, que requisitam dados para o *backend* formado por microsserviços. Outro estudo importante foi realizado pelos pesquisadores Todi *et al.* [42], que utilizaram o método de Aprendizado por Reforço com o algoritmo Busca por Árvore de Monte Carlo (MCTS), e redes neurais para gerar aplicações adaptativas. A melhoria na experiência do usuário foi percebida, a partir de uma pesquisa empírica realizada com usuários, aos quais avaliaram que a reorganização (adaptações) dos itens do menu facilitou a navegação, visto que simplificou a tarefa de encontrar os itens com maior interesse.

1.3 Objetivos

1.3.1 Objetivo Geral

O objetivo deste trabalho é **propor uma arquitetura para sistemas bancários via web, que permita melhorar a experiência dos usuários, com base em conceitos de interfaces adaptativas e sistemas reativos, que se integram com serviços e bases de dados heterogêneas**.

[2] *Mashup* é um tipo específico de software que agrupa serviços de diferentes fontes em uma mesma aplicação [41].

1.3.2 Objetivos Específicos

Para atingir o objetivo deste trabalho, os seguintes objetivos específicos foram identificados:

1. realizar a Revisão Sistemática de Literatura Revisão Sistemática de Literatura (RSL) para identificar algoritmos que busquem proporcionar uma interface adaptativa ao usuário;
2. selecionar um, dentre os algoritmos resultantes da RSL, para ser utilizado na arquitetura proposta;
3. verificar se o algoritmo selecionado proporciona uma aplicação web adaptativa capaz de melhorar a experiência dos usuários, por meio de experimentos empíricos;
4. definir como o algoritmo selecionado será incorporado na arquitetura proposta;
5. identificar quais dados devem ser analisados pelo algoritmo selecionado e como esses dados serão extraídos.
6. definir a camada de integração de dados que conectará as aplicações e as fontes de dados heterogêneas;
7. definir e descrever o desenho final da arquitetura proposta;
8. definir o escopo de um MVP, para que seja possível realizar uma validação empírica da arquitetura proposta por usuários de bancos on-line;
9. implementar o MVP, utilizando a arquitetura proposta;
10. definir e executar um roteiro de testes a serem aplicados no MVP;
11. utilizar os resultados obtidos nos testes executados no MVP para analisar a utilização da arquitetura proposta.

1.4 Resultados Esperados

A contribuição que se vislumbrou com este trabalho foi a proposição de uma arquitetura para sistemas bancários via web, que permite melhorar a experiência dos usuários, com base em conceitos de interfaces adaptativas e sistemas reativos, que se integram com serviços e bases de dados heterogêneas.

Outra contribuição importante é o resultado da validação empírica dos métodos utilizados. Essa arquitetura permite o desenvolvimento de softwares responsivos, resilientes,

elásticos e orientados a mensagens, que proporcionam feedbacks para os usuários, em tempo real, e que se integrem com serviços e fontes de dados heterogêneas.

Como consequência, a arquitetura proposta visa facilitar o desenvolvimento paralelo de sistemas que poderão ser construídos com base em paradigmas, linguagens e *frameworks*, que melhor se adéquem às regras de negócio. Essas características têm o objetivo de conceder liberdade, velocidade e qualidade para a criação de produtos, novas funcionalidades e correções de problemas, que permitam aos sistemas fornecer experiência positiva aos seus usuários.

A arquitetura de software apresentada, além de permitir a aplicação em sistemas bancários complexos, também poderá ser estendida para sistemas web relacionados, isto é, para empresas ou instituições que possuem equipes diversas e heterogêneas, que desenvolvem projetos de forma paralela e requerem velocidade e qualidade de produção, além de reação positiva, mesmo sob alta demanda. Ademais, o protótipo desenvolvido baseado na arquitetura proposta visa validar a sua aplicabilidade, bem como mostrar os benefícios e os desafios de sua utilização.

1.5 Estrutura do Trabalho

Este trabalho está organizado em sete capítulos. No Capítulo 2, é apresentada a fundamentação teórica acerca dos assuntos necessários para o entendimento deste trabalho.

O Capítulo 3 descreve a execução de uma revisão sistemática de literatura, a fim de identificar estruturas, padrões e protocolos da arquitetura de microsserviços, que serão utilizados no *backend*; e estruturas, padrões e protocolos que possam ser integradas em uma solução de *micro-frontends*.

O Capítulo 4, Arquitetura Proposta, apresenta uma arquitetura para sistemas bancários via web, que permite melhorar a experiência dos usuários, com base em conceitos de interfaces adaptativas e sistemas reativos, que se integram com serviços e bases de dados heterogêneas.

No Capítulo 5, é apresentado um protótipo desenvolvido conforme a arquitetura proposta para validar a viabilidade de sua utilização e, eventualmente, reajustá-la conforme os resultados dos testes executados.

O Capítulo 6 apresenta a validação empírica, por intermédio de um protótipo baseado em conceitos apresentados neste trabalho. Também apresenta os resultados quantitativos e qualitativos.

O Capítulo 7 apresenta as ponderações acerca da proposta defendida neste trabalho e os possíveis pontos negativos, além de propor trabalhos futuros.

Capítulo 2

Fundamentação Teórica

Este capítulo apresenta um detalhamento teórico acerca dos principais conceitos utilizados no presente trabalho.

2.1 Aprendizado por Reforço Baseado em Modelo

O Aprendizado por Reforço (AR), conforme apresentado na Figura 2.1, é uma classe de métodos de aprendizado de máquina, o qual permite que um agente aprenda em um ambiente interativo, por tentativa e erro, baseado no *feedback* de suas próprias ações e experiências [42].

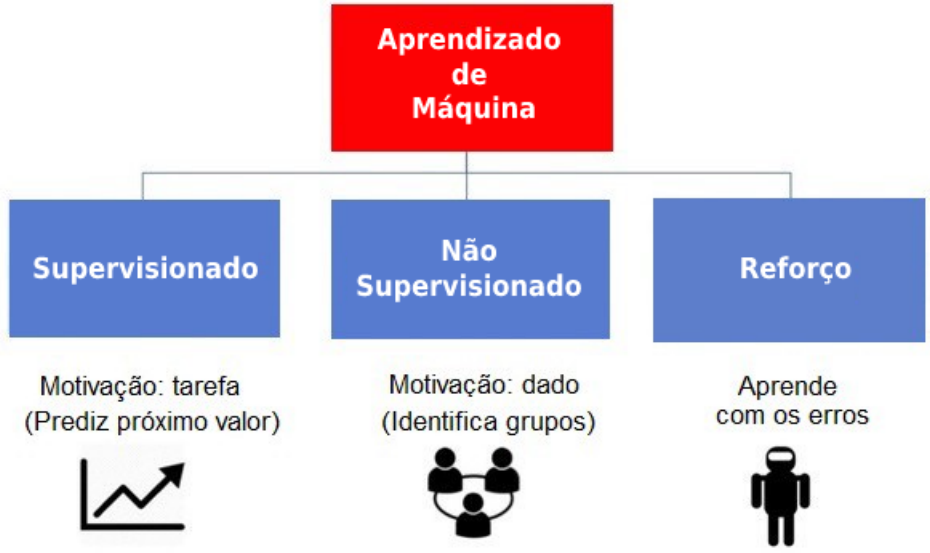

Figura 2.1: Tipos de aprendizado de máquina
Fonte: Adaptada de Bhatt [43]

Embora tanto o aprendizado supervisionado quanto o AR usem mapeamento entre entrada e saída, ao contrário do aprendizado supervisionado, em que o *feedback* fornecido ao agente é o conjunto correto de ações para executar uma tarefa; o AR usa recompensas e punições como sinais de comportamento positivo e negativo. Em comparação com o aprendizado não supervisionado, o AR difere em termos de objetivos. Enquanto o objetivo do aprendizado não supervisionado é encontrar semelhanças e diferenças entre os pontos de dados, no caso do AR, o objetivo é encontrar um modelo de ação adequado, que maximize a recompensa cumulativa total do agente. A Figura 2.2, a seguir, ilustra o ciclo de *feedback* ação-recompensa de um modelo genérico de AR.

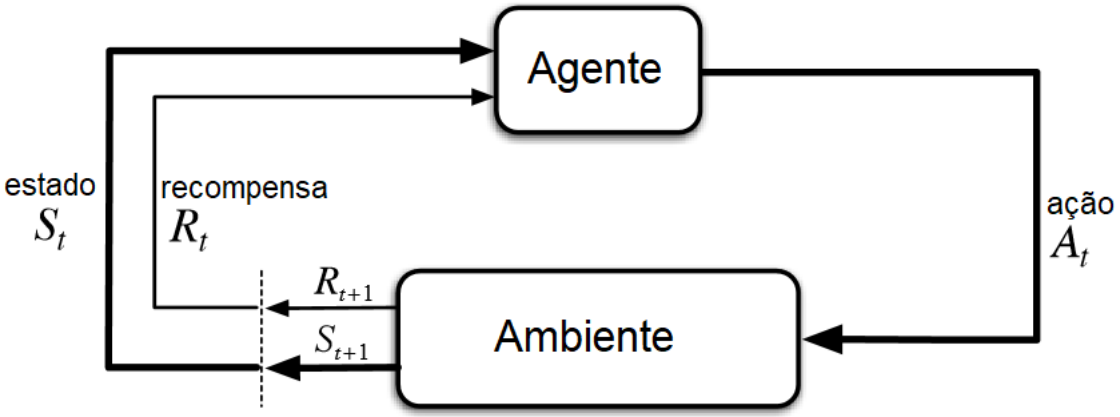

Figura 2.2: Tipos de aprendizado de máquina
Fonte: Adaptada de Bhatt Bhatt

Já o Aprendizado por Reforço Baseado em Modelo (ARBM) usa um modelo preditivo para simular possibilidades, sem teste prévio, o que é útil para interfaces adaptativas, pois melhora, significativamente, a capacidade de encontrar soluções eficientes [42]. Assim, uma decisão pode ser feita com muito menos tentativas e, se o modelo for eficiente, a decisão poderá ser tomada diretamente. Neste trabalho, o ARBM foi utilizado em conjunto com o algoritmo Busca por Árvore de Monte Carlo (MCTS), pois encontrar a melhor adaptação para o usuário e avaliar o valor de cada sequência de adaptações com modelos preditivos é, computacionalmente, custoso. O MCTS pode operar sob incerteza, analisar as adaptações mais promissoras e expandir nós de árvore com abordagens baseadas em amostras aleatórias.

2.2 Busca por Árvore de Monte Carlo

O MCTS é um algoritmo de tomada de decisão que consiste em buscar espaços combinatórios representados por árvores. Em tais árvores, os nós denotam estados, também chamados configurações do problema, enquanto as arestas denotam transições (ações) de um estado para outro [44]. A busca pelo melhor resultado é feita ao realizar as simulações com uma árvore que cresce para acomodar novos nós, temporalmente; os nós mais promissores são, em teoria, alcançados primeiro e visitados com mais frequência do que os nós que, provavelmente, não serão atraentes [45]. O MCTS é um método interativo, que executa quatro etapas, até que o tempo disponível se esgote, como mostra a Figura 2.3.

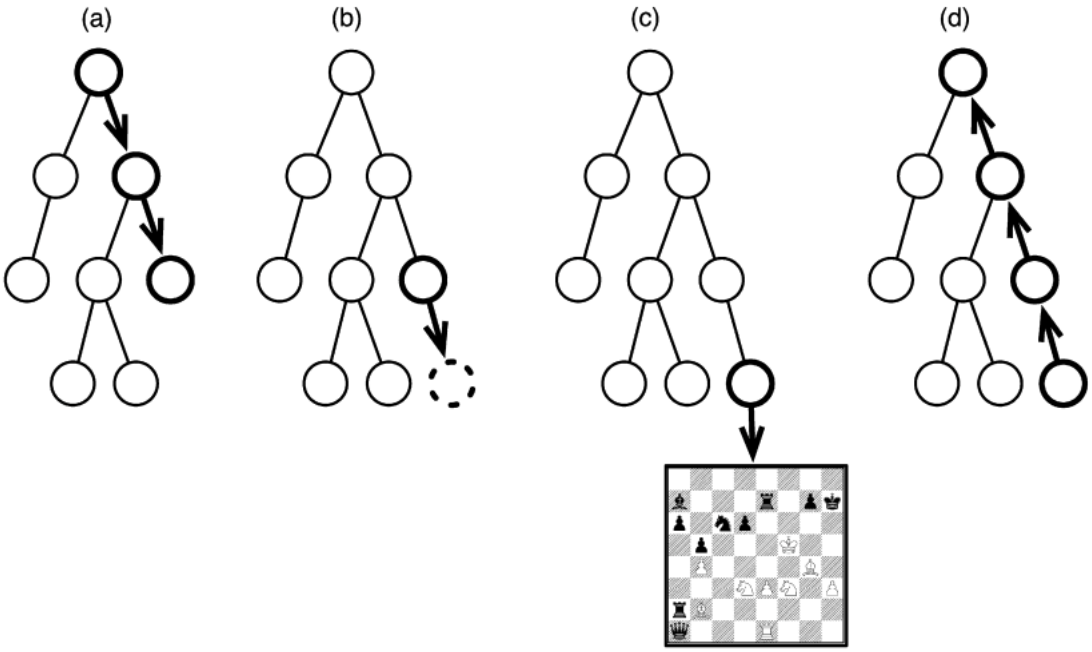

Figura 2.3: Fases do algoritmo MCTS
Fonte: Adaptada de Ciancarini e Favini [45]

As etapas são:

- Seleção (a): o algoritmo seleciona um nó na folha da árvore com base no número de visitas e seu valor médio;
- Expansão (b): o algoritmo adiciona novos nós à árvore, de maneira opcional;
- Simulação (c): o algoritmo simula o resto, uma ou mais vezes, e retorna o valor do estado final (ou sua média, se simulado várias vezes);

- Retropropagação (d): o valor é propagado para os ancestrais do nó até a raiz e novos valores médios são calculados para esses nós.

Após executar as quatro fases, até o tempo permitir, o programa escolhe o filho da raiz que recebeu mais visitas. Isso pode não coincidir com o nó com o maior valor médio. Há discussão acerca do motivo por que o operador médio sozinho não é uma boa escolha [46].

2.3 Arquitetura de Software

A arquitetura de software surgiu como uma importante disciplina da ES, a qual representa a divisão do sistema, em diversas partes, e as relações específicas entre estas. Essa divisão é o que permite que grupos de pessoas - muitas vezes separados por limites organizacionais, geográficos e até de fuso horário - trabalhem cooperativa e produtivamente para resolver um problema muito maior, que qualquer um deles poderia resolver individualmente. Cada grupo desenvolve uma parte do software, que interage com as partes que outros grupos de pessoas estão desenvolvendo; todas essas partes se comunicam umas com as outras, por meio de interfaces elaboradas, que revelam informações mínimas e mais estáveis necessárias para a interação.

Dessa interação emergem os atributos de funcionalidade e qualidade de software - segurança, modificabilidade, desempenho, entre outros - que as partes interessadas do sistema exigem. Quanto maior e mais complexo o sistema, mais crítico é esse particionamento e a arquitetura. Além disso, quanto mais exigentes forem esses atributos de qualidade, mais crítica também será a arquitetura, pois deverá responder a questões importantes para o funcionamento do software, conforme os exemplos a seguir [47].

- O sistema necessita de alto desempenho? Então, será necessário:
  - explorar o paralelismo potencial e decompor o trabalho em processos de cooperação ou sincronização;
  - gerenciar o volume de comunicação entre processos e as frequências de acesso a dados;
  - conseguir estimar latências e rendimentos esperados;
  - identificar potenciais gargalos de desempenho.
- A segurança é importante para o sistema? Então, será necessário:
  - descrever as relações de uso e as restrições de comunicação entre as partes;

  - identificar as partes do sistema em que uma intrusão não autorizada causará mais danos;
  - introduzir componentes de segurança que já tenham conquistado um alto grau de confiança.

- É necessário se preocupar com a modificabilidade e portabilidade? Então, as preocupações entre as partes do sistema devem ser separadas, para que quando uma mudança afetar um componente, ela não se espalhe pelo sistema.

Na prática, a arquitetura de software é modelada e documentada, por intermédio de visões de arquitetura, que são, basicamente, representações de um sistema para interesses particulares. O conjunto de visões não é fixo, pois, vários pontos de vista são introduzidos, conforme a necessidade. Nesse mesmo sentido, a norma ISO/IEC 42010 [48] orienta que, em um sentido abstrato, uma descrição de arquitetura consiste em um conjunto de visões, cada uma das quais concorda com um ponto de vista que compreende as várias preocupações dos diferentes *stakeholders* [49].

2.4 Sistemas Reativos

Um sistema é classificado como reativo se for responsivo, resiliente, elástico e orientado a mensagens [23], isto é:

- Responsivo: o sistema responde às requisições em um tempo razoável, se possível. O fator fundamental da usabilidade e da utilidade é a responsividade, isso significa que os problemas podem ser detectados, rapidamente, e tratados com a máxima eficácia. Sistemas responsivos pretendem prover tempos de resposta curtos e consistentes; estabelecer margens de tolerância confiáveis, que garantem uma qualidade de serviço consistente, fato que reforça a confiança do usuário final;
- Resiliente: o sistema responde ao usuário, mesmo em caso de falhas. A resiliência é obtida por replicação, contenção, isolamento e delegação. Falhas são contidas dentro de cada componente, isolados uns dos outros, portanto, com a garantia de que partes do sistema podem falhar e se recuperar, sem comprometer o sistema na totalidade. A recuperação de cada componente é delegada a outro componente (externo); assim, alta disponibilidade é garantida por replicação, quando necessária e os clientes de cada componente não são sobrecarregados com a responsabilidade do tratamento de falhas;

- Elástico: o sistema continua responsivo, mesmo sob variações de carga de demanda. Sistemas Reativos podem reagir a mudanças na taxa de solicitações, por meio do aumento ou diminuição dos recursos alocados para lidar com essas entradas;
- Orientados a Mensagens: sistemas reativos se baseiam em transferência de mensagens assíncronas, para estabelecer fronteiras entre os componentes e garantir baixo acoplamento, isolamento e transparência na localização. Comunicação não bloqueante permite que os destinatários consumam recursos apenas enquanto estão ativos, o que produz menor sobrecarga no sistema.

Essa categoria de sistema isola o estado, o comportamento, as falhas e o ciclo de vida de seus componentes. Os componentes são desacoplados, em termos de tempo, porque seus ciclos de vida são, potencialmente, independentes [50].

2.5 Microsserviços

Microsserviços é um estilo arquitetural que enfatiza a divisão do sistema em serviços pequenos e leves, que são, propositadamente, construídos para desempenhar uma função de negócios muito coesa. Os benefícios comumente acordados desse estilo incluem: aumento da agilidade, produtividade do desenvolvedor, resiliência, escalabilidade, confiabilidade, capacidade de manutenção, separação de responsabilidade (cada microsserviço resolve um problema de negócio) e facilidade de implantação, de modo a permitir que equipes e organizações de software sejam mais produtivas e criem produtos de software frequentemente mais bem-sucedidos [51].

No entanto, esses benefícios vêm com desafios, como descoberta de serviços na rede, gerenciamento de segurança, otimização de comunicação, compartilhamento de dados e desempenho [52]. Por meio dos microsserviços, é possível desenvolver sistemas reativos, pois cada serviço tem o seu estado, comportamento, falhas e ciclo de vida isolados e independentes, conforme o descrito na Seção 2.4.

2.6 Banco de Dados Chave e Valor

Um banco de dados chave-valor é uma categoria de sistema de gerenciamento de banco de dados NoSQL, que armazena e recupera informações, por intermédio de uma estrutura simples, composta por chaves e valores [53]. Nesse modelo, os dados são organizados em pares únicos de chave e valor, em que cada chave está associada a um valor correspondente. O valor pode assumir diversas formas, como *strings*, números, objetos JSON ou até documentos completos. A principal característica distintiva de um banco de dados

chave-valor reside em sua capacidade de acesso e recuperação muito eficientes dos dados, visto que as buscas são realizadas diretamente pela chave. Esse modelo é, especialmente, vantajoso quando há a necessidade de acessar ou atualizar informações, com base em uma chave específica, sem a necessidade de percorrer uma estrutura complexa, em busca dos dados desejados [54].

Ademais, demonstra alta escalabilidade e flexibilidade, consegue lidar com grandes volumes de dados e pode ser distribuído em múltiplos nós, com o resultado de um aumento da capacidade de armazenamento e do desempenho geral. Essa flexibilidade também possibilita a integração simplificada desses bancos de dados com outros sistemas e aplicativos. Entretanto, é importante ressaltar que os bancos de dados chave-valor possuem um modelo de dados muito simples e podem não ser adequados para todas as aplicações. Eles não oferecem recursos avançados de consulta, como junções ou consultas complexas, comumente encontrados em bancos de dados relacionais [55]. Portanto, seu uso é mais apropriado em situações em que a velocidade e a simplicidade na recuperação dos dados são fundamentais, como em *caches*, sistemas de sessão, armazenamento de metadados e outras aplicações em que a chave representa o principal meio de acesso aos dados.

2.7 Usabilidade e UX

A palavra usabilidade foi evidenciada na década de 1980, como um substituto para o termo "facilidade de uso". De acordo com Sarodnick e Brau [56], a primeira definição formal de usabilidade foi proposta por Shackel [57] que, descreveu a usabilidade de um produto como "[...] a capacidade de ser usado por humanos de forma fácil e eficaz". Desde então, muitas definições de usabilidade surgiram, resumidas em quatro perspectivas diferentes [58]:

1. a **perspectiva orientada ao produto** enfatiza que a usabilidade pode ser capturada como uma característica inerente ao produto;

2. de acordo com a **perspectiva orientada ao usuário**, a usabilidade é uma função do esforço mental do usuário e sua atitude em relação ao uso do produto;

3. a **perspectiva orientada para o desempenho** enfatiza que a usabilidade é descrita em termos da interação do usuário com o produto;

4. a **perspectiva orientada ao contexto** enfatiza que a usabilidade depende do grupo de usuários estudado, das tarefas que esses usuários realizam e do ambiente em que as tarefas são concluídas.

Dessa forma, a *International Standard Organization* (ISO) percebeu que todas essas perspectivas precisavam ser consideradas em uma definição formal. Com isso, em 2010, publicou a norma ISO 9241-210 [59], que definiu a usabilidade como a medida em que um sistema, produto ou serviço pode ser utilizado pelos usuários para atingir seus objetivos específicos, com efetividade, eficiência e satisfação, em um contexto específico de utilização.

Apesar das várias diferenças nas definições de usabilidade, elas incluem, normalmente, medidas de resultados relacionadas ao desempenho, como: taxa de erro, taxa de retenção e taxa de aprendizado. Além desses, são consideradas as medidas de resultados relacionadas ao tempo, como: satisfação, alegria, prazer. Por exemplo, Nielsen e Landauer [15] definiram que um artefato com bom nível de usabilidade é aquele fácil de aprender e de lembrar, eficiente, que produz poucos erros e seja subjetivamente agradável. Apesar desses elementos compartilhados, a usabilidade tem sido criticada por representar um "conceito guarda-chuva"[60], e muitas vezes, é confundida com UX.

De fato, usabilidade e UX estão relacionadas, porém, antes de realizar essa comparação, é importante citar o conceito relacionado ao segundo termo. A UX é, frequentemente, considerada um conceito difuso e mal definido [61]. Ao examinar as definições apresentadas na literatura de pesquisa, elas podem ser classificadas em três tipos principais [62]:

1. visão holística da experiência do usuário;
2. extensão do conceito de usabilidade;
3. foco principal na emoção.

A abordagem holística se refere à UX como todas as ações, sensações, considerações, sentimentos e criação de sentido de uma pessoa, ao interagir com um dispositivo ou um serviço técnico [62]. De acordo com essa abordagem, a UX abrange uma gama muito grande de resultados da interação usuário-dispositivo, que inclui cognições, atitudes, crenças, comportamento, intenções comportamentais e afetos. No segundo grupo de definições, que descrevem a UX como uma extensão da usabilidade, seu escopo (eficiência, eficácia, satisfação) é estendido, e adiciona-se afeto e emoções. A terceira abordagem é semelhante à segunda, mas concentra-se mais no resultado afetivo da interação usuário-dispositivo. Seguindo essa abordagem, Desmet e Hekkert [63] descrevem a UX como um conjunto de resultados afetivos, influenciados pela experiência de estética, emoção e significado.

Assim, a abordagem mais abrangente é a holística, enquanto a terceira abordagem representa a mais restrita. No caso da UX, o nível de diversidade de opiniões parece ser particularmente alto [62]. Com o objetivo de padronização conceitual, assim como fez para usabilidade, a ISO 9241-210 [59] definiu o conceito de UX como: "[...] as respostas

e percepções de uma pessoa resultantes do uso de um produto, sistema ou serviço". Essa definição alerta para uma importante questão: a UX representa uma extensão significativa do conceito de usabilidade ou a UX apenas ajudou a estender e melhorar o significado do conceito de usabilidade?

A compreensão da UX não é apenas mais ampla, mas também mais diversificada, em comparação com o conceito de usabilidade [64]. Quando a UX é definida conforme a abordagem holística, ela engloba todas as consequências subjetivas, emocionais e comportamentais da interação de um usuário com um dispositivo. Por outro lado, quando os aspectos de satisfação e comportamento do usuário também são incluídos nesta definição, a UX representa uma extensão do conceito de usabilidade, que incorpora todos os componentes desta. Se a segunda abordagem for adotada, a definição de experiência do usuário engloba todos os componentes subjetivos da interação usuário-dispositivo, mas não inclui o componente "desempenho/comportamento". A experiência do usuário e a usabilidade descrevem, portanto, dois conceitos separados, que compartilham o elemento satisfação. Quanto à terceira e mais restrita definição, a qual a UX é meramente o resultado afetivo da interação usuário-dispositivo, experiência do usuário e usabilidade representam dois conceitos independentes, ou seja, que não compartilham componentes entre si.

Assim como Lindgaard e Dudek [65], este trabalho reconhece que os conceitos de afeto e satisfação não são mutuamente exclusivos, portanto, não considera viável a terceira abordagem. Durante o desenvolvimento deste trabalho, as definições da ISO 9241-210 [59] serão consideradas, reforçadas por Norman e Nielsen [11], que afirmam que a UX é o conjunto de sensações e atitudes que o usuário vivencia, ao interagir com produtos ou serviços. Ademais, Park *et al.* [12] dizem que a usabilidade é um atributo de qualidade para avaliar a facilidade de uso de uma interface; portanto, é um elemento que influencia, significativamente, a UX. Dessa maneira, entende-se que os critérios de usabilidade podem ser utilizados para atingir uma experiência positiva [14].

A arquitetura proposta por este trabalho considera os aspectos de usabilidade e de UX, pois esses dois elementos são essenciais para que um software atinja os seus objetivos, em consonância com o que Bass e John [22] afirmaram: "[...] arquitetura de software é o artefato mais antigo que afeta a usabilidade, também o artefato mais difícil de mudar posteriormente no processo de desenvolvimento".

2.8 *Micro-Frontends*

As aplicações *frontends* tradicionais são construídas com base na arquitetura monolítica, cuja execução é centralizada em um único servidor. Nesse tipo de arquitetura, a tendência é conter um grande número de módulos; as dependências não são claras; os limites entre

os módulos são confusos; e o código tem alto nível de acoplamento. Além disso, para cada modificação, em alguma parte do sistema, mesmo que mínima, deve-se realizar a implantação de todo o código do sistema novamente, o que causa desperdício de recursos computacionais [26]. A depender do tamanho do sistema e da quantidade de linhas de código, a implantação leva muito tempo para ser efetuada, gerando um impacto negativo para o usuário.

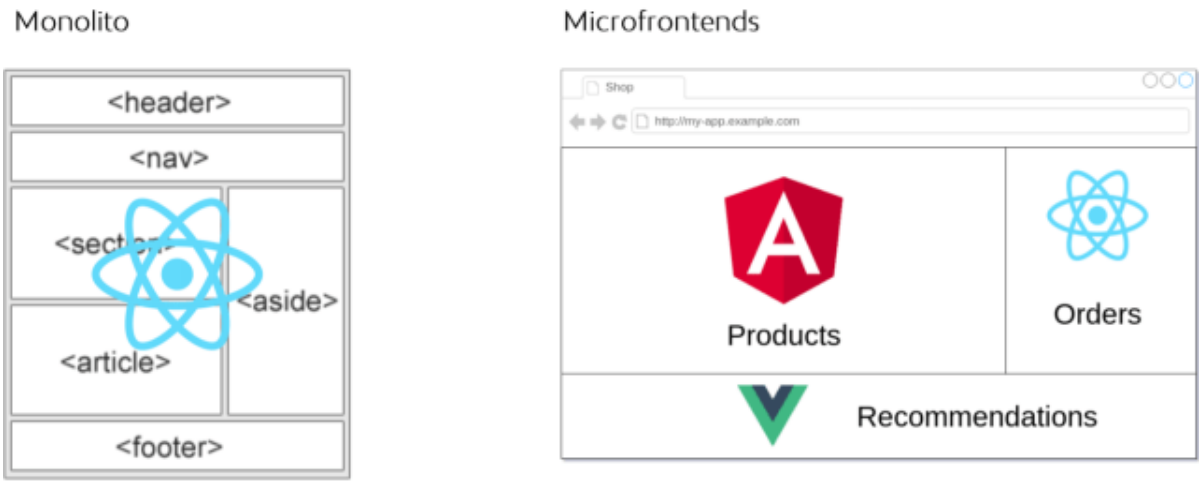

Figura 2.4: *Frameworks* diferentes na mesma página
Fonte: Adaptado de Insights [66]

A ideia dos *micro-frontends* é tratar uma aplicação da web como uma combinação de componentes pertencentes a diferentes equipes, em que cada componente (*micro-frontend*) tem o fluxo de implantação independente e pode ser desenvolvido com o uso de diferentes *frameworks*, que convivem harmonicamente, na mesma página web, como mostra a Figura 2.4. Isso contribui para que as equipes sejam multifuncionais e desenvolvam as regras de negócios necessárias de ponta a ponta, isso é, do *backend* ao *frontend* [67].

2.9 Interface Adaptativa

As IU são consideradas, por alguns autores, como um aspecto primordial da satisfação do usuário de computador, e, segundo El-Bakry *et al.* [68], a Interface do Usuário (IU) é o recurso pelo qual os usuários interagem com o sistema (máquina, dispositivo, software ou outra ferramenta complexa) que fornece meios de:

- Entrada: permite que os usuários manipulem um sistema;
- Saída: permite que o sistema indique os efeitos da manipulação dos usuários.

Em algumas circunstâncias, os computadores podem observar o usuário e reagir de acordo com suas ações, sem este realizar algum comando específico. De toda forma, uma interface correta deve tornar a interação do usuário o mais simples e eficiente possível, a fim de atingir os objetivos do usuário. Ao criar uma IU, o *designer* deve equilibrar a funcionalidade técnica e os elementos visuais para criar um sistema que não seja apenas operacional, mas também utilizável e adaptável às mudanças, conforme a necessidade do usuário [68].

Uma interface que se adapta conforme a necessidade do usuário é chamada de Interface do Usuário Adaptativa (IUA). Segundo Langley [69], a IUA é um artefato de software que melhora a capacidade de interagir com um usuário, por meio da construção de um modelo baseado na experiência de utilização desse mesmo usuário. Uma IUA não existe de forma isolada, mas é um meio pelo qual o usuário interage com o computador; e o computador, no que lhe concerne, utiliza experiências passadas para também interagir com o usuário e adaptar a interface conforme o hábito de utilização.

A definição de interface adaptativa tem semelhança com as definições comuns de aprendizado de máquina, por exemplo, a definição de Langley e Morgan [70]:

O aprendizado de máquina é o estudo de métodos computacionais para melhorar o desempenho, mecanizando a aquisição de conhecimento, a partir da experiência. O aprendizado de máquina visa fornecer níveis crescentes de automação no processo de engenharia do conhecimento, substituindo muitas atividades humanas demoradas por técnicas automáticas que melhoram a precisão ou a eficiência, descobrindo e explorando regularidades nos dados de treinamento. O teste final do aprendizado de máquina é sua capacidade de produzir sistemas que são usados regularmente na indústria, educação e outros lugares

Nesse sentido, as IUA constituem uma classe especial de sistemas de aprendizado, projetados para ajudar os humanos, em contraste com grande parte do objetivo inicial do aprendizado de máquina, que visava desenvolver sistemas baseados em conhecimento, de modo a substituir especialistas em determinada atividade [69].

2.10 Aprendizado Profundo

O Aprendizado Profundo (AP) é um subconjunto da matéria de aprendizado de máquina. É uma Rede Neural (RN) composta por camadas e parâmetros. A maioria dos métodos de AP utiliza arquiteturas de RN. Por isso, também é referido como RNP. Dessa forma, o AP utiliza as camadas de unidades de processamento não linear para extração e transformação de recursos. As camadas inferiores próximas à entrada de dados aprendem recursos simples, enquanto as camadas superiores aprendem recursos mais complexos, derivados

de recursos da camada inferior [71], que formam uma representação hierárquica e poderosa de recursos. O AP é adequado para analisar e extrair conhecimento útil de grandes quantidades de dados e dados coletados de diferentes fontes [72].

A camada é a principal estrutura de dados nas RN; funciona como um processador de dados que recebe um ou mais vetores e retorna um ou mais vetores com os dados já transformados. As camadas possuem pesos, que especificam as transformações a serem feitas nos dados para criar o modelo preditivo. A aprendizagem do modelo consiste em encontrar um conjunto de valores para os pesos de todas as camadas de uma rede, de modo a maximizar o poder preditivo desse modelo.

Na arquitetura proposta por este trabalho, as adaptações na aplicação devem ocorrer em tempo real, enquanto o usuário estiver utilizando a aplicação. Dessa maneira, são incorporadas redes neurais de valor pré-treinadas para ajudar a prever quais ramificações de uma MCTS têm o maior valor esperado e, assim, lidar com instâncias de problemas maiores, e calcular, mais rapidamente, sequências mais longas.

Capítulo 3

Revisão Sistemática de Literatura

Este capítulo descreve a execução de uma Revisão Sistemática de Literatura (RSL) para exploração de evidências que alimentem a fase empírica da pesquisa, conforme descrito no objetivo específico 1. O objetivo dessa execução é identificar e analisar algoritmos frequentemente utilizados para o desenvolvimento de aplicações web adaptativas.

A RSL é uma forma de estudo secundário, que utiliza uma metodologia bem definida para identificar, analisar e interpretar todas as evidências disponíveis, a respeito de uma questão de pesquisa particular, de maneira imparcial e repetível [73]. Cook *et al.* [74] acrescentam que a RSL é um tipo de investigação científica, com métodos pré-planejados, que sintetizam os resultados de estudos primários, por intermédio de estratégias que limitam o viés e o erro aleatório. Essas estratégias incluem os passos de buscar, selecionar e avaliar estudos potencialmente relevantes, com base em critérios explícitos e reproduzíveis [74].

As primeiras técnicas formais que combinaram e sintetizaram resultados de diferentes estudos foram elaboradas e publicadas no *British Medical Journal*, pelo matemático britânico Karl Pearson, em 1904. Ele estudou o efeito preventivo das inoculações contra a febre entérica [75]. Em 1955, aparece a primeira revisão sistemática acerca de um cenário clínico, publicada no *Journal of American Medical Association* [76]. A era das revisões sistemáticas na área da saúde se consolidou no fim da década de 1980 com a publicação do livro *Effective Care During Pregnancy and Childbirth* [77]. Na década de 1990, nasce a fundação da *Cochrane Collaboration*, com o objetivo de preparar, manter e disseminar revisões sistemáticas na área da saúde. Em 1995, um grupo de cientistas reunidos em Potsdam (Alemanha) definiu como revisão sistemática "[...] a aplicação das estratégias científicas que limitam o viés de seleção e avaliam com espírito científico os artigos e sintetizam todos os estudos relevantes em tópicos específicos"[78].

A RSL surgiu no momento em que a medicina mudou a forma de realizar pesquisas, pois eram produzidas de maneira aleatória, não sistemáticas e carregadas de vieses.

Diante disso, passou-se a adotar um paradigma baseado em evidências que considera o julgamento clínico de especialistas e organiza de forma mais efetiva a pesquisa médica. Posteriormente, a comunidade de Engenharia de Software (ES) percebeu que a metodologia também poderia ser aplicada na área de computação, visto que os levantamentos da literatura eram realizados de forma aleatória e propensos a erros [79].

O artigo influente de Kitchenham [80] introduz e discute o conceito de Engenharia de Software baseada em Evidência (*Evidence-based Software Engineering*), ao qual se baseia nos conceitos da Medicina baseada em Evidência (*Evidencebased Medicine*) para propor uma mudança de paradigma em como as pesquisas na área de ES deveriam ser conduzidas. Essa nova forma deveria empregar um processo metódico para identificar, avaliar e interpretar todas as evidências científicas disponíveis e relevantes relacionadas a um tema específico de interesse [73].

Em geral, os estudos na área de ES têm menos pesquisas empíricas e os métodos de pesquisa utilizados pelos engenheiros de software são menos rigorosos do que os utilizados por pesquisadores da área médica [80]. Engenheiros, normalmente, não têm evidências suficientes para determinar adequação, limitações, qualidades, custos e riscos no processo de desenvolvimento de software. A RSL pode ajudar nesse processo, pois, por meio das sínteses de pesquisa, é possível escolher tecnologias e metodologias apropriadas baseadas em fatos [81].

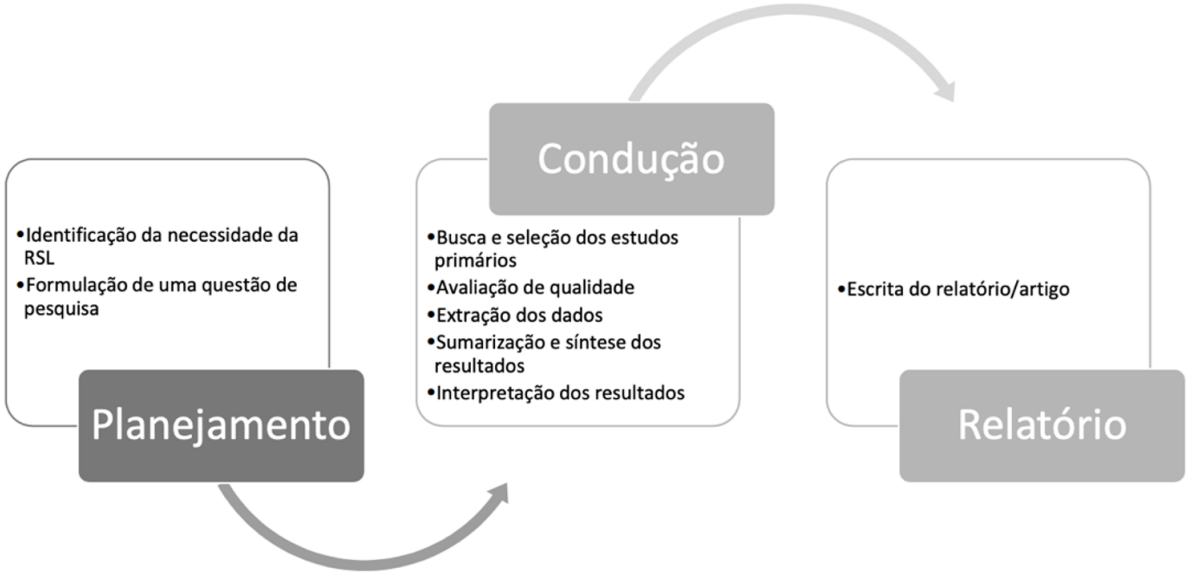

Figura 3.1: Fases da RSL
Fonte: Adaptada de Demerval *et al.* [79]

Como ilustrado na Figura 3.1, a RSL pode ser dividida em três fases principais [73]:

Planejamento, Condução e Relatório. No **Planejamento**, a necessidade de uma revisão é identificada, o objetivo, as fontes de pesquisa, as palavras-chave da busca, os CI, exclusão e de qualidade, a estratégia de extração e sintetização dos dados são definidos e o protocolo de revisão é desenvolvido e avaliado.

O protocolo de pesquisa é parte essencial da RSL, por isso, deve ser relatado antes da sua execução, como explicado na seção 3. Na fase da **Condução**, os seguintes passos são realizados: busca e seleção dos estudos primários, avaliação da qualidade do estudo, extração, sumarização e síntese dos resultados. Na última fase, o relatório da revisão é elaborado.

Segundo Biolchini *et al.* [82], as principais vantagens em utilizar a RSL são:

- evitar a duplicação de esforço: evitar que dois ou mais pesquisadores executem a mesma pesquisa simultaneamente;
- aumentar a abrangência da pesquisa: como as fontes e estratégias de busca são definidas no planejamento, a possibilidade de eliminação de estudos relevantes é reduzida;
- reduzir o viés: as buscas são feitas por expressões definidas no planejamento da pesquisa, a fim de minimizar a chance de que o pesquisador introduza informações enviesadas;
- aumentar a objetividade: os parâmetros e critérios relacionados à busca são definidos com antecedência, o que reduz, consideravelmente, a subjetividade da pesquisa;
- aumentar a confiabilidade da pesquisa: é possível executar novamente os passos do protocolo de revisão sistemática de literatura, para possibilitar que outros pesquisadores validem os resultados obtidos.

O Protocolo da Revisão Sistemática tem por objetivo reduzir o viés do pesquisador e identificar as informações, critérios e planejamentos necessários para a execução da RSL. O protocolo deve detalhar os objetivos, explicitar as questões de pesquisa e as estratégias de busca, definir os CI e exclusão de estudos, definir os procedimentos de avaliação de qualidade do estudo, detalhar a estratégia de extração dos dados e sintetizar os dados obtidos [83].

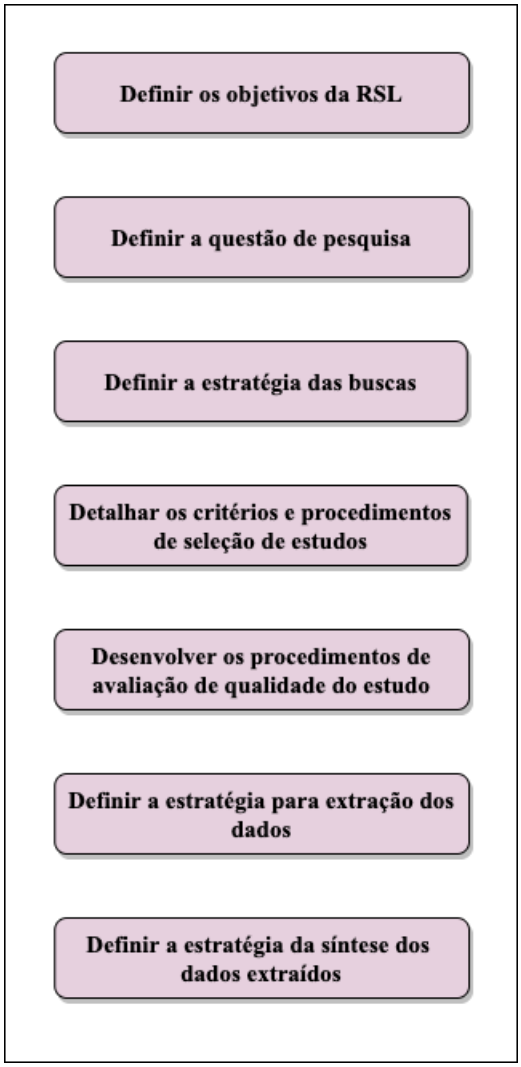

Figura 3.2: Procedimentos do protocolo da RS
Fonte: Adaptada de Demerval *et al.* [79]

Na Figura 3.2, estão ilustrados os procedimentos de definição do Protocolo da RSL conforme as seguintes atividades:

1. definir os objetivos da RSL: definição do objetivo almejado com a execução da RSL;
2. definir a questão de pesquisa: identificação das questões que se pretendem responder com a RSL;
3. definir a estratégia de busca: definição da estratégia a ser usada para pesquisar estudos primários, que incluem termos de pesquisa e recursos a serem pesquisados (banco de dados, periódicos específicos e anais de conferências);

4. detalhar os critérios e procedimentos de seleção de estudos: identificação dos critérios para incluir ou excluir um estudo da revisão sistemática. Geralmente, é útil testar os critérios de seleção em um subconjunto de estudos primários. O protocolo deve descrever como os critérios serão aplicados, por exemplo, quantos avaliadores validarão cada estudo primário e como as discordâncias entre eles serão resolvidas;

5. desenvolver os procedimentos de avaliação de qualidade do estudo: os pesquisadores devem identificar os critérios de verificação de qualidade para avaliar os estudos;

6. definir a estratégia para extração dos dados: detalhamento de como será feita a extração das informações buscadas nas fontes de dados. Também, deve detalhar as possíveis manipulações, inferências ou suposições a serem feitas;

7. definir a estratégia da síntese dos dados extraídos: essa definição deve esclarecer se uma meta-análise formal é necessária ou não e, em caso afirmativo, quais técnicas serão utilizadas.

As seções, a seguir, detalham a execução de cada fase da RSL executada neste trabalho. Para apoiar a organização da grande quantidade de dados gerada pela pesquisa, foi utilizada a ferramenta *State of the Art through Systematic Review* (StArt))[1].

3.1 Planejamento

O passo inicial para desenvolver uma RSL é declarar a justificativa que motive sua execução [80], pois a proposta gerada deve ser inovadora e útil para a resolução do problema de pesquisa. Além disso, essa proposta deve ser sustentada pelo conhecimento decorrente da RSL. Baseado nessas observações, o seguinte objetivo foi gerado: **identificar e analisar algoritmos frequentemente utilizados para o desenvolvimento de aplicações web adaptativas**. Não foram encontrados trabalhos anteriores que tenham executado uma RSL com objetivo semelhante, por essa razão, entende-se que esta revisão seja relevante.

3.1.1 Questões de Pesquisa

Uma RSL sempre busca responder uma Questão de Pesquisa (QP), que seja importante tanto para o pesquisador quanto para os profissionais que serão beneficiados pelos resultados da pesquisa [80]. A QP é mais bem definida quando embasada em três pontos de vista: população, intervenção e resultados. Segundo Uman [84], é possível sintetizar

[1] http://lapes.dc.ufscar.br/tools/start_tool

uma QP para uma RSL, da seguinte forma: Intervenção para uma População com uma Condição.

O acrônimo **PICO**, desenvolvido pela Cochrane Collaboration [84][85][86], é normalmente utilizado para a formação de uma QP e suas letras significam: P - **População**, I - **Intervenção**, C - **Comparação**, O - **Outcomes**. A **População** está relacionada aos grupos populacionais ou a uma área de aplicação. A **Intervenção** corresponde às ferramentas, às intervenções, às técnicas, às tecnologias ou aos métodos usados nos estudos primários. A **Comparação** é usada quando a pergunta a ser respondida envolve comparação entre métodos. Já os Resultados, ou ***Outcomes***, remetem à finalidade dos estudos e seus resultados [85][86].

Neste trabalho, o acrônimo PICO foi utilizado da seguinte forma:

- **População**: trabalhos ou projetos que busquem desenvolver aplicações web, por meio de conceitos de interface adaptativa;
- **Intervenção**: a intervenção a ser estudada é uma interface adaptativa, baseada no conhecimento gerado a partir dos dados cadastrados pelo usuário (contexto e perfil) e dados produzidos durante a navegação nas páginas web (interações);
- **Comparação**: esta RSL tem a intenção de comparar os diversos algoritmos capazes de extrair conhecimento, a partir dos dados e navegação do usuário para o desenvolvimento de interfaces adaptativas;
- **Resultados**: após a execução desta RSL, será selecionado um algoritmo, dentre os diversos existentes, para ser utilizado no desenvolvimento de aplicações adaptativas. O algoritmo selecionado deverá compor a arquitetura proposta por este trabalho.

Com base nestas definições, foi possível selecionar a seguinte QP, a ser respondida durante a execução desta RSL: **QP - Quais são os algoritmos frequentemente utilizados para o desenvolvimento de aplicações web adaptativas?**

A fim de executar a análise de respostas da questão principal com mais precisão, é possível que a QP seja dividida em Questões Secundárias (QS) e mais específicas [87]. Assim, as QS definidas para esta questão de pesquisa são apresentadas no Quadro 3.1.

Quadro 3.1: Questões Secundárias de Pesquisa (QS)

QS	**Descrição**
QS.1	Quais dados são considerados pelo algoritmo para a tomada de decisão?
QS.2	Quais as tecnologias de software (linguagens de programação, *frameworks*, ambientes etc.) utilizadas para a renderização da página web adaptativa?
QS.3	Qual técnica foi utilizada para extrair os dados dos usuários?
QS.4	Quais modelos, tarefas, técnicas e/ou algoritmos são utilizados para analisar os dados extraídos das interações dos usuários?
QS.5	Houve melhoria na experiência do usuário após a utilização do algoritmo e da página adaptativa?

Fonte: Elaborado pelo autor

A seguir, apresentam-se as estratégias utilizadas nesta pesquisa.

3.1.2 Estratégias de Pesquisa

Como apresentado na Figura 3.3, este trabalho adotou duas estratégias de pesquisa, em que a primeira são buscas automáticas nas bases de dados eletrônicas, por meio de uma *string* de busca. Para evitar o viés de publicação, o qual ocorre motivado pela tendência das publicações de trabalhos com resultados positivos, e para encontrar trabalhos que não foram alcançados por meio da busca automática, foi necessário realizar uma segunda estratégia, a busca manual, em algumas das mais importantes conferências e revistas que versam acerca de IHC. O uso de buscas manuais é defendido por alguns pesquisadores na condução de RSL, por exemplo Lumsden [88].

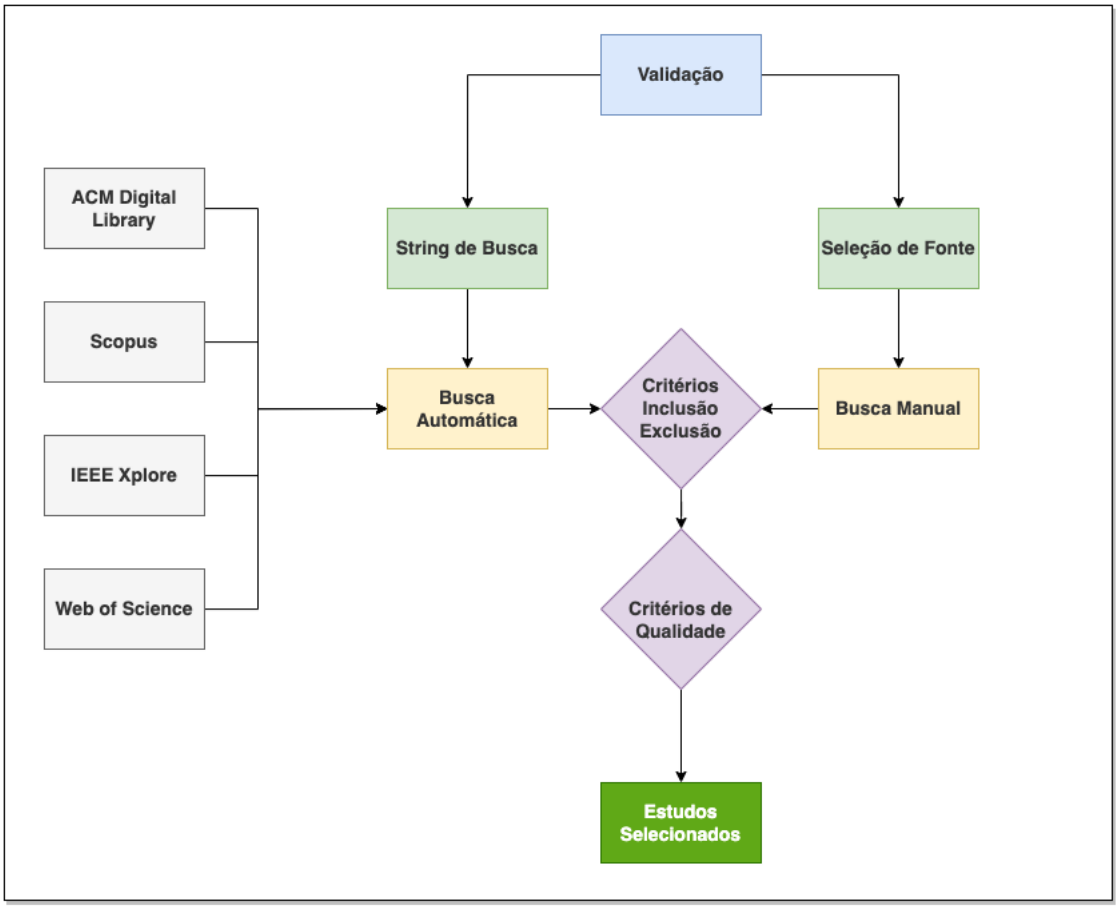

Figura 3.3: Estratégia de busca para seleção de estudos relevantes
Fonte: Elaborada pelo autor

Em relação à busca manual, foram considerados dois congressos e três periódicos que versam acerca de interação IHC. Os periódicos foram escolhidos com base no seu fator de impacto; e as conferências, com base na sua relevância para a área. O Quadro 3.3 detalha as fontes utilizadas nesta busca manual. Artigos de outros periódicos relevantes, por exemplo, UAIS e ACM SIGCHI, foram alcançados pela busca automática.

3.1.3 String de Busca

É importante que a *string* de busca compreenda palavras-chave que alcancem a maior quantidade de trabalhos relevantes possíveis [80][84], durante a busca automática. Dessa forma, é possível recorrer às definições dadas para cada letra do acrônimo PICO para a escolha dessas palavras-chave.

- **População:** "web", "www", "application", "interface";

- **Intervenção:** "adaptive interface", "customized interface", "personalized interface", "intelligent interface", "smart interface";
- **Comparação:** "algorithm", "technique", "approach", "paradigm";
- **Resultados:** "usability", "user experience", "user satisfaction".

Após a definição das palavras-chave relativas a cada letra do acrônimo PICO, foi possível reorganizar cada termo em cinco grupos, conforme apresentado no Quadro 3.2, em que cada grupo contém termos que são sinônimos, formas diferentes da mesma palavra ou termos que têm significado semântico semelhante ou relacionado no domínio. A implementação dessa estratégia de busca pode ser alcançada, ao aplicar os operadores *AND* e *OR*, em que o *OR* pode ser usado nos grupos e o *AND* entre os grupos [89]. A fim de facilitar a visualização, as palavras no plural não foram incluídas na tabela de termos agrupados; mas, sim, na *string* de busca final.

Com base nestas palavras-chave, foi gerada a seguinte *string* de busca: **("algorithm"OR "algorithms"OR "technique"OR "techniques"OR "approach"OR "approaches"OR "paradigm"OR "paradigms") AND ("adaptive *interface?"OR "customized *interface?"OR "personalized *interface?"OR "intelligent *interface?"OR "smart *interface?") AND ("usability"OR "user *experience"OR "user *satisfaction") AND ("user data"OR "interaction"OR "interactions"OR "user profile"OR "behavior"OR "behavior"OR "behavioral") AND ("web"OR "www"OR "page"OR "application"OR "applications"OR "interface"OR "interfaces")**.

Quadro 3.2: Termos agrupados por similaridade utilizados em conjunto

Palavra	**G1**	**G2**	**G3**	**G4**	**G5**
algorithm	*				
technique	*				
approach	*				
paradigm	*				
adaptive interface		*			
customized interface		*			
personalized interface		*			
intelligent interface		*			
smart interface		*			
usability			*		
user experience			*		
user satisfaction			*		
user data				*	
user interaction				*	
user profile				*	
behavior				*	
web					*
application					*
interface					*
www					*

Fonte: Elaborado pelo autor

Os critérios de seleção adotados são apresentados a seguir.

3.1.4 Critérios de Seleção

Os seguintes critérios de seleção foram definidos para filtrar os trabalhos utilizados para responder às questões de pesquisa:

Critérios de Seleção das Bases (CSB)

Existem diversas bases eletrônicas aptas a serem escolhidas para a execução de uma RSL, de modo que, dada a limitação de tempo e computacional, é impossível consultar todas. Portanto, as bases devem ser, criteriosamente, escolhidas, de acordo com o escopo da pesquisa. Os CSB para esta pesquisa foram:

1. a base deve permitir a exportação em Bibtex;
2. a base deve permitir consultas automatizadas;

3. a base deve ser acessada por intermédio do Portal de Periódicos da Capes, em convênio com a Universidade de Brasília (UnB);

4. a base deve permitir a filtragem por ano;

5. a base deve estar contida na lista de fontes do StArt [90];

6. a base deve contemplar trabalhos relacionados à aplicação web adaptativa.

Diante desse contexto, as bases escolhidas para este trabalho foram: Scopus, IEEE Xplore, ACM Digital Library e Web of Science.

Critérios de Inclusão (CI)

É importante seguir critérios para tornar a pesquisa realizável, dentro da limitação de recursos existentes; e obter um conjunto de trabalhos que consiga responder às questões de pesquisa. Nesse sentido, foram considerados trabalhos e estudos que apresentem

1. a entrega dos artefatos produzidos durante o desenvolvimento do tema;

2. métodos e técnicas para desenvolver aplicações web adaptativas, que considerem os dados ou as interações do usuário;

3. métodos e técnicas para analisar os dados dos usuários em aplicações web;

4. o resultado validado por um estudo de caso prático ou deve ser utilizado em alguma organização;

5. estudos apresentados em inglês;

6. data de publicação entre os anos 2011 e 2022.

Os estudos precisaram contemplar todos os Critérios de Inclusão (CI) para serem utilizados na fase de extração de dados.

Critérios de Exclusão (CE)

Os () foram definidos da seguinte maneira:

1. trabalhos fora do escopo de aplicações web adaptativas;

2. estudos que não apresentem um algoritmo para a criação da aplicação web adaptativa;

3. o trabalho não apresenta um estudo empírico, em relação ao algoritmo proposto;

4. o algoritmo proposto por meio da abordagem não é funcional;

5. não foi possível ter acesso ao artigo completo, de maneira gratuita, por intermédio das bases de dados;

6. estudos duplicados.

Os trabalhos identificados com pelo menos um critério de exclusão foram rejeitados.

3.1.5 Critérios de Qualidade

Após a execução da primeira iteração da seleção dos estudos primários e por meio da definição de Critérios de Qualidade (CQ), é possível classificar estudos pela qualidade e relevância, de acordo com o tema proposto [80]. Segundo [91], esses critérios podem ser definidos em três categorias: alto, médio e baixo, considerando as múltiplas dimensões de aplicação. Neste trabalho, os estudos classificados como nível baixo, em pelo menos uma das dimensões, foram rejeitados. As dimensões de qualidade consideradas foram:

- qualidade da execução do estudo: relativo à capacidade de o estudo de se adequar aos padrões científicos aplicados pela área técnica científica;
- adequação à questão de pesquisa da revisão: relativo à aplicabilidade do estudo em responder, adequadamente, à QP;
- adequação ao foco da revisão: relativo à semelhança do contexto ao qual os resultados poderão ser aplicados.

Vale evidenciar que, do ponto de vista contextual, o resultado da pesquisa será útil para o desenvolvimento de uma proposta de arquitetura para o Banco Beta, uma vez que essa arquitetura proporcionará o desenvolvimento de aplicações web adaptativas para melhorar a experiência do usuário.

3.1.6 Validação do Protocolo

A RSL é um método consistente e deve propiciar conhecimento preciso, sistematizado e fundamentado em critérios rigorosos; por isso, a revisão do processo por outro pesquisador é essencial no direcionamento de uma prática fundamentada em conhecimento científico [73]. O revisor deve entender, claramente, o propósito e o escopo da RSL, para, assim, verificar a adequação do material à pergunta a ser respondida. Diante disso, um segundo pesquisador analisou e revisou o protocolo de pesquisa utilizado neste trabalho. Foi realizada uma reunião em que o revisor utilizou a ferramenta StArt [90] para reproduzir os

passos descritos na pesquisa principal e a lista de trabalhos avaliados foi idêntica à lista descrita neste trabalho.

Uma busca piloto foi feita na IEEE Xplore, antes de agrupar as palavras, de acordo com o Quadro 3.2, o que resultou em múltiplos estudos não relacionados ao tema. Esse agrupamento de palavras semelhantes ou usadas frequentemente juntas ajudou a recuperar estudos mais condizentes com o objetivo da pesquisa.

Além disso, foi realizada uma busca piloto, que retornou muitos estudos relacionados a ambientes distintos da web, considerados em desacordo com o objetivo deste trabalho. Isso ocorre porque há uma rápida e constante evolução de tecnologia em dispositivos eletrônicos, que faz com que esse tema seja bastante estudado. Consequentemente, o operador *NOT* e as palavras *tv, vehicle, car, voice, sound, transport, medicine, glass, sensor* foram incluídas na *string*, para que os artigos que continham essas palavras fossem removidos do resultado. Assim, a *string* final adotada para a busca foi: **("algorithm"OR "algorithms"OR "technique"OR "techniques"OR "approach"OR "approaches"OR "paradigm"OR "paradigms") AND ("adaptive *interface?"OR "customized *interface?"OR "personalized *interface?"OR "intelligent *interface?"OR "smart *interface?") AND ("usability"OR "user *experience"OR "user *satisfaction") AND ("user data"OR "interaction"OR "interactions"OR "user profile"OR "behavior"OR "behavior"OR "behavioral") AND ("web"OR "www"OR "page"OR "application"OR "applications"OR "interface"OR "interfaces") AND NOT ("tv"OR "vehicle"OR "vehicles"OR "car"OR "cars"OR "voice"OR "voices"OR "sound"OR "sounds"OR "transport"OR "transports"OR "medicine"OR "glass"OR "glasses"OR "sensor"OR "sensors")**.

3.2 Condução

Baseado na definição do protocolo de revisão, foi possível executar as tarefas da fase de condução. A primeira etapa dessa fase é a seleção dos estudos primários, a qual será apresentada na seção a seguir.

3.2.1 Seleção dos Estudos Primários

A *string* estabelecida no protocolo foi submetida à execução da Busca Automática, e, como os sistemas computacionais tendem a se tornar obsoletos rapidamente [92], o horizonte temporal foi delimitado a dez anos - correspondente ao intervalo de 2011 a 2022. Assim, esta revisão abrange a literatura mais recente a respeito do tema, que recuperou trabalhos que já atualizaram abordagens obsoletas. Além disso, para possibilitar uma

futura replicabilidade, a revisão se limitará às publicações escritas em inglês. Na ferramenta da base IEEE Xplore, a busca foi configurada para ser submetida nos campos *Full Text & Metadata*, que resultou em **128 estudos**. Na ACM Digital Library, a *string* foi reescrita para melhor utilização dos operadores permitidos nessa plataforma, como o *wildcard* "?", que significa um ou zero caracteres. Outra diferenciação relevante diz respeito à quantidade de *wildcards*: a plataforma IEEE Xplore permite o uso de somente sete *wildcards*, enquanto a ACM Digital Library não impõe esse limite. A *string* utilizada para esta plataforma foi **("algorithm?"OR "technique?"OR "approach*"OR "paradigm?") AND (adapti*interface? OR customized*interface? OR personalized*interface? OR intelligent*interface? OR smart*interface?) AND ("usability"OR user*experience OR user*satisfaction) AND ("user data" OR "interaction?"OR "user profile"OR "behavi*") AND ("web"OR "www"OR "page"OR "application?"OR "interface?") AND NOT ("tv"OR "vehicle?"OR "car?"OR "voice?"OR "sound?"OR "transport?"OR "medicine"OR "glass*"OR "sensor?")**, que resultou em **37 estudos**.

A *string* foi reescrita da seguinte forma para ser executada na fonte Web of Science: **ALL=(algorithm$ OR technique$ OR approach* OR paradigm$) AND ALL=("adaptive *interface$"OR "customized *interface$"OR "personalized *interface$"OR "intelligent *interface$"OR "smart *interface$") AND ALL=(usability OR "user *experience"OR "user *satisfaction") AND ALL=("user data"OR "interaction$"OR "user profile"OR behavio*) AND ALL=("web"OR "www"OR "page"OR "application$"OR "interface$") NOT ALL=("tv"OR "vehicle$"OR "car$"OR "voice$"OR "sound$"OR "transport$" OR "glass*"OR "sensor$")**, e a execução resultou em **18 estudos**. A reescrita foi realizada, pois foi necessário adicionar o operador "ALL"para que a busca fosse realizada em todo o texto e não somente no título, resumo e palavras-chave.

Na base da Scopus, a *string* foi alterada para seguir as regras de sintaxe, adicionar a limitação de data e limitar em artigos de acesso aberto. Assim, obteve-se a seguinte forma: **ALL(("algorithm"OR "algorithms"OR "technique"OR "techniques"OR "approach"OR "approaches"OR "paradigm"OR "paradigms") AND ("adaptive *interface?"OR "customized *interface?"OR "personalized *interface?"OR "intelligent *interface?"OR "smart *interface?") AND ("usability"OR "user *experience"OR "user *satisfaction") AND ("user data"OR "interaction"OR "interactions"OR "user profile"OR "behavior"OR "behaviour"OR "behavioral") AND ("web"OR "www"OR "page"OR "application"OR "applications"OR "interface"OR "interfaces") AND NOT ("tv"OR "vehicle"OR "vehicles"OR "car"OR "cars"OR "voice"OR "voices"OR "sound"OR "sounds"**

OR "transport"OR "transports"OR "medicine"OR "glass"OR "glasses"OR "sensor"OR "sensors")) AND PUBYEAR > 2010 AND PUBYEAR < 2022 AND (LIMIT-TO (OA,"all")), o que resultou em **88 estudos**.

A busca manual realizada nas fontes detalhadas no Quadro 3.3 resultou em **23 estudos**.

Quadro 3.3: Conferências e Periódicos consultados na Busca Manual

Sigla	**Journal / Conferência**	**Endereço Eletrônico**
IJHCS	International Journal of Human Computer Studies	https://journals.elsevier.com/ international-journal-of-human-computer-studies
IWC	Interacting with Computers	https://academic.oup.com/iwc
TOCHI	Transaction on Computer-Human Interaction	https://dl.acm.org/journal/tochi
INTERACT	IFIP Interact	https://link.springer.com/conference/ interact
W4A	Cross-Disciplinary Conference on Web Accessibility	https://dl.acm.org/conference/w4a

Fonte: Elaborado pelo autor

A volumetria dos estudos encontrados em cada fonte é ilustrada na Figura 3.4.

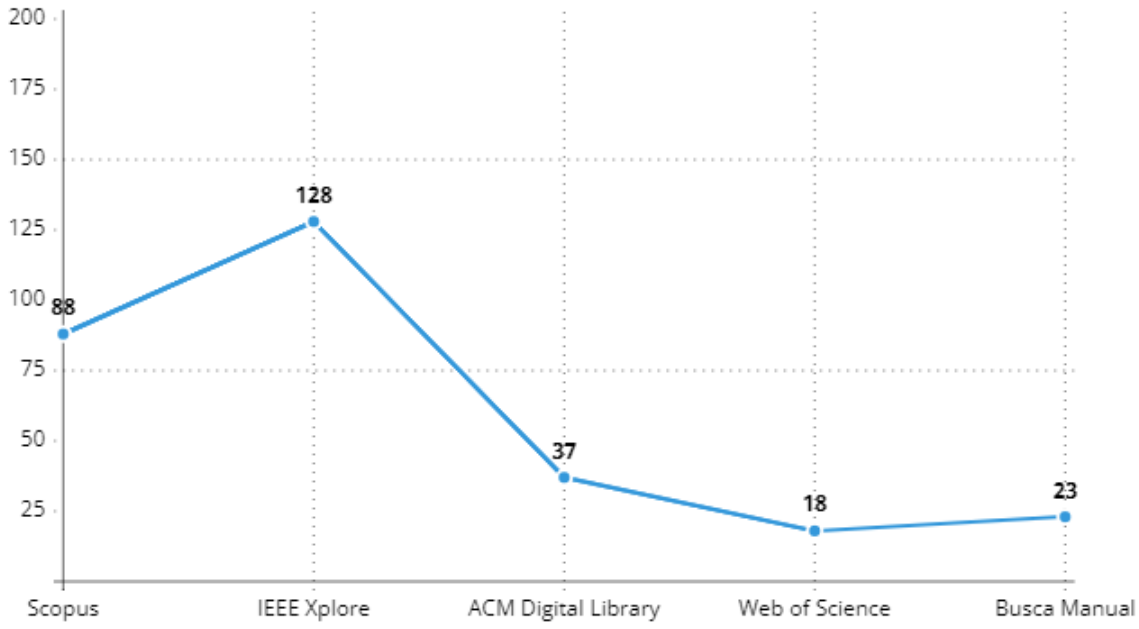

Figura 3.4: Fluxo do processo de condução desta revisão

Fonte: Elaborada pelo autor

Em suma, as buscas retornaram **294 estudos primários**, dos quais 13% dos resultados vieram da fonte ACM Digital Library; 30% da busca na Scopus; 44% da busca na IEEE Xplore; 6% da busca na Web of Science; e 7% da busca manual nos anais das conferências e periódicos.

A Figura 3.5 mostra que o tema foi bastante estudado nos últimos dez anos, o que evidencia a relevância desta RSL para obtenção de algoritmos que busquem proporcionar uma interface adaptativa ao usuário. Em 2012, houve uma pequena queda no número de artigos publicados; porém, nos demais anos, os números foram superiores a 20 artigos, o que corresponde a, aproximadamente, dois artigos por mês.

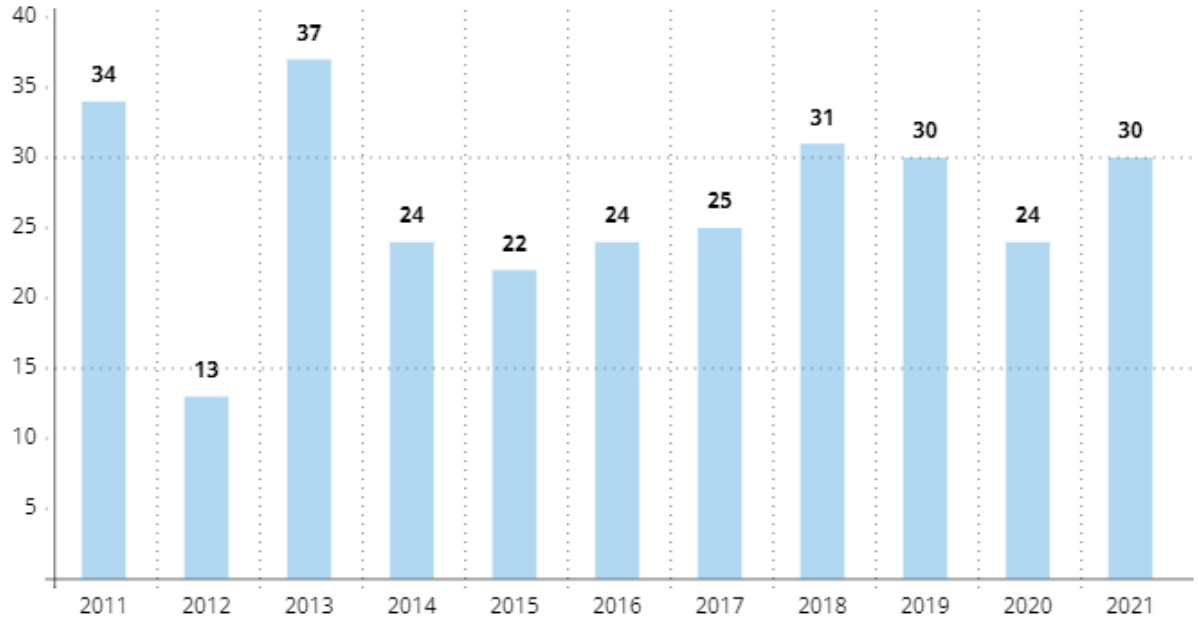

Figura 3.5: Artigos acerca de algoritmos para aplicações web adaptativas
Fonte: Elaborada pelo autor

A seleção dos estudos primários ocorreu por meio da utilização dos CI e de exclusão apresentados na Seção 3.1.4, executada em três etapas (3.6). Na primeira, os estudos foram avaliados com base na leitura do título, resumo e palavras-chave, com um total de **185 estudos selecionados**. Na segunda etapa, os estudos foram aceitos e/ou rejeitados com base nos CI e de exclusão, que resultaram na seleção de **46 estudos**. Na terceira etapa, os estudos foram selecionados e/ou rejeitados, a partir da leitura completa do artigo e aplicação dos critérios de qualidade descritos na Seção 3.1.5, que possibilitou a seleção de **10 estudos**, os quais foram utilizados para responder às questões de pesquisa propostas (fase de extração de dados).

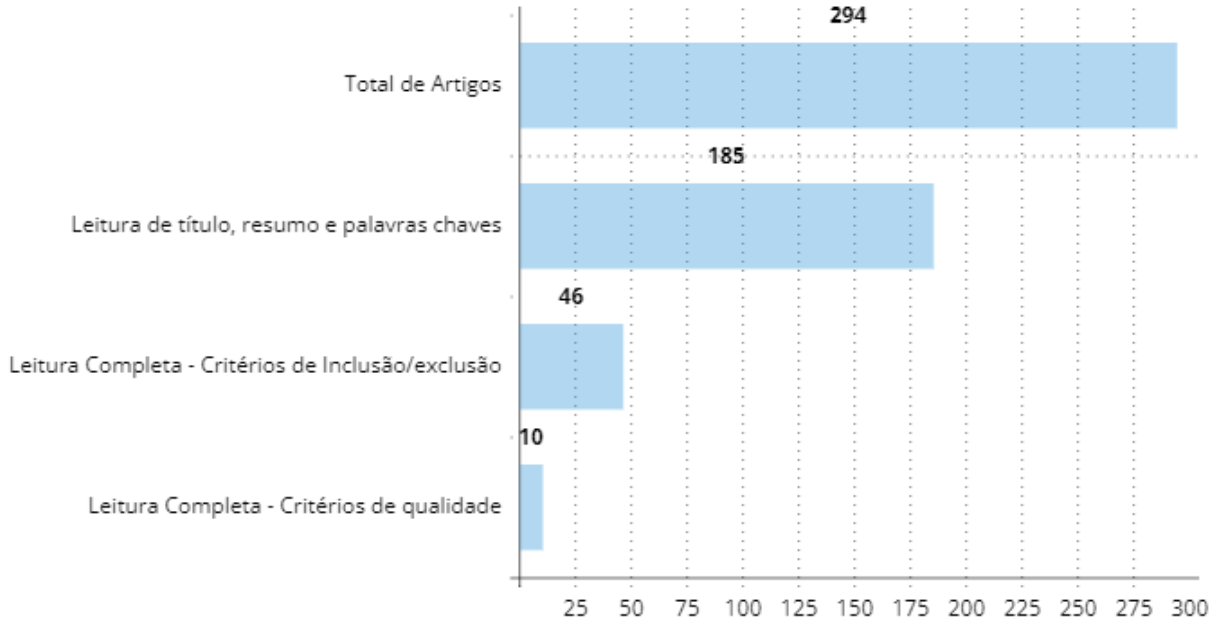

Figura 3.6: Histórico do volume de artigos em cada etapa de seleção
Fonte: Elaborada pelo autor

Descrevem-se, a seguir, os passos para a extração de dados.

3.2.2 Extração de Dados

A etapa de extração de dados foi conduzida com auxílio da ferramenta StArt [90]. As funções dessa ferramenta permitem agregar, organizar e classificar os estudos selecionados.

3.3 Resultados

Com o objetivo de responder às questões de pesquisa descritas no protocolo, foi realizada uma extração de dados nos estudos selecionados. A função da StArt [90] de criar questões a serem respondidas após a leitura detalhada de cada estudo auxiliou na extração dos dados. Baseado nas questões secundárias descritas no Quadro 3.1, as questões respondidas na ferramenta StArt foram:

1. Em qual ano o artigo foi publicado? (CE 6)
2. Em qual fonte o artigo foi publicado? (CSL 5)
3. Em qual canal o artigo foi publicado? (CSL 3)

4. Quais dados dos usuários foram utilizados pelo algoritmo para a tomada de decisão? (QS 1)
5. Qual técnica foi utilizada para extrair os dados dos usuários? (QS 3)
6. Qual linguagem de programação foi utilizada para a implementação do algoritmo? (QS 4)
7. Qual *framework* foi utilizado para auxiliar a implementação do algoritmo? (QS 4)
8. Qual foi a abordagem utilizada pelo algoritmo? (QS 4)
9. Qual foi o algoritmo utilizado? (QS 4)
10. Qual *framework* foi utilizado para auxiliar a implementação da página web? (QS 2)
11. Houve avaliação da experiência do usuário após a implementação do algoritmo? (QS 5)
12. Houve melhoria da experiência do usuário após a implementação do algoritmo? (QS 5)

As respostas dessas perguntas ajudaram a responder as Questões Secundárias (QS), os CI, os CE, os Critérios de Seleção de Base (CSL) e, consequentemente, a QP principal. As respostas foram extraídas dos estudos, após a seleção final, as quais são apresentadas no Quadro 3.4. A coluna Código é uma palavra única, que identifica cada estudo selecionado; a coluna Ref indica a referência bibliográfica; a coluna Ano indica em que ano o estudo foi publicado; a coluna Canal descreve o canal no qual o estudo foi publicado, que pode ser conferência ou anais de congressos (C) ou em Jornais (J); a coluna Descrição contém uma pequena explicação do estudo; e a coluna QS indica quais QS foram respondidas por meio do respectivo estudo.

Quadro 3.4: Lista de artigos selecionados, após a execução da RSL ordenado pelo ano da publicação

Código	**Ref**	**Ano**	**Canal**	**Descrição**	**QS**
E1	[93]	2011	J	Apresenta uma estrutura baseada no algoritmo heurístico e na ontologia existente. Foi aplicada uma avaliação de utilização na abordagem proposta. Os resultados mostraram que os usuários perceberam as alterações de interface automáticas e classificaram-nas como adequadas e satisfatórias.	QS.1, QS.4, QS.5
E2	[94]	2015	C	Apresenta uma interface web adaptativa, com base no modelo de usuário Bayesiano e dá importância especial à conexão entre computação em nuvem e a interface de usuário adaptativa.	QS.1, QS.2, QS.4
E3	[95]	2016	J	Apresenta o *Role-Based UI Simplification* (RBUIS) como um mecanismo para melhorar a usabilidade, por meio do comportamento adaptativo, que fornece, aos usuários finais, um conjunto mínimo de recursos e um layout ideal, com base no contexto de uso. Foi realizado um estudo de usabilidade para avaliar se as interfaces simplificadas com RBUIS resultam em melhorias em relação às interfaces comumente utilizadas.	QS.1, QS.2, QS.3, QS.4, QS.5
E4	[96]	2017	C	Proposta de uma arquitetura para interface adaptativa, que aproveita os desenvolvimentos em aprendizado profundo, técnicas de filtragem colaborativa e aproximação rápida mais próxima, com base nos métodos do vizinho mais próximo em espaços euclidianos.	QS.1, QS.3, QS.4

continua na próxima página

continuação

Código	Ref	Ano	Canal	Descrição	QS
E5	[97]	2017	C	O desenho universal representa um desafio difícil para o *designer*, pois depende da complexidade das intenções humanas em um determinado tempo e lugar. Por esta razão, este artigo propõe uma nova abordagem que visa apoiar o *design* de ambientes inclusivos, a fim de melhorar a interação usuário-ambiente.	QS.1, QS.4
E6	[98]	2018	J	Apresenta contribuições para tratar dados em tempo real. A primeira contribuição utiliza *feedback* em tempo real, gerado por um modelo probabilístico de previsão de tarefas para construir um sistema adaptativo em tempo real. A segunda propõe duas novas abordagens de visualização adaptativa, que consideram a presença de incerteza nas saídas dos modelos de previsão. A terceira apresenta um método de personalização para sugerir qual abordagem será mais adequada para cada usuário. A quarta quantifica os efeitos das abordagens de visualização propostas e erros de previsão no comportamento do usuário.	QS.1, QS.3, QS.4
E7	[99]	2019	J	Propõe uma arquitetura baseada em microsserviços, que gera conjuntos de dados que contém o comportamento do usuário para análise posterior. Portanto, a experiência do usuário e a usabilidade em interfaces de usuário distribuídas podem ser aprimoradas por meio de modelos de previsão gerados a partir dos dados. A arquitetura oferece uma infraestrutura para lidar e processar a interação de dados de aplicativos entre dispositivos com várias formas de interação. Essa abordagem foi implantada em uma aplicação de *mashup* real no qual novos conjuntos de dados foram criados, processados e validados.	QS.1, QS.3, QS.4

continua na próxima página

continuação

Código	Ref	Ano	Canal	Descrição	QS
E8	[100]	2019	C	Apresenta uma plataforma de software genérica para geração automática de interface do usuário adaptativa, por meio da análise de padrões de comportamento do usuário e customização de interfaces de usuário, por meio de aprendizado de máquina. O classificador apresentado mostrou 100% de precisão para grandes componentes de interface e cenários de usuário.	QS.1, QS.3, QS.4, QS.5
E9	[101]	2019	J	Este artigo contribui com modelos preditivos individualizados de pesquisa visual e uma abordagem computacional para reestruturar *layouts* gráficos para um usuário individual, de modo que os recursos em uma interface nova e não visitada possam ser encontrados mais rapidamente. Dado um histórico de interfaces vistas anteriormente, o *layout* é reestruturado com o objetivo de tornar seus recursos mais facilmente localizáveis.	QS.1, QS.2, QS.4
E10	[42]	2021	C	Proposta de uma nova abordagem para o desenvolvimento de interfaces do usuário adaptativas, que predizem se a mudança a ser realizada melhorará ou não a usabilidade. Caso o algoritmo identifique que melhorará, a interface será alterada. Essa abordagem utiliza o método de aprendizado por reforço baseado em modelo. Apresenta resultados empíricos e de simulação no caso de menus adaptativos.	QS.1, QS.2, QS.4, QS.5

Fonte: Elaborado pelo autor, com os dados da pesquisa

3.3.1 QS.1 — Quais dados são considerados pelo algoritmo para a tomada de decisão?

Sem os dados, não é possível fazer o treinamento de modelos de *machine learning* ou validação de algoritmos heurísticos. Muitos projetos não se concretizam pela falta de dados relevantes ou pela dificuldade e demora do processo de coleta. A escolha do algoritmo

para predizer uma ação ou gerar a melhor composição de componentes em uma página web é um fator importante para atingir um resultado positivo. Não menos importante é a seleção dos dados avaliados pelo algoritmo para gerar a decisão correta [102]. Diante disso, é importante analisar quais dados os estudos selecionados utilizaram para gerar aplicações web adaptativas, a fim de analisar a viabilidade da utilização do algoritmo selecionado na arquitetura proposta por este trabalho.

O estudo apresentado por Soh *et al.* [96] considerou não apenas os usuários e os componentes da tela (controle de UI e elementos de conteúdo), mas também o histórico de interações do usuário. Para realizar os experimentos, foram utilizadas as bases de dados Equation Grapher UI (EqGraph) [103], Microsoft Web Data (MSWeb) [104] e o conjunto de dados ASSISTments [105].

Já Todi *et al.* [42] utilizaram os cliques nos itens do menu para aproximar a experiência e o interesse dos usuários, que influenciam, diretamente, na ordenação do menu, ou seja, os itens mais interessantes para o usuário aparecem primeiro.

No estudo realizado por Rathnayake *et al.* [100], além do clique, também se considerou a movimentação do mouse para destacar a importância de um componente para o usuário, assim, um mapa de calor é gerado à medida que cursor do mouse percorre os componentes renderizados na interface.

Todi *et al.* [101] foram além; pois, apesar de utilizar o clique e movimento do mouse, também consideraram informações acerca do olhar do usuário, que inclui a posição dos olhos e fixações do olhar, por meio do equipamento EyeTribe Tracker[2].

No estudo apresentado por Gullà *et al.* [97], não só as interações do usuário na aplicação foram consideradas, mas também dados do contexto, como status dos dispositivos, medições dos sensores e informações do clima no ambiente físico.

Na aplicação apresentada por Akiki *et al.* [95], o algoritmo detecta novas situações e mudanças de comportamento do usuário para alterar a composição da interface, por exemplo, tamanho da tela, taxa de uso dos campos de entrada, novas atualizações instaladas na plataforma e informações coletadas sobre o ambiente, a partir da utilização de sensores. A aquisição dos dados é uma tarefa complexa, o sistema precisa suportar múltiplas interações por segundo, produzidas por múltiplos usuários simultaneamente. Por isso, Fernandez Garcia *et al.* [99] criaram uma arquitetura com base em conceitos de microsserviços para armazenar e processar dados advindos de dispositivos heterogêneos, como notebooks, computadores, celulares e *tablets*, por meio de interfaces de mouse, teclado, gestos ou voz.

Kong *et al.* [93] fizeram uma proposta para criação de interfaces multimodais, por meio de dados da camada de aplicação (contexto de interação do usuário, dispositivo e

[2]https://www.theeyetribe.com

ambiente), da camada do sistema (recursos) e da camada de rede (requisitos de Qualidade de Serviço (QoS)). Por exemplo, o movimento dos olhos, a tela multitoque, o gesto da mão e um microfone.

Por fim, Çağla Çığ Karaman e Sezgin [98] apresentaram uma abordagem para gerar interfaces inteligentes, que podem ajudar os usuários, ao gerar comandos com base em um modelo probabilístico previamente aprendidos por meio do olhar. O sistema fornece feedbacks para gerar interfaces adaptativas em tempo real.

3.3.2 QS.2 - Quais as tecnologias de software (linguagens de programação, *frameworks*, ambientes etc.) utilizadas para a renderização da página web adaptativa?

As propostas que os estudos selecionados apresentaram utilizam algoritmos que exigem processamento computacional considerável, o que inviabiliza a possibilidade de execução pelo dispositivo do usuário (*frontend*). Todavia, a arquitetura proposta por este estudo também contemplará a interface do usuário, por isso, é importante analisar como os estudos selecionados desenvolveram a interface para ser adaptada a cada usuário que interaja com a aplicação, conforme o resultado da execução do algoritmo.

Todi *et al.* [42] utilizaram um Apple Magic Mouse[3] com velocidade de rastreamento padrão para que o usuário interagisse com a interface do menu desenvolvido. O menu foi implementado utilizando *Hypertext Markup Language* (HTML) e Javascript e exibido em uma janela do navegador. O registro dos movimentos do mouse, assim como a data e hora, foram enviados para o algoritmo por requisições desenvolvidas com Javascript, sem o envolvimento de *frameworks*.

Com uma abordagem similar, Rim *et al.* [94] desenvolveram uma interface utilizando HTML5, sob uma justificativa de possibilitar uma experiência rica para os usuários, além de possibilitar uma aparência agradável e intuitiva. Nesse caso, a interface é executada pelo navegador, porém recebe informações em tempo real de uma série de aplicações executadas na nuvem (*backend*).

O estudo publicado por Todi *et al.* [101] destaca o desenvolvimento da interface. Foi apresentada uma aplicação executada pelos navegadores, que permite que os usuários visitem a aplicação por diversos dispositivos. Assim que um usuário acessa a página, a origem da página (HTML) é analisada pelo sistema. Caso o algoritmo indique que os componentes devam ser alterados, uma nova página é reestruturada, com base nos valores retornados pelo algoritmo, e os componentes são reposicionados. Além disso, foi utilizado

[3]https://www.apple.com/br/

o Familiariser[4], um navegador que reestrutura dinamicamente os layouts de sites para torná-los mais fáceis de usar, ao resolver sobreposições de componentes, enquanto tenta mantê-los alinhados sem obscurecer ou omitir conteúdo.

Por fim, Akiki *et al.* [95] adotaram uma abordagem de posicionamento relativo para o layout da interface do usuário, em que os componentes podem ser incorporados um dentro do outro. Por exemplo, caixas de texto dentro de uma caixa de grupo e posicionadas, que usam as propriedades de posição superior e esquerda. Essa abordagem é suportada por muitas tecnologias de apresentação, como: HTML, Java Swing e Windows Forms. Cada componente tem um tipo de elemento gráfico como: botão, caixa de texto etc. Esses tipos definem uma propriedade que indica onde outro componente está localizado na interface, de modo a posicionar-se, em relação aos demais componentes. Os usuários finais podem acessar a interface do usuário, por meio de links, em uma estrutura de navegação, como um menu.

3.3.3 QS.3 - Qual técnica foi utilizada para extrair os dados dos usuários?

Existem diversas formas de extrair os dados do usuário para que o algoritmo processe e retorne um *feedback* para a aplicação adaptativa. Quando se trata de aplicações web, os dados podem ser enviados por meio de requisições HTTP ou orientado a eventos, com o uso de Websocket [106], por exemplo. É importante entender como os estudos selecionados realizaram essa integração, pois essa etapa também é objeto de estudo da arquitetura proposta no presente trabalho.

Para validar o algoritmo proposto, Soh *et al.* [96] utilizaram três bases de dados com históricos de navegações públicas já conhecidas pela comunidade acadêmica. A primeira foi a EqGraph [103], que contém 51 usuários, 16 itens e 1.277 interações. Cada interação compreende símbolos que representam uma das 16 regiões de área de interesses, correspondentes aos elementos da interface do usuário. A segunda foi a Microsoft Web Data (MSWeb) [104], que contém 1.205 usuários, 246 itens e 6.623 interações. A terceira e última foi a ASSISTments, que contém 4.130 usuários, 100 itens e 51.318 interações.

Já no trabalho do Rathnayake *et al.* [100], foi desenvolvida uma aplicação web que utiliza autenticação JavaScript Web Token (JWT) para que as requisições sejam realizadas de maneira segura. A captura dos dados utiliza APIs REST [107] para enviar as informações a respeito das interações dos usuários. Os componentes da interface do usuário se comunicam com o servidor para analisar e executar os dados. A comunicação é realizada por funções JavaScript, quando o mouse é movimentado pelo usuário. O servidor

[4]https://www.kashyaptodi.com/familiarisation/

responde à requisição com uma nova configuração de página para ser renderizada pelo navegador web. Essa configuração contém componentes ativados e desativados, além da ordem em que devem ser apresentados na interface.

O trabalho apresentado por Fernandez Garcia *et al.* [99] segue a mesma linha para a captura e extração dos dados, em que um conjunto de aplicações foi desenvolvido com base na arquitetura de microsserviços, em que a interface do usuário se comunica com o intuito de enviar dados para os algoritmos de aprendizado de máquina; e o *feedback* dos algoritmos de aprendizado pode ser obtido e salvo por serviços RESTful.

Por outro lado, Akiki *et al.* [95] criaram monitores na camada de apresentação, que detectam novas situações e mudanças de comportamento e reportam a uma camada de decisão. Algumas situações que podem ser monitoradas incluem: taxa de uso de campos de entrada de um usuário final; novas atualizações instaladas na plataforma; informações coletadas acerca do ambiente, com o uso de sensores; entre outras. Por exemplo, o usuário pode receber, inicialmente, acesso à interface inicial; no entanto, um monitor de comportamento pode detectar que, mesmo nessa parte da interface do usuário, ainda existem alguns campos não utilizados. Assim, o monitor pode acionar uma atualização dos dados de comportamento para indicar que esses campos também devem ser removidos. A camada de decisão verifica se a taxa de uso de determinados elementos da interface do usuário (por exemplo, campos de entrada) é baixa o suficiente para excluir esses elementos da interface do usuário. O Windows Workflow Foundation (WF), que faz parte do .NET Framework, foi utilizado para implementação do comportamento adaptativo.

Diferentemente, Çağla Çığ Karaman e Sezgin [98] utilizaram um rastreador ocular autônomo Tobii X120 [5] e um *tablet* para coletar dados sincronizados do olhar e da caneta, respectivamente. O Tobii X120 opera com taxa de dados de 120 Hz, precisão de rastreamento de 0,5° e desvio inferior a 0,3°. O rastreador permite o movimento livre da cabeça dentro de uma caixa virtual com dimensões 30 × 22 × 30 cm.

3.3.4 QS.4 - Quais modelos, tarefas, técnicas e/ou algoritmos são utilizados para analisar os dados extraídos das interações dos usuários?

Os estudos selecionados fizeram propostas de algoritmos que utilizam dados do usuário para gerar aplicações adaptativas. O intuito da RSL é compreender quais algoritmos são utilizados para obter resultados satisfatórios na arquitetura proposta por este trabalho.

Soh *et al.* [96] fizeram uma proposta de arquitetura para IUA que incorpora um modelo de recomendação sequencial profundo (Modelo de Recomendação Sequencial Profundo

[5] https://www.tobiipro.com/product-listing/tobii-pro-x3-120

(DRNN)). A abordagem utiliza DRNN para aprender um espaço latente, que permite o compartilhamento de dados, de maneira colaborativa. Ao consultar esse espaço latente compartilhado, a IUA é capaz de explorar padrões de uso de toda a base de usuários para fazer adaptações e recomendações personalizadas para usuários relativamente novos. Além de utilizar as interações do usuário em tempo real, a DRNN também analisa o histórico de interação de longo prazo, o que permite a aplicação de métodos de busca aproximados rápidos *k-neighbor-neighbor (k-NN)* que escalam com grandes conjuntos de elementos de interação comuns em modernas.

Já Todi *et al.* [42] contribuíram com métodos para interfaces adaptativas projetadas para operar de forma autônoma, sem feedback explícito ou amostras de treinamento do usuário. O problema computacional central analisado foi como escolher uma adaptação; não abordaram questões como elicitação prévia, explicabilidade, nem o espaço de design de técnicas de interação inteligente. O plano foi selecionar uma sequência de adaptações com o objetivo de maximizar a utilidade para o usuário. Algoritmos de planejamento, como minimax e A-star, utilizam uma representação em árvore do espaço de busca formada por nós conectados por ramos, representando estados válidos e transições entre eles. No entanto, os algoritmos clássicos de busca em árvore, geralmente, exigem a expansão de toda a árvore; porém, isso é computacionalmente caro, devido ao grande número de possíveis adaptações (amplitude) e a profundidade em interfaces adaptativas. O algoritmo MCTS tem sido empregado com sucesso em várias aplicações de jogo para planejar uma sequência de movimentos, de forma eficiente [108]. Um *insight* importante apresentado neste trabalho foi incorporar redes neurais para ajudar a prever quais ramificações têm o maior valor esperado mais rapidamente.

Rathnayake *et al.* [100] propuseram um *framework* para gerar interfaces do usuário adaptativas baseadas no comportamento do usuário. O *framework* utilizou técnicas de agrupamento como *k-means clustering* [109] e DBSCAN [110] para agrupar usuários em diferentes perfis, com base no subconjunto de componentes de IU, que eles usam com frequência; técnicas de classificação como regressão logística [111] e Adaboost [112] para identificar os novos perfis de usuários, com base em padrões comportamentais no site. O aprendizado de máquina foi utilizado para classificar usuários em perfis, de acordo com seu comportamento na interface. Após o agrupamento, os modelos de classificação foram treinados por meio de partes de dados agrupados de tamanhos diferentes e testados em relação às partes restantes dos dados. Os algoritmos GaussianNB (NB), DecisionTreeClassifier (DT), LogisticRegression (LR), Random Forest classificador (RF) e AdaBoostClassifier (AB) na biblioteca Python SciKit Learn [113] foram usados para treinar diferentes modelos de classificação e foram testados quanto à precisão durante essa fase. A precisão do classificador AdaBoost (AB) permaneceu estável, em nível de precisão de 100% para

todos os conjuntos de dados de treinamento utilizados.

O trabalho apresentado por Rim *et al.* [94] utiliza um modelo de Rede Bayesiana para avaliar as preferências do usuário. Esse modelo preserva o conhecimento do sistema acerca das interações e das preferências do usuário final. Depois que um conjunto de interações do usuário foi suficientemente obtido, por meio da aplicação web, uma estrutura do modelo de usuário Bayesiano foi definida, por meio do algoritmo de Recozimento Simulado [114]. Após essa etapa, o algoritmo Inferência por Árvores de Junção [114] foi responsável por calcular as distribuições de probabilidade *a posteriori* para todas as variáveis. Com base nas informações contextuais, o modelo pode deduzir as preferências de todos os usuários.

Gullà *et al.* [97] também utilizaram Rede Bayesiana para avaliar as preferências do usuário e gerar a aplicação adaptativa. O motor adaptativo representa o pivô central do sistema e é composto por um mecanismo adaptativo e um sistema de monitoramento de mudanças. A adaptabilidade consiste em mecanismos de mudança, que incluem todas as características dinâmicas, como preferências baseadas no histórico de interação do usuário, conteúdo de informações, ícones, layout etc. Em particular, o mecanismo adaptativo é baseado na *Bayesian Belief Network* (BBN). Atualmente, as redes Bayesianas são uma das ferramentas mais abrangentes e consistentes para a aquisição, representação e exploração do conhecimento, em condições de incerteza. A fim de alcançar uma ação adequada de adaptação, um Algoritmo de Tomada de Decisão (DMA) [115] foi desenvolvido. Após mapear a rede com as informações da IU e explorar o poder da Rede Bayesiana, que permite obter inferências acerca do estado mais provável de seus nós, o algoritmo define um limite para acionar o evento de adaptação do nó correspondente. A rotina de adaptação permite gerenciar duas informações diferentes: uma atualização em tempo real das informações correlacionadas à interação do usuário; e um armazenamento do fluxo complexo de interação do usuário para rastrear seu histórico de interação.

Uma abordagem distinta das demais foi apresentada por Todi *et al.* [101], em cujo trabalho foram utilizadas quatro propriedades principais:

- **Frequência:** o sistema registra o número de visitas a cada interface. Ao selecionar um modelo para o usuário, ele simplesmente precisa iterar nas interfaces visitadas para encontrar a interface com a maior frequência. Portanto, N iterações são necessárias para selecionar um projeto de modelo, e o tempo de computação é desprezível;
- **Curva de Posição Serial:** após uma extensa pesquisa empírica acerca de memória de longo e curto prazo, viu-se que o primeiro e o mais recente objeto acessado é importante para o usuário. Desse modo, foi implementada uma função matemática, que descreve a relação entre ordem e probabilidade de revisitação. Isso é usado para escolher o design que será mais útil para o usuário;

- **Aprendizagem Estatística Visual:** esse princípio utiliza mapas de distribuição de probabilidade para selecionar posições para cada recurso. Para gerá-los, inicia-se com um *array* 2×2 vazio, com as dimensões iguais ao tamanho da tela de um layout de interface. Para cada recurso, ele itera nas N interfaces e verifica se o recurso foi encontrado. Se presente, ele incrementa o valor das posições correspondentes, pixel a pixel, na matriz de recursos. Uma vez que todos os mapas de distribuição de probabilidade são gerados, ele itera sobre cada um deles para encontrar a posição mais provável para o recurso e, assim, cria o modelo computacionalmente mais caro;
- **Modelo Cognitivo Gerador:** o sistema registra o histórico sequencial de M visitas às N interfaces, juntamente com os tempos de acesso correspondentes. Dada a posição dos recursos, o modelo gera a duração do olhar e atualiza isso para cada uma das M visitas. Portanto, itera M vezes para gerar todos os pontos de ativação. O modelo é criado, ao selecionar o ponto de ativação mais proeminente para cada recurso encontrado nas interfaces. O tempo de computação é, portanto, influenciado pelo número total de visitas à interface (M) e pela duração das visitas.

Akiki *et al.* [95] apresentaram *Role-Based UI Simplification* (RBUIS), um mecanismo para melhorar a usabilidade, por intermédio do comportamento adaptativo, que fornece aos usuários um conjunto mínimo de recursos e um layout ideal, baseado no contexto de utilização do usuário. Com RBUIS, a adaptação começa com o método minimização do conjunto de recursos, que elimina o subconjunto de interface do usuário que não é exigido pelos usuários. Posteriormente, a otimização do layout torna a interface do usuário mais adequada para os aspectos do contexto de uso (por exemplo: alfabetização em informática, cultura, tamanho da tela), e adapta fatores de interface do usuário (como: tamanho da fonte, agrupamento e tipos de *widgets*).

Já Kong *et al.* [93] desenvolveram um algoritmo heurístico, por meio de programação linear para gerar adaptações em interfaces multimodais, baseado no clássico problema da mochila 0-1.

Fernandez Garcia *et al.* [99] não trataram diretamente do algoritmo, porém desenvolveram uma arquitetura para que os dados dos usuários sejam avaliados por algoritmos de Inteligência Artificial (IA), estatísticos ou de aprendizado de máquina.

Por fim, Çağla Çığ Karaman e Sezgin [98] criaram uma máquina de aprendizagem que consiste em três etapas para predizer interfaces do usuário. A primeira etapa envolve a extração de vetores de recursos de um conjunto de amostras de dados. A segunda etapa se refere ao treinamento de modelos de previsão, por meio de vetores de recursos extraídos. Para isso, um único modelo de *Support Vector Machine* (SVM) foi treinado, com o uso do *kernel* de uma função radial gaussiana (RBF), que teve como entrada os dados inteiros,

ao contrário de particionados. A terceira etapa consiste em utilizar os dados do usuário em tempo real durante o estudo de usabilidade.

3.3.5 QS.5 - Houve melhoria na experiência do usuário, após a utilização do algoritmo e da página adaptativa?

A implementação realizada no estudo dos pesquisadores Todi *et al.* [42] foi submetida a uma validação empírica com 18 participantes, e o resultado foi positivo. Os participantes perceberam que a categorização dos itens em grupos melhorou com a adaptação e os auxiliou na busca por itens relacionados. Um dos participantes comentou: "[...] os itens eram organizados em categorias frequentemente - o que ajudava no momento da seleção".

O *framework* apresentado por Rathnayake *et al.* [100] foi testado com avaliações heurísticas. A precisão das técnicas de aprendizado de máquina medidas mostra que o mecanismo para analisar os dados comportamentais do usuário e rotular os usuários é eficaz e preciso. O *clustering* eliminou a necessidade de anotação manual de dados durante o treinamento, o que torna todo o processo automatizado. O sistema apresenta usabilidade aprimorada, em relação aos sistemas existentes, devido à sua configuração simples e à capacidade de ser usado por usuários não técnicos.

Akiki *et al.* [95] apresentaram o RBUIS, mecanismo para melhorar a usabilidade, por meio do comportamento do usuário, que fornece aos usuários finais um conjunto mínimo de recursos e um layout ideal com base no contexto. Eles avaliaram a eficiência do projeto desenvolvido com 23 participantes e, em relação à perspectiva humana, foi visto que interfaces com um conjunto de recursos minimizado e um layout otimizado provocaram uma melhoria estatisticamente significativa, em relação à percepção do usuário final. O rastreamento ocular também foi realizado e mostrou que minimizar o conjunto de recursos de interfaces complexas diminui, significativamente, o tempo que usuários finais perdem ao pesquisar campos de entrada.

Em uma avaliação qualitativa, Çağla Çığ Karaman e Sezgin [98] perceberam que os usuários consideraram as interfaces preditivas em tempo real mais fáceis de usar, mais rápidas de aprender e se sentiram mais confiantes ao usá-las. Além disso, descreveram como mais simples, mais consistentes e também precisaram de menos informações prévias antes de usá-las. Por fim, os usuários julgaram que foram mais bem-sucedidos na conclusão das tarefas, e se sentiram menos frustrados com as interfaces preditivas.

3.3.6 QP - Quais são os algoritmos frequentemente utilizados para o desenvolvimento de aplicações web adaptativas?

Como descrito na seção 3.3.4, os estudos selecionados utilizaram técnicas e/ou algoritmos para gerar aplicações web adaptativas. Como resumo, Soh *et al.* [96] desenvolveram um DRNN. Todi *et al.* [42] utilizaram o algoritmo MCTS aprimorado com redes neurais. O *framework* apresentado por Rathnayake *et al.* [100] utilizou técnicas de agrupamento como *k-means clustering* e DBSCAN [110] e técnicas de classificação como regressão logística [111] e Adaboost [112]. Rim *et al.* [94] e Gullà *et al.* [97] utilizaram um modelo de Rede Bayesiana para avaliar as preferências do usuário. Já Todi *et al.* [101] utilizaram um modelo baseado em quatro propriedades principais: Frequência, Curva de Posição Serial, Aprendizagem Estatística Visual e Modelo Cognitivo Gerador. Akiki *et al.* [95] utilizaram o método de minimização do conjunto de recursos, Kong *et al.* [93] desenvolveram um algoritmo heurístico por meio de programação linear baseado no clássico problema da mochila 0-1 e, por fim, Çağla Çığ Karaman e Sezgin [98] utilizaram um único modelo de Support Vector Machine (SVM) treinado por meio do *kernel* de uma função radial gaussiana.

Em resposta ao Objetivo Específico 2 (Selecionar um, dentre os algoritmos resultantes da RSL, para ser utilizado na arquitetura proposta), foi selecionada a abordagem apresentada no estudo *Adapting User Interfaces with Model-based Reinforcement Learning* [42]. Essa abordagem utiliza o modelo AR, juntamente com o algoritmo MCTS, para aumentar a eficiência da amostra. Nesse contexto, o MCTS é importante; pois, quando há um grande número de estados alcançáveis e ações possíveis, a árvore é podada e apenas o nó mais promissor, que resulta em uma recompensa esperada maior, é selecionado até algum nível. Outro *insight* importante foi incorporar redes neurais para ajudar a prever quais ramificações têm o maior valor esperado e, assim, lidar com instâncias de problemas maiores. Com isso, também foi incorporado ao método uma rede neural para calcular mais rapidamente a grande quantidade de adaptações possíveis.

Esta decisão é apresentada na seção 4.2. No entanto, publicou-se um artigo, em 2021, como uma visão geral, que avalia estudos mais recentes. Além disso, a abordagem apresentada foi submetida a uma validação empírica, com 18 participantes, que resultou em uma melhor experiência para os usuários. Essa análise é importante, pois responde ao Objetivo Específico 3. Por último, foram utilizadas tecnologias modernas para o desenvolvimento da solução, como HTML e Javascript para o *frontend*, Python e TensorFlow para o *backend*.

Diferentes abordagens computacionais para esse problema foram apresentadas nos artigos selecionados, como sistemas baseados em regras, heurísticas, otimização Bayesiana e aprendizado supervisionado. Embora tenham sido obtidos resultados empíricos positivos,

as abordagens conhecidas têm sido criticadas por serem imprevisíveis e não confiáveis, pois tomam decisões prejudiciais, com frequência. Por outro lado, o AR permite políticas de aprendizado para sequências de ações em que as recompensas não necessitam ser imediatamente alcançadas. Esse algoritmo utiliza tentativa e erro para encontrar uma solução, que recebe recompensas ou penalidades pelas ações que executa. Seu objetivo é maximizar a recompensa total, por meio de um modelo preditivo para simular possibilidades para encontrar uma solução e melhorar, significativamente, a eficiência em encontrar boas soluções [116]. Essa abordagem ajuda a responder a uma questão técnica que se encontra na interseção da interação humano-computador e da pesquisa de aprendizado de máquina, que é: como selecionar a melhor adaptação e quando aplicar a alteração?

Capítulo 4

Arquitetura Proposta

Neste capítulo, será apresentada a arquitetura para aplicação web bancária, que tem como base conceitos de sistemas reativos e cenários baseados nas demandas do Banco Beta. Foi definida a utilização de conceitos de sistemas reativos, pois são sistemas mais flexíveis, desacoplados e escaláveis, que os tornam mais fáceis de desenvolver, mais preparados para mudanças, mais tolerantes a falhas e com alto grau de responsividade, de modo a permitir aos usuários um efetivo *feedback* interativo [23].

Em vista disso, essa arquitetura será documentada, de tal forma que o estilo e os padrões arquiteturais serão descritos, a iniciar pelos conceitos e ideias principais, em uma visão funcional, até as tecnologias de implementação e implantação. Dessa forma, este capítulo contempla tanto as abstrações de alto nível, isto é, as decisões necessárias para que o sistema atenda às metas de desenvolvimento, comportamento e qualidade, quanto os detalhes de nível mais baixo, que descrevem a estrutura interna de cada componente.

Primeiramente, é realizada uma descrição dos fatos que motivaram esta proposta de arquitetura (seção 4.1). Em seguida, é apresentado o algoritmo que realiza as adaptações no *frontend* 4.2, seguido por uma visão geral da arquitetura 4.3, que contempla um diagrama e sua descrição, a fim de identificar o funcionamento de uma aplicação desenvolvida com base na arquitetura proposta. A seção seguinte descreve o projeto da arquitetura 4.4, composta pelo **escopo**, que apresenta os aspectos estruturais da referida arquitetura; os **interesses**, que mostram uma visão de alto nível dos componentes; os **princípios**, que descrevem como a arquitetura segue os princípios de sistemas reativos; as **restrições**, que delimitam a documentação, implantação e implementação; os **stakeholders**, que elencam as partes interessadas na arquitetura; e os **cenários de uso**, que apresentam as funcionalidades que o sistema deve apresentar. Em conjunto, isso demonstra como o sistema executará as funções exigidas dele [117]. Por último, a seção 4.3.1 descreve o estilo arquitetural, em termos do padrão de organização estrutural, a Arquitetura Orientada a Serviços (SOA).

4.1 Motivação para a proposição da arquitetura

Apesar de que o cenário descrito no Capítulo 1 possibilite novas oportunidades de negócio, também traz desafios importantes para o setor bancário. No Brasil, a concorrência entre os bancos tradicionais e os novos bancos, chamados digitais, está cada vez mais acirrada. Isso ocorre porque os serviços bancários são semelhantes; assim, a UX se torna um fator fundamental para diferenciar a qualidade entre os bancos; e a escolha de muitos clientes ocorre com base nesse quesito [10].

Como descrito na Seção 2.7, a usabilidade é um elemento que influencia, significativamente, a UX [12][13]. Dessa maneira, os critérios de usabilidade podem ser utilizados para atingir uma experiência positiva [14]. Uma das maneiras de elevar o nível de usabilidade e oferecer uma melhor UX para os usuários é construir uma arquitetura de software robusta, em que também observa os critérios de usabilidade [22].

Portanto, a motivação desta proposta de arquitetura é possibilitar o desenvolvimento de aplicações web bancárias, que observem também critérios de UX. Por esse motivo, a arquitetura proposta por este trabalho inclui uma IU adaptativa para o usuário interagir com o sistema. Conforme explicado na Seção 2.9, a IU adaptativa fornece uma melhor experiência na utilização da aplicação, pois serviços personalizados têm maior impacto na experiência do usuário e podem afetar o seu nível de satisfação [118]. A utilização de critérios de sistemas reativos, proposta por este trabalho, também contribui para uma melhor experiência de utilização, pois um dos objetivos desses critérios é fazer com que o sistema seja responsivo, ou seja, que responda às requisições dos usuários em um tempo razoável. Conforme Bonér *et al.* [23], responsividade é a pedra fundamental da usabilidade, pois permite que os problemas possam ser detectados rapidamente e tratados com a máxima eficácia.

4.2 Algoritmo de Adaptação

Sistemas adaptativos podem fornecer maiores benefícios, ao planejar sequências de adaptações e realizar mudanças graduais. No entanto, considerar somente um curto tempo pode gerar resultados não ideais e adaptações que não condizem com o comportamento do usuário. Isso pode acontecer quando, por exemplo, o usuário realizou ações atípicas durante um curto espaço de tempo. Por outro lado, considerar uma longa sequência de adaptações aumenta o tamanho do espaço de busca, que pode impossibilitar o processamento. O ARBM é uma abordagem eficaz para resolver essa categoria de problemática, como descrito na Seção 2.1. Como o sistema adaptativo deve considerar uma sequência de adaptações, temporalmente, então, pode-se definir como um problema de decisão se-

quencial estocástico [119]. Portanto, encontrar a melhor adaptação - ao avaliar o valor de cada sequência de adaptações - é computacionalmente custoso, especialmente, quando considera sequências de mudanças por um longo tempo. Para resolver esse problema computacional, é utilizada uma combinação de MCTS e AP para aumentar o desempenho, diminuir a quantidade de tentativas e erros durante a navegação do usuário. A Seção 2.2 detalha o algoritmo Busca por Árvore de Monte Carlo (MCTS), e a Seção 2.10 detalha os conceitos de AP.

A fim de cumprir o objetivo específico 4, foi selecionada a técnica apresentada por Todi *et al.* [42]. Este trabalho avaliou estudos mais recentes para chegar ao resultado; além disso, foi submetido a uma validação empírica, que resultou em uma melhor experiência para os usuários. Por último, essa técnica utilizou tecnologias modernas para o desenvolvimento da solução, como HTML e Javascript para o *frontend*; e Python e TensorFlow para o *backend*. A abordagem selecionada também alcançou um ótimo resultado, pois demonstrou melhora na experiência dos usuários durante a utilização.

4.2.1 Formação do problema

Para abordar o problema de adaptação, formulou-se um modelo de tomada de Decisão Sequencial Estocástico (DSE) [119], em que o sistema adaptativo decide quais elementos devem ser adaptados, com base em suas observações. O objetivo é escolher uma sequência de adaptações que maximize o valor esperado para o usuário, em uma janela mais longa de interações. Na arquitetura apresentada, uma página web foi otimizada para melhorar o desempenho da seleção de serviços bancários, divididos em várias sessões de interação. Porém, embora o sistema possa mudar uma IU, ele não pode alterar o comportamento do usuário com seus próprios processos, por exemplo: os usuários podem aprender e mudar seus interesses, em questão de tempo. Isso complica o problema, pois as adaptações escolhidas podem levar a interações irreversivelmente ruins. Portanto, é essencial escolher as adaptações e considerar esses fatores de mudança de comportamento do usuário.

O problema pode ser apresentado como um Processo de Decisão de Markov (PDM). Dessa maneira, é possível obter uma abordagem precisa e eficaz, além de obter uma compreensão mais profunda do processo de tomada de decisão, conectá-lo às abordagens teóricas e práticas existentes na pesquisa de IA e Aprendizado de Máquina (ML) e identificar soluções algorítmicas adequadas. Portanto, a formulação do PDM oferece uma metodologia rigorosa para lidar com o esse tipo em questão.

O problema das IU adaptativas é maximizar as recompensas descontadas cumulativas esperadas $r(s_t, a_t)$ de agir, de acordo com uma política ótima π^*, como mostrado na Figura 4.1.

$$\pi^* = \underset{\substack{\pi \\ p(s_{t+1}|s_t,a_t)}}{\operatorname{argmax}} \; \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right]$$

Figura 4.1: Formulação de PDM para maximizar as recompensas cumulativas esperadas
Fonte: Adaptada de Todi *et al.* [42]

Em que:

- $s \in S$ é um estado de interação que consiste na IU (d) e no usuário (u);
- $a \in A$ é uma adaptação, ou seja, uma mudança que pode ser realizada;
- p é uma função de transição que fornece a probabilidade de transição do estado s para o estado s_{t+1}, após realizar a adaptação a, isto é, $p(s_{t+1}|s_t, a_t)$;
- r é uma recompensa coletada por fazer a adaptação a no estado s;
- γ é um fator de desconto, que controla o quanto favorece a recompensa imediata (pequeno γ), em relação a longo prazo (grande γ).

Considerando a adaptação da IU inicial de uma aplicação web, com uma grade de ícones (componentes), que representa os serviços bancários, o estado do sistema, denotado como s, é uma função tanto da interface inicial d quanto do estado futuro do usuário u, que interage com a aplicação. A interface d inclui fatores como a disposição dos ícones. Quanto ao usuário u, fatores como experiência, interesses e habilidades são considerados.

Quando o usuário interage com a aplicação, uma adaptação a pode ser feita para criar uma IU. Isso pode envolver a alteração da interface ou da ordem dos componentes, por exemplo. A função de transição p descreve como o estado interno do usuário muda, em resposta ao estado da interface externa. A recompensa r reflete os benefícios da adaptação, como a redução do tempo necessário para selecionar um componente. O fator de desconto $/gamma$ indica a importância relativa dos benefícios imediatos e das melhorias de longo prazo no cálculo da recompensa. O objetivo do sistema adaptativo é encontrar uma política adequada t para selecionar adaptações que maximizem a recompensa cumulativa.

No entanto, existem vários desafios no desenvolvimento de uma política para IU adaptativas. O verdadeiro estado da página é totalmente observável, mas o verdadeiro estado do usuário não é diretamente acessível. Portanto, modelos preditivos de IHC são usados para estimar o estado do usuário. Além disso, calcular a recompensa não é simples, pois não há *feedback* direto do usuário. Em vez disso, modelos preditivos de IHC podem ser usados para criar funções objetivas relacionadas a custos de desempenho e reaprendizagem.

Finalmente, há um grande número de adaptações possíveis, que podem ser feitas na página em qualquer estado. Ao considerar uma sequência de adaptações, em um longo período, o espaço de estados cresce exponencialmente. Portanto, para enfrentar esses desafios, este trabalho utilizou a técnica de Aprendizado por Reforço Baseado em Modelo (ARBM).

4.2.2 O Método Aprendizado por Reforço Baseado em Modelo (ARBM)

A essência da abordagem utilizada neste trabalho gira em torno do planejamento, que envolve a seleção de uma sequência de etapas de adaptação, destinada a maximizar a utilidade do usuário. Algoritmos de planejamento, como minimax e A-star, empregam uma representação em árvore do espaço de busca. No entanto, algoritmos tradicionais de busca em árvore, geralmente, exigem a expansão de toda a árvore, o que pode ser computacionalmente caro, especialmente, devido à amplitude e profundidade das possíveis adaptações em IU adaptativas. O MCTS pode lidar com a incerteza, ao analisar as adaptações mais promissoras e expandir os nós da árvore, por meio de técnicas de amostragem aleatória. Porém, neste trabalho, foi adicionada uma integração com redes neurais de valor para auxiliar na predição dos ramos com maior valor esperado, a fim de facilitar o tratamento de instâncias de problemas maiores.

Planejamento com oMCTS

Como o usuário pode continuar a interagir com a IU, sem limites pré-determinados, não há um horizonte definido. Portanto, foi utilizada a implementação padrão do MCTS, conforme detalhado na Seção 2.2. Diante disso, a recompensa é estimada de maneira cumulativa, ao longo de um período razoável.

No fim da sessão de interações do usuário, o estado-raiz s_0 é determinado pela interface atual d_0 e pelas observações do usuário u_0. A **seleção** de um estado $s_j \in S$ é realizada por meio do método Limites Superiores de Confiança Aplicados a Árvores (UCT), conforme citado por Todi *et al.* [42], que é um estimador comumente usado no planejamento baseado em modelos [120]. Um aspecto crucial do UCT é seu coeficiente C, que equilibra entre as explorações para avaliar múltiplas possíveis adaptações, como mostrado na Figura 4.2.

$$\mathrm{UCT} = \frac{r_j}{n_j} + C\sqrt{\frac{\ln n_i}{n_j}}$$

Figura 4.2: Fórmula do método UCT
Fonte: Adaptada de Todi *et al.* [42]

Nessa equação, r_j representa a recompensa total para o estado-filho s_j, n_j, denota o número de vezes que s_j foi visitado e n_i se refere ao número de vezes em que o estado-pai s_i foi visitado. Já a constante de exploração C é definida como $1/\sqrt{2}$, conforme a convenção. Se todas as adaptações possíveis do estado selecionado s_j tiverem sido exploradas anteriormente, então, o processo de seleção é repetido até que um estado de folha com adaptações inexploradas seja escolhido.

No passo de **expansão**, o nó escolhido s_j é expandido, ao selecionar uma adaptação $a \in A$, que leva a um novo estado $n_{j+1}, r_{j+1} = 0$. Neste trabalho, também foi considerada a possibilidade de que o estado expandido seja visível ou invisível para o usuário. Aproveitar a invisibilidade permite o planejamento de várias adaptações em um único turno.

Nas **simulações**, como a árvore não possui estimativas de valor para guiar a seleção de estados consequentes, uma sequência de adaptações $a_0, ..., a_N$ é selecionada, aleatoriamente, e as recompensas são estimadas por meio de modelos preditivos de IHC. Cada modelo simula o resultado da sequência de adaptação e fornece uma estimativa de valor. Esse processo é repetido por um número fixo de etapas, determinado pelo horizonte H; e as recompensas cumulativas são calculadas para cada modelo preditivo, como mostra a Figura 4.3.

Figura 4.3: Seleção das adaptações e estimativas das recompensas

$$r_{j+1} = \sum_{k=j+2}^{H} r_k$$

Fonte: Adaptada de Todi *et al.* [42]

Por fim, na etapa de **retropropagação**, isto é, no fim das simulações, a recompensa cumulativa r_{j+1} é retropropagada do estado recém-expandido s_{j+1} para o estado inicial s_0; e os valores (recompensas r e visitas s) são atualizados. As etapas são repetidas várias vezes, para obter estimativas de valor para cada estado adaptado.

Uma **nova adaptação será selecionada**, ao utilizar a Equação da Figura 4.3, que define $C = 0$, para maximizar a utilidade esperada para o usuário. Foram utilizados dois métodos para escolher uma estratégia de adaptação, que permita ao usuário equilibrar risco e ganho de forma eficaz. A abordagem utilizada neste trabalho é baseada na suposição de que existem vários modelos preditivos em IHC, que podem definir limites para o comportamento real, como fornecer estimativas de melhor caso e de pior caso. Além disso, é possível utilizar modelos preditivos que atendem a vários objetivos, como reduzir o tempo de conclusão da tarefa, minimizar a carga cognitiva e evitar interrupções [121]. Ao combinar as estimativas de valor de todos esses modelos, foi implementada a estratégia ótima, a qual seleciona o modelo que maximiza as recompensas.

Estimativa com Redes Neurais Profundas

Para enfrentar o desafio de lidar com a abundante quantidade de dados produzida em um ambiente on-line, em que a realização de um número adequado de simulações MCTS para obter estimativas confiáveis não é viável, utilizou-se uma arquitetura de RNP capaz de fornecer previsões em tempo real, de forma eficiente, ao utilizar uma rede neural de valor pré-treinada para obter estimativas de valor para estados ainda não explorados.

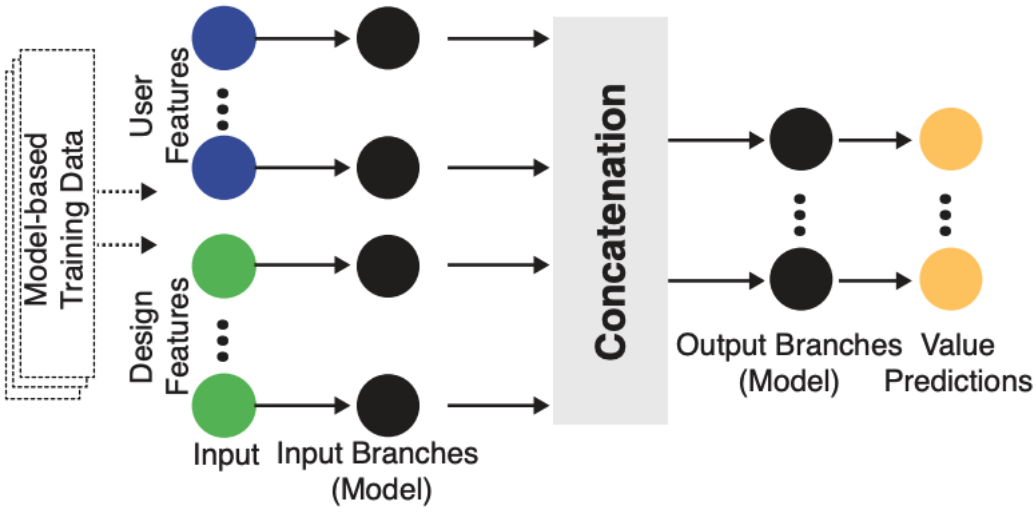

Figura 4.4: Arquitetura da Rede neural para obter o valor estimado
Fonte: Adaptada de Todi *et al.* [42]

A arquitetura proposta, conforme representada na Figura 4.4, consiste em uma interface com $m-cabeça$ e $n-cauda$, treinada de ponta a ponta, por meio de retropropagação. Cada um dos parâmetros de entrada m é tratado como uma ramificação ou cabeça de

modelo independente, concatenada e, subsequentemente, passada para ramificações ou caudas de modelo independente n. Durante o treinamento, os dados baseados em modelo são gerados, ao executar a implementação do MCTS, a partir de estados aleatórios. Estimativas de valor com informações de estado, acerca da interface e usuário são usadas como amostras de entrada, de modo a treinar um modelo de rede neural. O modelo é parametrizado com a interface e os dados do usuário para prever estimativas de valor para os estados no ambiente on-line. As estimativas de valor são desacopladas das funções objetivas, de modo que o sistema adaptativo determina como usar as informações de vários modelos. Os principais benefícios do uso de redes neurais para estimativa são sua alta capacidade de aprendizado e capacidade de avaliar milhares de estados, em tempo real, sem incorrer em simulações computacionalmente custosas.

4.2.3 Aplicação em IU Adaptativas

A abordagem utilizada por este trabalho pode ser aplicada a várias dimensões, como organizar ícones em telas iniciais de dispositivos móveis, adaptar o layout de páginas da web, reorganizar menus de aplicativos ou adaptar menus. O objetivo pode diferir, com base na aplicação, e varia desde a redução da carga cognitiva até a diminuição do tempo para selecionar um componente. Também, é possível utilizá-la para com diferentes categorias de adaptações, que incluem mudanças na apresentação e comportamento de elementos gráficos.

Para isso, há modelos IHC [122], que capturam, com precisão, o impacto dessas adaptações nos objetivos desejados. Na próxima seção, a eficácia dessa abordagem será demonstrada em uma página que contém um conjunto de componentes, que representam serviços bancários, além disso, será apresentado um novo modelo IHC para prever o tempo que o usuário leva para selecionar esses componentes dispostos na IU.

4.2.4 Aplicação Bancária

A aplicação prática desta abordagem foi executada em uma IU de um sistema bancário. A disposição dos componentes tem o potencial de melhorar a usabilidade, mas alterações inesperadas podem levar a uma queda temporária de desempenho, aumentar a carga cognitiva e até fazer com que o usuário rejeite a adaptação. Esta abordagem fornece uma solução geral, que pode acomodar grades de componentes com até nove itens, divididos em três linhas de três colunas. A execução considera apenas adaptações que modifiquem a posição dos componentes; outras adaptações de apresentação são sugeridas para trabalhos futuros.

Definição do problema

De acordo com a formulação geral da Seção 4.2.1, primeiro foi definido o problema da página web bancária. A Figura 4.5 representa a ilustração de um exemplo.

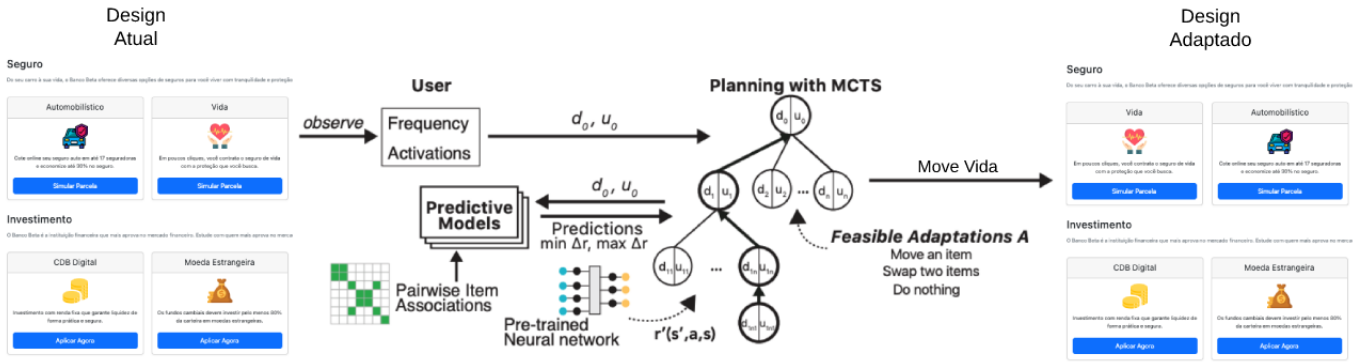

Figura 4.5: Adaptação com Aprendizado por Reforço Baseado em Modelo
Fonte: Elaborada pelo autor

Estado (S): o estado $s \in S$ contém a informação do estado atual da IU e do usuário.

O sistema emprega um método para estimar a proficiência e os interesses dos usuários, ao analisar seus cliques nos componentes dispostos na IU. O objetivo é calcular o nível de habilidade do usuário de selecionar os componentes (denotado como i_l) em uma IU que contém n componentes:

$$B(i_l) = \sum_{j=1}^{n} (T - T_{j,i_l})^{-\rho}$$

Figura 4.6: Definição do problema aplicado à interface
Fonte: Adaptada de Todi *et al.* [42]

Em que:

$B(i_l)$ é o nível de ativação do item i na localização l da memória; T é o tempo atual; T_{j,i_l} é o tempo da seleção j^{th} de i_l; e p é um parâmetro de decaimento igual a 0,5. O interesse do usuário [123] é dado pela distribuição de frequência dos comandos selecionados durante a sessão de interação anterior, que contém N cliques. Modelos estatísticos adicionais de interesse e experiência do usuário [124][125][126][127] podem ser conectados à arquitetura.

Adaptações possíveis: o conjunto de adaptações possíveis para a reorganização dos componentes inclui três opções:

1. realocar um componente para uma posição específica;

2. trocar as posições de dois componentes;

3. deixar a IU inalterada.

Funções de Transição (p): para determinar qual adaptação fazer do estado inicial s_0 para o estado resultante s_1, emprega-se o MCTS e utiliza-se a fórmula UCT para calcular a probabilidade de selecionar uma adaptação específica. Durante a fase de planejamento, houve equilíbrio entre explorar adaptações de alta recompensa e explorar outras possibilidades. Para selecionar a adaptação a a ser feita na IU atual d_0, que resulta em uma nova interface d_1, é usada uma abordagem gulosa, como mostra a Figura 4.7.

A abordagem gulosa é uma estratégia de resolução de problemas de otimização, que faz escolhas locais ótimas em cada etapa, a fim de encontrar uma solução globalmente ótima. A abordagem não considera todas as possibilidades, mas toma decisões imediatas baseadas nas informações disponíveis no momento, com o objetivo de obter o melhor resultado possível [128].

$$\underset{a}{\operatorname{argmax}} \frac{r_j}{n_j}$$

Figura 4.7: Fórmula da abordagem gulosa utilizada na função de transição
Fonte: Adaptada de Todi *et al.* [42]

Recompensa (r): é expandido o escopo de modelos preditivos de IHC para gerar estimativas de recompensa que consideram os custos implícitos de adaptações. Esses modelos utilizam a experiência do usuário e estimativas de interesse para prever os tempos de seleção de componentes para diferentes estratégias do usuário. A recompensa r para cada modelo é, então, calculada como a diferença no tempo médio de seleção, conforme os pesos de interesse do usuário. Quando o estado adaptado é visível para o usuário, é simulada uma sessão de interação, com base no interesse desse usuário para gerar novos cliques e atualizar seu nível de experiência. Por outro lado, o sistema pode usar estados invisíveis para reter a apresentação e combinar várias adaptações, de modo que o estado seja apenas um caminho para outro estado, sem ser exibido. Nesse caso, nenhuma atualização é aplicada.

Modelos para simular a busca em uma grade de componentes

Como relatado anteriormente, a interface bancária definida neste trabalho é formada por uma grade de componentes agrupados por categorias. Primeiramente, as categorias são apresentadas em forma de uma lista linear vertical, em que cada uma dessas categorias

contém uma lista linear horizontal, composta por três componentes (serviços bancários). Os componentes são formados por um título, por um ícone que representa determinado serviço bancário e um botão para o usuário realizar a ação, como retrata a Figura 4.8. Desse modo, a adaptação ocorre tanto na organização das categorias, isto é, as linhas de categorias mais úteis para o usuário deverão ficar no topo e as linhas menos úteis ficarão abaixo. Ao clicar em qualquer componente de determinada categoria, mais útil essa categoria será para o usuário. Por outro lado, os componentes nas categorias também serão adaptados, isto é, os componentes mais úteis para o usuário ficarão à esquerda, já os menos úteis ficarão à direita.

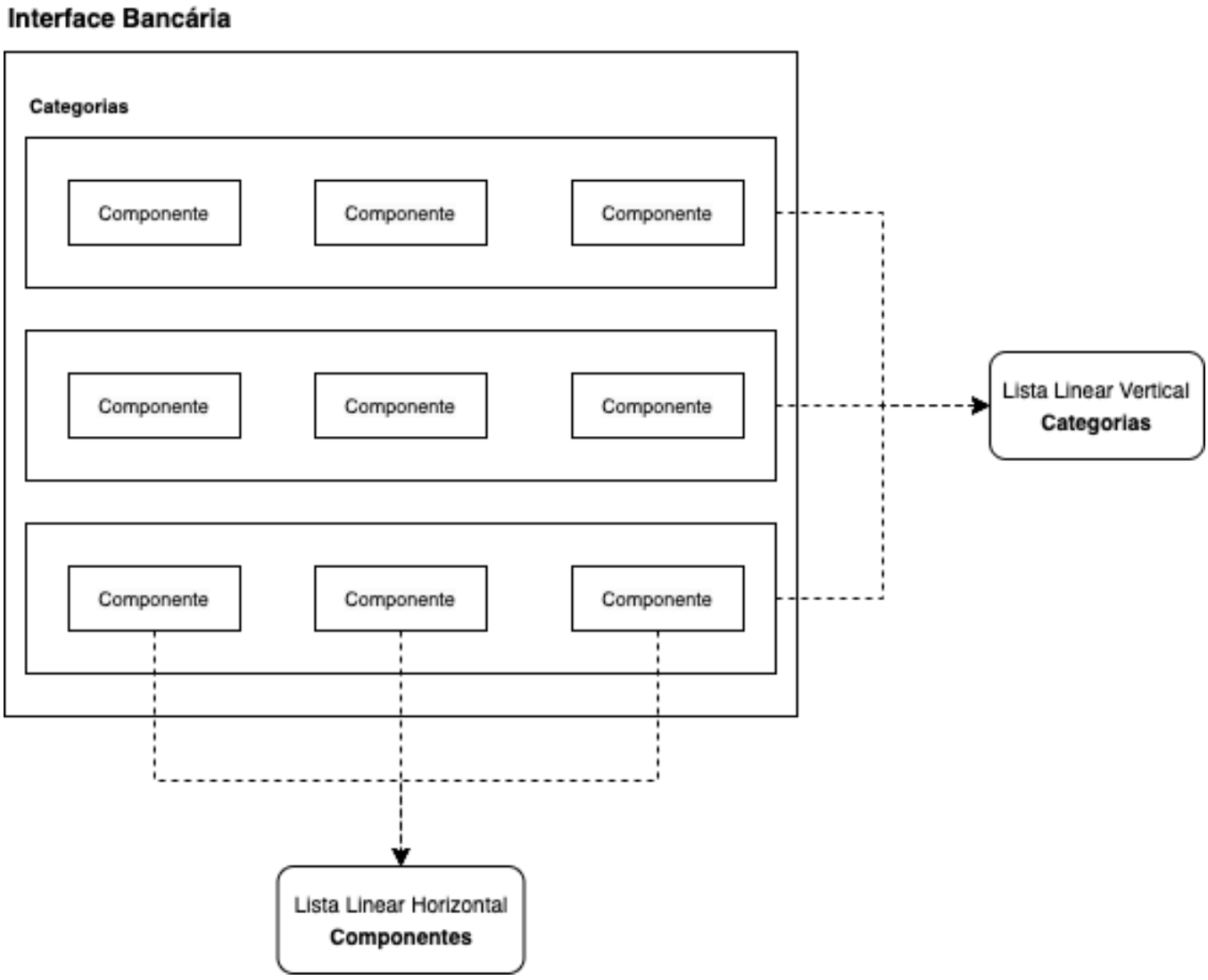

Figura 4.8: Grade de componentes que representa os serviços bancários
Fonte: Elaborada pelo autor

Na pesquisa realizada por Faraday [129], foi detectado que elementos na parte superior

e esquerda da página, com base na ordem de leitura, tendem a ser notados primeiro; por isso, essa localização é considerada mais importante. Reforçando essa premissa, [130] indica que, no layout da página, a parte superior é sempre mais dominante. De acordo com esses artigos, a interface adaptativa proposta manterá as categorias mais úteis para os usuários sempre no topo da página; do mesmo modo, os componentes mais úteis ficarão mais à esquerda. Além disso, Quinn *et al.* [131] define que a proximidade descreve a tendência de elementos individuais serem mais associados com elementos próximos do que aqueles que estão mais distantes. A diretriz é que os componentes com mais sinergia devem ser agrupados, e esse agrupamento seguirá uma ordem de leitura, que se move da esquerda para a direita e de cima para baixo. Isso justifica as categorias propostas agruparem os componentes que representam os serviços bancários.

Cada categoria (ou linha horizontal) é um *micro-frontend* diferente, desenvolvido e mantido por uma equipe distinta na organização. Com isso, essa subaplicação pode ou não ser composta por um conjunto de três componentes, pois a própria equipe deverá selecionar a categoria de conteúdo que mais se adéqua ao serviço bancário, que pode ser, por exemplo: uma imagem clicável, um texto informativo customizado com HTML, um formulário, entre outros. Porém, quando a opção selecionada for um conjunto de três componentes, os componentes da respectiva subaplicação serão organizados horizontalmente, como explicado.

Vários modelos preditivos explicam como os usuários buscam em listas lineares [132] [133] [134] [135]. Este trabalho se baseou nos modelos para definir duas estratégias de buscas para avaliar a utilidade da IU, que simulam tarefas de buscas, como ilustra a Figura 4.9:

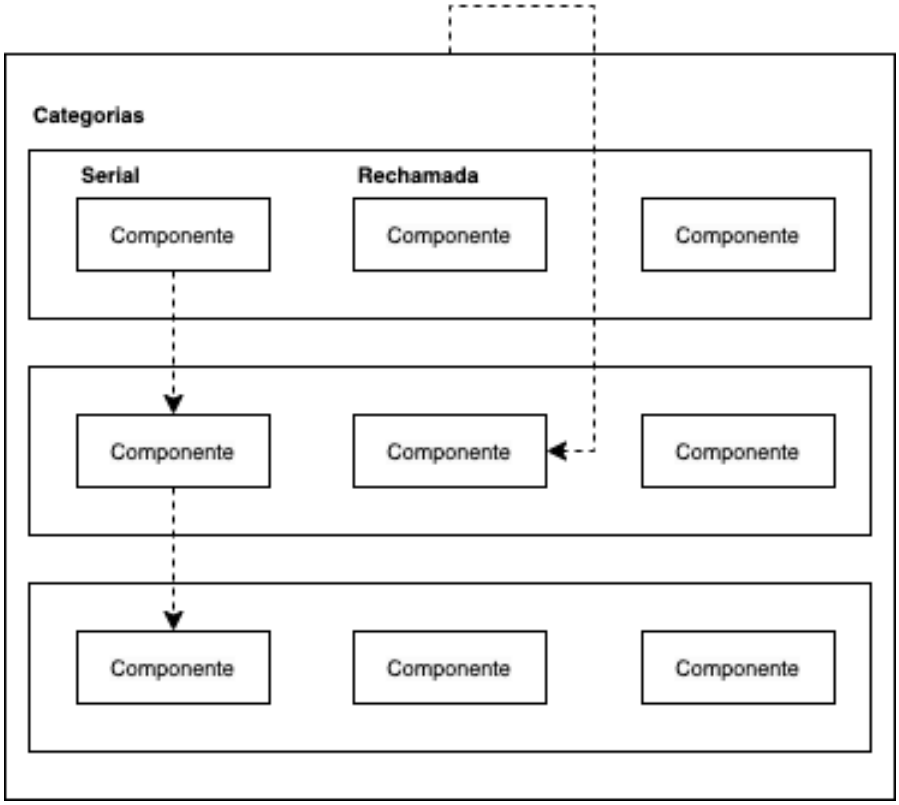

Figura 4.9: Exemplo das estratégias de busca
Fonte: Elaborada pelo autor

1. *Busca serial*: o usuário busca, serialmente, de cima para baixo ou da esquerda para a direita, até encontrar o item desejado na lista;

2. *Busca de rechamada*: o usuário depende da sua própria memória para procurar o item desejado, em um local esperado, na lista.

Esses dois modelos foram selecionados sob a suposição de que eles estabelecem limites para desempenho de melhor e pior caso. Também, considera-se a possibilidade de que o usuário opte por uma estratégia subótima (rechamada), por motivos como desinteresse ou falta de esforço.

Quadro 4.1: Principais notações usadas nesta seção

Notação	**Descrição**
i_l	Item-alvo i na localização l
$T_{modelo}(i)$	Tempo de busca para um item i de determinado modelo
δ	Constante para ajuste no custo de um item específico
T_c	Constante para penalizar quando um item não foi encontrado como esperado
T_cauda	Tempo de apontamento constante quando o cursor acompanha o olhar
$M(i_k, i_l)$	Relação booleana entre i_k e i_l de uma matriz de associação M
$B(i_l)$	Nível de ativação para i em l

Fonte: Elaborado pelo autor

Busca Serial

Ao buscar um componente i_e, em uma posição esperada e, a busca serial [132][136][135] consiste em uma inspeção de cima para baixo, até que o componente seja encontrado. O custo de inspeção de qualquer item i_l é determinado pela fórmula da Figura 4.10.

$$T_{\text{read}}(i_l) = \frac{\delta}{1 + B(i_l)}$$

Figura 4.10: Fórmula do custo de inspeção
Fonte: Adaptada de Todi *et al.* [42]

O tempo total de busca serial para um alvo no local esperado e é determinado pela fórmula da Figura 4.11.

$$T_{\text{serial}}(i_e) = \sum_{j=1}^{e} T_{\text{read}}(i_j) + T_{\text{trail}}$$

Figura 4.11: Fórmula do tempo total de busca serial
Fonte: Adaptada de Todi *et al.* [42]

Em uma IU adaptada, se a nova localização do componente a estiver antes da localização esperada e, o tempo de busca diminui, conforme a equação da Figura 4.11. No entanto, se a estiver depois de e, a penalidade T_c é adicionada, ao não encontrar o componente como esperado. Em seguida, o usuário inspeciona, cuidadosamente, os itens restantes em um ritmo mais lento δ. O tempo total de busca é determinado pela fórmula da Figura 4.12.

$$T_{\text{serial}}(i_a) = T_{\text{serial}}(i_e) + T_c + (a - e) \cdot \delta + T_{\text{pointing}}$$

Figura 4.12: Fórmula do tempo total de busca serial
Fonte: Adaptada de Todi *et al.* [42]

Para oferecer suporte à busca serial, é vantajoso para um sistema adaptativo mover os itens usados com frequência para o topo.

Busca de Rechamada

A busca de rechamada (direta) [132] depende da memória do usuário, fornecida pelas ativações B, para olhar diretamente para os componentes, sem inspecionar toda a lista. Para um componente de destino i, se não houver ativações B_i acima de um limite θ (usa-se 0,5), o usuário reverte para a busca serial. Quando $B(i_l) >= \theta$ para um alvo i_l na localização l, o usuário tenta a busca de rechamada, ao inspecionar o item em l. Se encontrado em l, o usuário executa uma tarefa de apontar o mouse. Aqui, a busca visual e a tarefa de apontar o mouse são executadas sequencialmente, pois o movimento dos olhos é mais rápido que o movimento do mouse [132]. Usa-se a Lei de Fitts para estimar o tempo de apontar em listas, conforme a fórmula da Figura 4.13.

$$T_{\text{pointing}}(i_l) = a_p + b_p \cdot \log(1 + i_l)$$

Figura 4.13: Fórmula da Lei de Fitts
Fonte: Adaptada de Todi *et al.* [42]

Se não for encontrado em l, após incorrer na penalidade T_c, o usuário tenta a busca local, ao inspecionar, aleatoriamente, N_{local} componentes nas proximidades da localização l, definido na fórmula da Figura 4.14.

$$T_{\text{local}}(i_l) = T_c + \delta \cdot N_{\text{local}} + T_{\text{trail}}$$

Figura 4.14: Fórmula da busca local com inspeção aleatória
Fonte: Adaptada de Todi *et al.* [42]

Em listas não ordenadas, N_{local} é igual a duas vezes o número de componentes na fóvea. Em uma IU adaptativa, o componente de destino i pode ter sido encontrado em vários locais. Aqui, o usuário tenta a busca de rechamada, em todos os locais $l \in L$ onde

$B(i_l) >= \theta$, até que o alvo seja encontrado. O tempo total de busca de rechamada para o alvo i_a no local adaptado a é determinado pela fórmula da Figura 4.15.

$$T_{\text{recall}}(i_a) = T_{\text{pointing}}(i_a) + \sum_{l \in L: B(i_l) \geqslant \theta}^{l \pm N/2} T_{\text{read}}(i_l) + T_{\text{local}}(i_l)$$

Figura 4.15: Fórmula do tempo total de busca de rechamada
Fonte: Adaptada de Todi *et al.* [42]

Se a busca de rechamada falhar ($a \notin LouB(t, a) <= \theta$), o usuário, eventualmente, reverte para a busca serial com cautela. Para melhorar a recordação do usuário, é vantajoso para um sistema adaptativo colocar componentes frequentemente encontrados em locais onde já foram vistos antes.

Rede Neural para Estimar a Recompensa

Os modelos de busca mencionados anteriormente permitem prever o tempo que os usuários levam para fazer seleções, de acordo com suas estratégias. É possível estimar o impacto das adaptações no desempenho do usuário, ao simular seus efeitos. No entanto, ao lidar com listas de tamanhos variados, a execução de simulações com períodos longos pode ser inviável, em um ambiente on-line. Para lidar com problema dessa monta, é utilizada a arquitetura de rede geral (mostrada na Figura 4.4) para IU adaptativa. A ideia principal é antecipar as recompensas associadas a determinada adaptação, conforme a interface anterior e o comportamento do usuário. O modelo ilustrado na Figura 4.16 recebe duas entradas:

1. **interface**: que inclui o *design* da página adaptada e as matrizes de associação da IU atual e adaptada;

2. **usuário**: que inclui a distribuição anterior e atual de cliques. O modelo gera predições de recompensa para cada um dos dois modelos de busca: serial e rechamada.

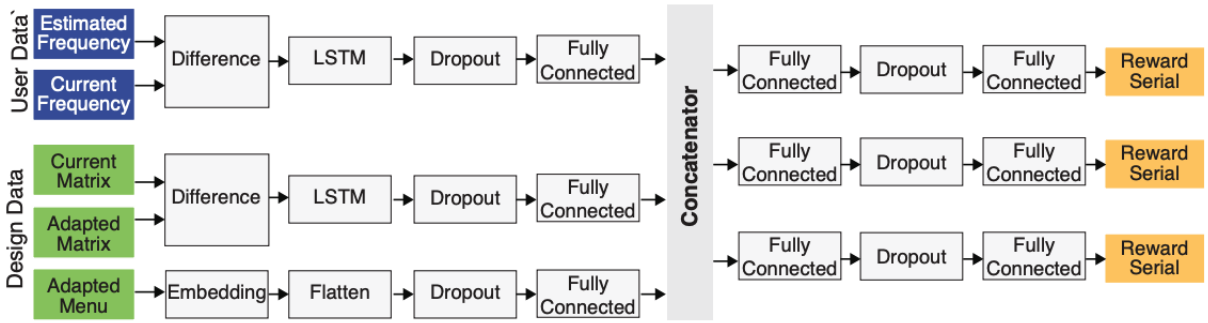

Figura 4.16: Arquitetura de rede de valor considera a UI e os dados do usuário como entrada para fornecer predições de recompensa individuais

Fonte: Adaptada de Todi *et al.* [42]

Para alcançar as recompensas desejadas, cada entrada é tratada como uma ramificação de modelo separada, ou "cabeça", que é, posteriormente, concatenada e passada para três ramificações de modelo independentes, ou "caudas". Para processar cada item da interface adaptada, primeiro é convertido em um vetor codificado *one-hot*, que é achatado e, depois, alimentado em uma camada totalmente conectada. As matrizes de associação passam por um processo diferencial, antes de passarem por uma Camada de Memória de Longo Prazo (LSTM) e, depois, para uma camada totalmente conectada. Além disso, passa-se o histórico de cliques por uma LSTM, projetada para modelar dados sequenciais e lidar com dependências de longo prazo, antes de passar para uma camada totalmente conectada.

As entradas concatenadas são alimentadas em cada uma das caudas da rede, que consistem em duas camadas totalmente conectadas. Para prever cada recompensa, é empregada a ativação linear, no fim de cada cauda, pois elas não estão vinculadas a limites específicos. Para evitar o *overfitting* dos dados de treinamento e melhorar a capacidade de generalização do modelo para dados não vistos, utilizam-se camadas *dropout* com uma taxa de queda de 0,5, antes de cada uma das camadas totalmente conectadas, como uma técnica de regularização. A função de perda para todas as caudas do modelo é o erro quadrático médio (MSE), que calcula a média das diferenças quadráticas entre os valores preditos e reais, e penaliza grandes erros. Adota-se o otimizador RMSProp, um algoritmo de descida de gradiente estocástico popular, com uma taxa de aprendizado de $\eta = 0{,}001$ e fator de decaimento $\beta = 0{,}9$ para otimização.

4.3 Arquitetura da Aplicação

Antes de apresentar os elementos da arquitetura, esta seção traz uma visão geral da arquitetura proposta por este trabalho. Esta seção diz respeito à consecução do Objetivo

Específico 4.3.

Como pode ser observado na Figura 4.17, primeiramente, há uma separação entre o *frontend* (1) e o *backend* (2), e o primeiro é uma aplicação web executada em um navegador. Essa aplicação é adaptativa (9), ou seja, é atualizada em tempo real, conforme a interação do usuário. Os dados do usuário, assim como os dados da navegação, como clique do mouse, são enviados por comunicação assíncrona (10) para o *backend* (2), especificamente, para os microsserviços do algoritmo (11). Esses dados processarão as informações e responderão, assincronamente (10), para o *frontend* (1), se algum aplicativo (6) precisar mudar de posição. Essa definição está relacionada ao Objetivo Específico 4.

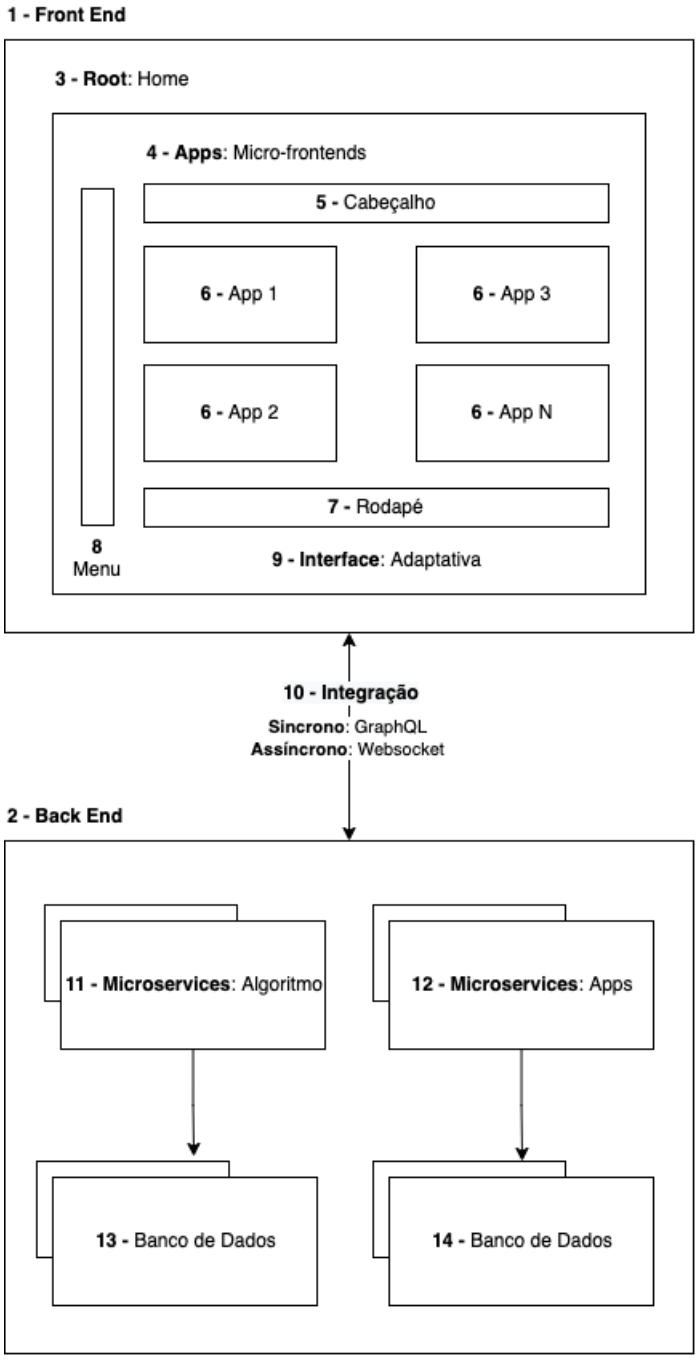

Figura 4.17: Visão geral da arquitetura proposta por este trabalho
Fonte: Elaborada pelo autor

Diante desse cenário, cada aplicativo *App* (6) que forma a aplicação *Home* (3) é implementado, por meio de técnicas da arquitetura de *micro-frontends* (4); com isso, pode ser desenvolvido por equipes e tecnologias distintas, como apresentado na Seção 2.8. Além disso, a implantação deve ser feita de maneira independente. Para tanto, a aplicação *Root* (3) pode ser implementada, por meio do *plugin* do *Webpack* [137], chamado *Module Fede-*

ration [138]; e assim, pode exibir os aplicativos (6), por intermédio do link do ambiente que o código está implantado, independentemente de qual seja o servidor que hospeda esse código.

Já no *backend* (2), todos os componentes devem seguir a arquitetura de microsserviços (11) (12), conforme descrito na Seção 2.5. O ideal é que cada aplicativo do *frontend* (1) tenha o seu próprio conjunto de microsserviços (12). Da mesma forma, o próprio aplicativo deve ser configurado para requisitar o seu *backend*. Deve ser analisado o caso de cada microsserviço, para entender qual categoria de banco de dados (14) é mais adequada para o negócio. De qualquer forma, cada microsserviço deve ter o seu banco de dados, em vez de compartilhá-lo. Ademais, esses microsserviços devem ser acessíveis por APIs GraphQL [139] (10), tanto para comunicação síncrona quanto assíncrona. Por fim, os microsserviços que recebem e processam os dados dos usuários devem usar banco de dados chave-valor, adequados para essa categoria de desenvolvimento, conforme descrito na Seção 2.6. Essa definição está de acordo com o Objetivo Específico 4.3.

Em relação à comunicação entre os dois componentes principais (10), *backend* e *frontend*, que pode ser assíncrona ou síncrona, é recomendado utilizar o GraphQL [139]. O GraphQL utiliza a busca de dados declarativos para suas consultas, e assim, beneficia os usuários, pois os dados, que incluem seus campos e entidades, podem ser selecionados com uma única solicitação de consulta. As consultas GraphQL são eficientes, pois o requisitante pode selecionar quais dados devem ser retornados pelos microsserviços, que evita o tráfego de rede e processamento de dados desnecessários. Também, é possível utilizar comunicação assíncrona, por meio das subscrições do GraphQL; desse modo, os dados dos usuários são enviados para o *backend*, de maneira não bloqueante, o que torna a navegação do usuário mais eficiente e fluída.

Quando o usuário clicar em alguma aplicação (4) listada na *Home* (3), uma nova aba do navegador deverá ser aberta e o aplicativo (6) deve ser exibido pelo *Root*, como representado na Figura 4.18.

A implantação de aplicações que seguem a arquitetura proposta pode ser realizada na nuvem AWS, um provedor de computação em nuvem, o qual permite utilizar o sistema operacional, linguagem de programação, plataforma de aplicativo da web, banco de dados e outros serviços. Além de oferecer uma infraestrutura de computação global escalável, confiável e segura, habilita as aplicações a aumentar ou diminuir o poder computacional com base na demanda [140]. Esses atributos contribuem para cumprir os objetivos específicos , e, consequentemente, o Objetivo Geral descrito na seção 1.3.1.

Na Figura 4.18, é possível observar que o *Root* (2) contém elementos padrão para manter a identidade visual da aplicação na totalidade.

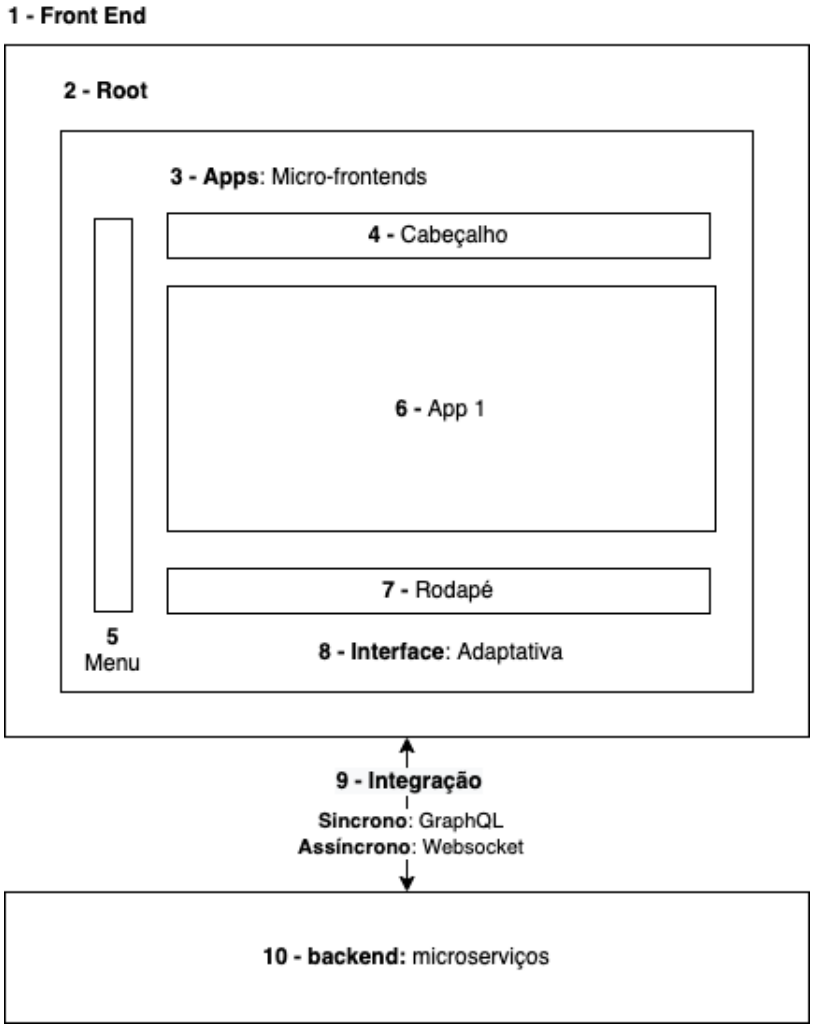

Figura 4.18: Visão geral de um aplicativo que segue esta arquitetura
Fonte: Elaborada pelo autor

Nesse caso, os elementos Cabeçalho (4), Rodapé (7) e Menu (5) ainda são exibidos para o usuário, com o aplicativo App 1 (6). Para exemplificar, esse aplicativo pode ser: extrato bancário, transferência bancária ou empréstimo. Ressalta-se que esse aplicativo requisita os seus próprios microsserviços *backend* e continua a monitorar os dados e as interações dos usuários. A aplicação *Home* (3), da Figura 4.17, age como uma vitrine de aplicativos posicionados, conforme a decisão do algoritmo.

4.3.1 Arquitetura Orientada a Serviços (SOA)

A arquitetura proposta segue o estilo arquitetural SOA. Ao considerar a arquitetura SOA, é útil revisar os principais termos relacionados a ela.

- Serviço: é um componente de software que pode ser acessado por uma rede para fornecer funcionalidade a um solicitante de serviço.
- SOA: refere-se a um estilo de construção de sistemas distribuídos confiáveis, que oferecem funcionalidade por serviços, com destaque especial para o baixo acoplamento entre esses serviços disponibilizados.

A SOA enfatiza a implementação de componentes como serviços modulares, que podem ser descobertos e utilizados pelos clientes. Os serviços, geralmente, apresentam as seguintes características:

- podem ser úteis, individualmente ou compostos, para fornecer serviços de nível superior. Isso favorece a reutilização da funcionalidade existente, além de trazer outros benefícios;
- a comunicação entre os serviços e seus clientes ocorre por troca de mensagens, definidas pelas mensagens que podem aceitar e pelas respostas que podem dar;
- participam de um fluxo de trabalho em que a ordem das mensagens enviadas e recebidas influencia o resultado das operações executadas por um serviço. Esse conceito é conhecido como coreografia de serviços [141];
- podem ser completamente independentes ou depender da disponibilidade de outros serviços, ou de um recurso, como um banco de dados;
- anunciam detalhes como suas capacidades, interfaces, políticas e protocolos de comunicação suportados. Os detalhes de implementação, como a linguagem de programação e a plataforma de hospedagem, não interessam aos clientes e não são revelados.

O tipo de integração entre os serviços pode ser dividido entre síncronos e assíncronos. Os mais comuns para conexão assíncrona são SOAP e REST, além de tecnologias mais recentes como o GraphQL [142]:

- REST: nesse conector, um consumidor de serviço envia solicitações HTTP síncronas. Essas solicitações utilizam, principalmente, quatro verbos HTTP básicos (*post, get, put* e *delete*) para informar ao provedor de serviços para criar, recuperar, atualizar ou excluir um recurso, respectivamente. O formato do dado pode ser XML, JSON ou notação semelhante [143];
- SOAP: protocolo padrão para comunicação na tecnologia de serviços da Web. Os consumidores e provedores de serviços interagem ao trocar mensagens XML de solicitação/resposta, geralmente, sobre HTTP;

- GraphQL: é uma linguagem de consulta e ambiente de execução processada no lado do servidor para as APIs, a qual retorna exata e somente os dados que os clientes solicitam [139].

Em uma comunicação assíncrona, os serviços trocam mensagens por sistemas de mensageria ou barramentos de eventos, como IBM WebSphere MQ [144], Microsoft MSMQ [145], Apache ActiveMQ [146] ou Amazon EventBridge [147]. A comunicação assíncrona, geralmente, oferece grande confiabilidade e escalabilidade e independe do sistema utilizado para gerenciar a comunicação; assim, é necessário escolher qual o tipo de mensagem a ser utilizada. Desenvolver um sistema que utiliza a comunicação assíncrona implica em seguir os conceitos da Arquitetura Orientada a Eventos (EDA). Como dizem Laliwala e Chaudhary [148], ao contrário da arquitetura tradicional baseada em chamadas síncronas, em que cada serviço espera por uma resposta antes de prosseguir, a EDA permite que os serviços se comuniquem sem a necessidade de um serviço permanecer bloqueado, enquanto aguarda a resposta de uma consulta realizada a outro serviço. Isso torna as aplicações mais escaláveis e menos acopladas, além de tornar a arquitetura altamente flexível. Uma das principais vantagens da EDA é que ela pode ser adaptada a sistemas de qualquer tamanho. Isso significa ser possível implementá-la em sistemas pequenos ou grandes, sem precisar alterar a arquitetura existente de forma complexa.

4.3.2 Arquitetura Orientada a Eventos

A EDA descreve um tipo de sistema de microsserviços, em que os serviços trocam informações por eventos, que resulta em um sistema fracamente acoplado. Os serviços utilizam eventos de publicação e escuta para absorver e transmitir informações no ecossistema orientado a eventos [149], como mostrado na Figura 4.19.

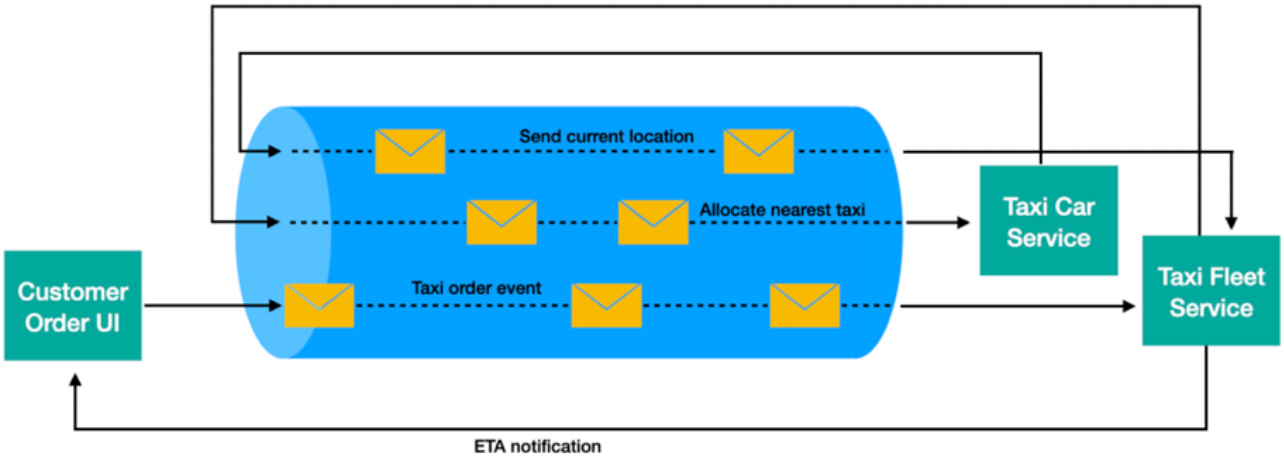

Figura 4.19: Processo de comunicação da EDA em microsserviço
Fonte: Adaptada de Rahma-tulloh *et al.*[150]

Ao contrário da Arquitetura Orientada a APIs, que se comunica de forma síncrona por chamadas de APIs, a EDA se comunica de forma assíncrona por eventos de mensagem [151]. A EDA ganhou popularidade em vários domínios, como detecção de intrusão de rede, redes de sensores, negociação no mercado de ações, controle de sistema em tempo real, monitoramento de saúde, mobilidade e computação vestível, devido à sua capacidade de desenvolver sistemas distribuídos serem altamente flexíveis e simultâneos [150].

Ressalta-se que este trabalho segue a EDA tanto para os serviços que compõe o *backend*, quanto para a comunicação entre o *backend* e o *frontend*, como apresentado nas Seções 4.3.3 e o 4.3.4.

Como mostrado por Jansen e Saladas [149], a Figura 4.19 segue a arquitetura de microsserviços e EDA: e nesse caso, três microsserviços estão envolvidos em tal cenário: a IU, a qual um cliente pode pedir um táxi, um serviço de frota que atribui táxis a pedidos e um serviço que coleta dados acerca dos táxis. O cilindro no centro do diagrama, que liga os diferentes microsserviços, representa o barramento de mensagens orientado a eventos. As setas no diagrama representam o fluxo de evento:

- o usuário faz um pedido de táxi, por meio da IU do pedido do cliente, que captura informações como a localização atual do usuário;
- o *Taxi Fleet Service* subscreve o evento de ordem de táxi (escutar esse tipo de evento);
- o *Taxi Car Service* coleta dados dos táxis individuais, como a localização atual de cada táxi e os envia;

- o *Taxi Fleet Service*, que está inscrito nos eventos de localização atual, aloca o táxi mais próximo ao usuário e envia um evento de alocação para o veículo mais próximo;
- o *Taxi Car Service* se inscreve nos eventos de táxi alocados mais próximos e avisa o motorista que precisa buscar um usuário;
- o *Taxi Fleet Service* pode monitorar, constantemente, a localização do táxi e atualizar a IU com notificações.

4.3.3 Arquitetura do *Frontend*

Micro-Frontend é um conceito de microsserviço que tem se tornado cada vez mais popular, devido a sua capacidade de aumentar a flexibilidade no desenvolvimento. Com essa abordagem, as equipes podem combinar componentes criados, com a mesma estrutura e várias estruturas ou bibliotecas diferentes, de modo a permitir o trabalho de forma independente, com agilidade e eficiência. Além disso, o uso de *micro-frontends* também pode ajudar a reduzir a complexidade dos aplicativos, ao torná-los mais fáceis de gerenciar e manter [152]. Os principais benefícios dessa categoria de arquitetura são [153]:

1. **implantação independente**: ajuda a limitar o escopo das implantações. Todas as bases de código desacopladas devem ter seus próprios pipelines de *Continuous Integration/Continuous Delivery* (CI/CD) para que as equipes possam determinar, independentemente se seu aplicativo está pronto para produção e, em caso afirmativo, se nenhum problema de outros aplicativos o prejudicará;
2. **equipes independentes**: com as bases de código são separadas e as implantações autônomas, as equipes podem controlar o projeto do início ao fim, e assim, ter controle total sobre suas entregas;
3. **fácil manutenção e correção de problemas**: os *micro-frontends* empregam a técnica de dividir e conquistar. A manutenção é facilitada, pois, cada equipe é responsável por uma aplicação com escopo limitado para realizar a manutenção dos requisitos de negócios, também, a correção de erros;
4. **tecnologia agnóstica**: as equipes podem criar aplicações, por meio de uma pilha tecnológica diferente. Uma delas é que pode dividir uma equipe em outras menores, com base na competência, em uma pilha de tecnologia específica, que também adere à ideia de responsabilidade única. Em segundo lugar, como muitas pilhas de tecnologia serão usadas no mesmo projeto, será mais fácil contratar novos desenvolvedores;

5. **unidades simples e desacopladas:** um monólito tem uma abundante quantidade de código que cresce com o tempo. Pelo contrário, o código-fonte para *micro-frontends* seria, significativamente, menor. Além disso, nesse modelo de arquitetura, as equipes têm que criar limites adequados entre aplicativos e evitar todos os acoplamentos que possam existir, involuntariamente, em monólitos.

Com base nesse conceito, Zack Jackson [154] projetou um *framework* em Javascript, chamado *Module Federation*. A ideia é tornar o compartilhamento de código mais gerenciável e autônomo. Ele permite que um programa JavaScript importe código de outro módulo, por meio de configurações no *Webpack*. Cada módulo é um arquivo JavaScript exclusivo. Esse arquivo pode ser acessado por outros aplicativos, conforme a imagem 4.20.

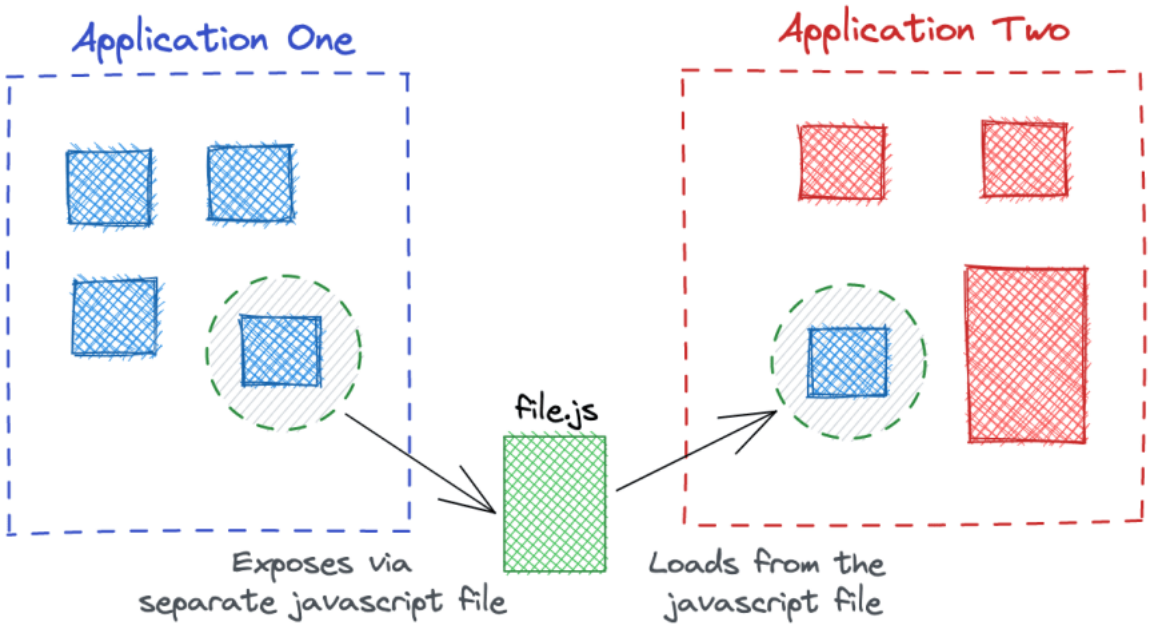

Figura 4.20: Visão geral de um aplicativo que segue a arquitetura *frontend*
Fonte: Adaptada de Perera [155]

O conceito de *micro-frontends*, com a facilidade de desenvolvimento com *framework Module Federation*, ajuda a cumprir o Objetivo Geral deste trabalho, no que tange ao desenvolvimento de sistemas reativos. Como explicado na Seção 4.3, a arquitetura final do *frontend* pode ser constatada na Figura 4.21.

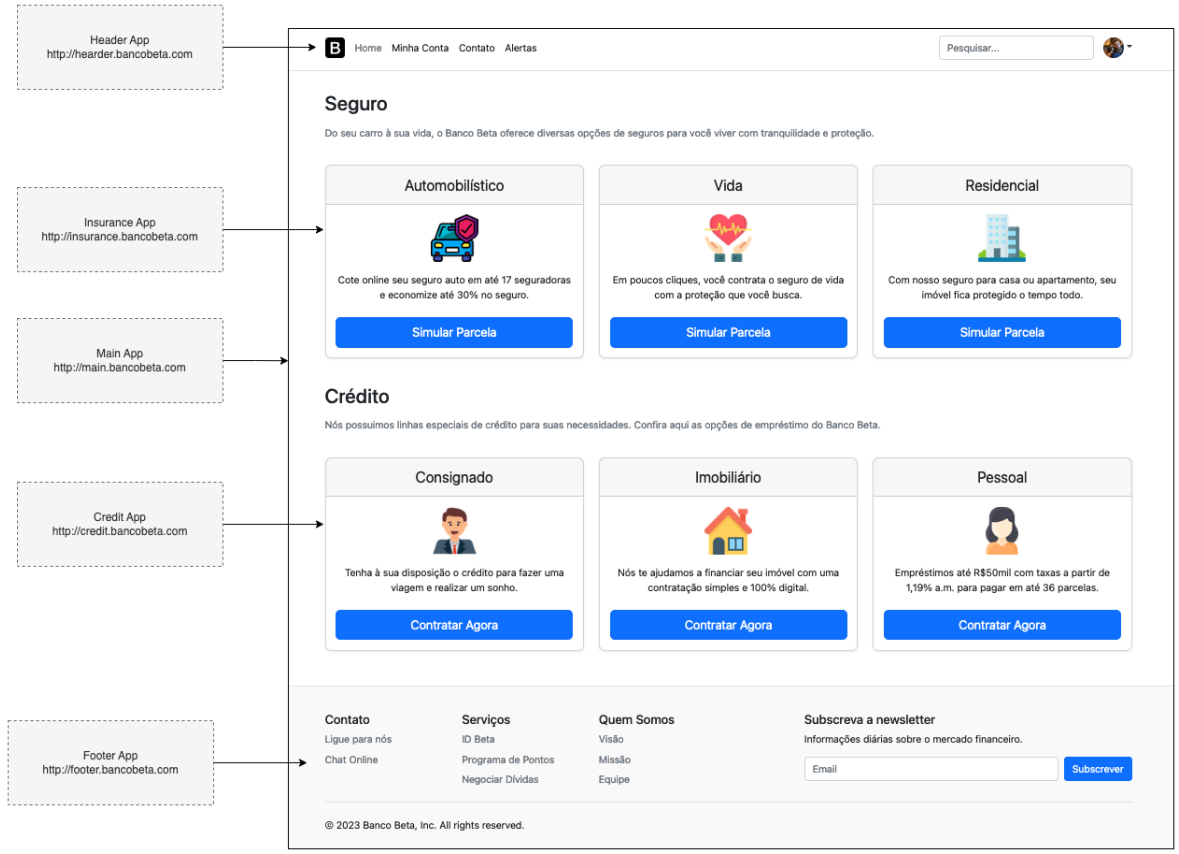

Figura 4.21: Arquitetura do *frontend* formada por *micro-frontends*
Fonte: Elaborada pelo autor

A página é formada por múltiplos aplicativos-filhos, em que cada um pode ser desenvolvido por uma equipe diferente; e ter uma pilha de tecnologias diferentes. A função e a descrição dos aplicativos que forma a página principal (vitrine) são:

1. ***Header App***: aplicativo responsável por apresentar o cabeçalho para o usuário. Como pode ser observado na imagem, o endereço dele difere dos demais, pois é um aplicativo totalmente separado. Se, por acaso, a equipe de desenvolvimento implanta o aplicativo com erros, então, o *header* não será apresentado; entretanto, os demais serão apresentados normalmente;

2. ***Insurance App***: a página principal é composta por diversos aplicativos-filho, o *Insurance App* é um deles. Nesse caso, segue um layout com três colunas e cada coluna contém o link para o aplicativo determinado. Porém, caso a equipe responsável indique, esse aplicativo pode ser somente uma imagem ou um código HTML;

3. ***Main App***: esse é o aplicativo principal da página, ele tem a lógica para requisitar todos os aplicativos-filhos e deixá-los disponíveis para os usuários. Por isso, ele deve saber buscar e apresentar os arquivos dos aplicativos-filhos;

4. ***Credit App***: assim como o *Insurance App*, esse aplicativo é mais um filho do *Main App*. Nesse caso, possui um layout com três colunas e cada coluna contém o link para o aplicativo determinado. Porém, caso a equipe responsável indique, esse aplicativo pode ser somente uma imagem ou um código HTML;

5. ***Footer App***: assim como o *Header App*, o *Footer App* é o aplicativo responsável por apresentar o rodapé para o usuário. Como pode ser observado na imagem, o endereço dele difere dos demais, pois é um aplicativo totalmente separado. Se, por acaso, a equipe de desenvolvimento implanta o aplicativo com erros, então, o *footer* não será apresentado; entretanto, os demais serão apresentados normalmente. Dessa forma, a instituição pode empregar o conteúdo que for mais relevante, que pode incluir, por exemplo, propagandas e sistemas de recomendação.

A implantação do *frontend* pode ser realizada de diversas maneiras, cada *micro-frontend* pode estar implantado em um local diferente, com suas próprias esteiras de construção e entrega contínua. A Figura 4.22 exemplifica a implantação na AWS, em que cada *micro-frontend* tem o seu código já construído e compilado em um *Bucket S3* distinto.

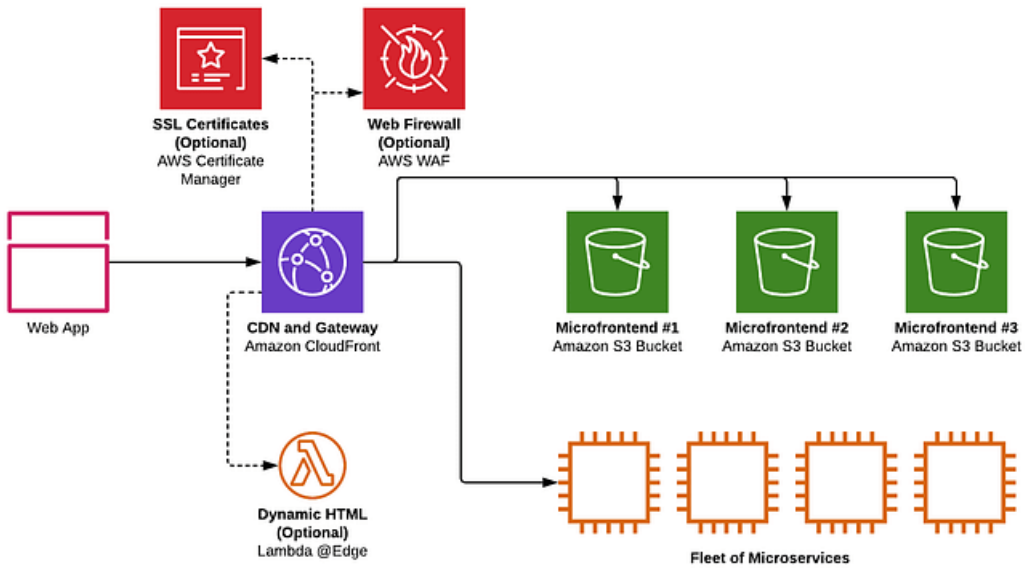

Figura 4.22: Implantação de *micro-frontends* na AWS
Fonte: Elaborada pelo autor

O aplicativo *Main App* conhece as localizações, requisita os arquivos via requisição HTTP e *Module Federation*; por último, apresenta a página completa para o usuário, como mostrado no exemplo da Figura 4.22. Por outro lado, também existe um conjunto de microsserviços implantados. Cada *micro-frontend* deve requisitar o seu próprio conjunto de microsserviço.

4.3.4 Arquitetura do *Backend*

Como já detalhado na Seção 4.3.2, o *backend* sugerido por este trabalho segue as especificações da EDA. Dessa forma, os microsserviços são acionados por eventos assíncronos.

Como apresentado na Figura 4.23, quando o usuário acessa a página inicial da aplicação, há uma primeira requisição síncrona (*AppSync*).

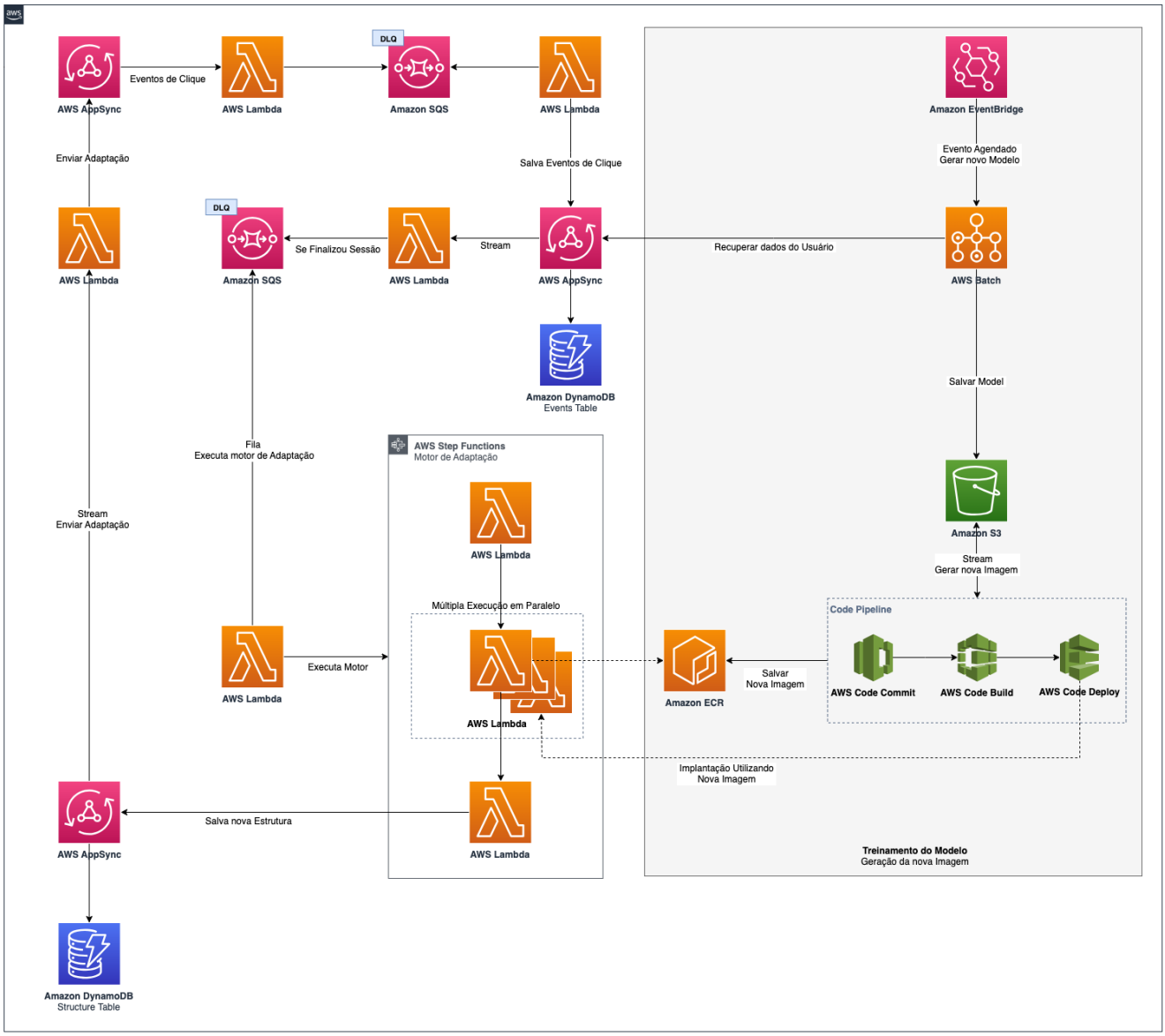

Figura 4.23: *Backend* que gerencia as páginas adaptativas
Fonte: Elaborada pelo autor

Com o resultado dessa requisição, o *Main App* consegue apresentar a IU adaptada ao usuário. A Figura 4.24 apresenta um exemplo de resposta dessa primeira requisição.

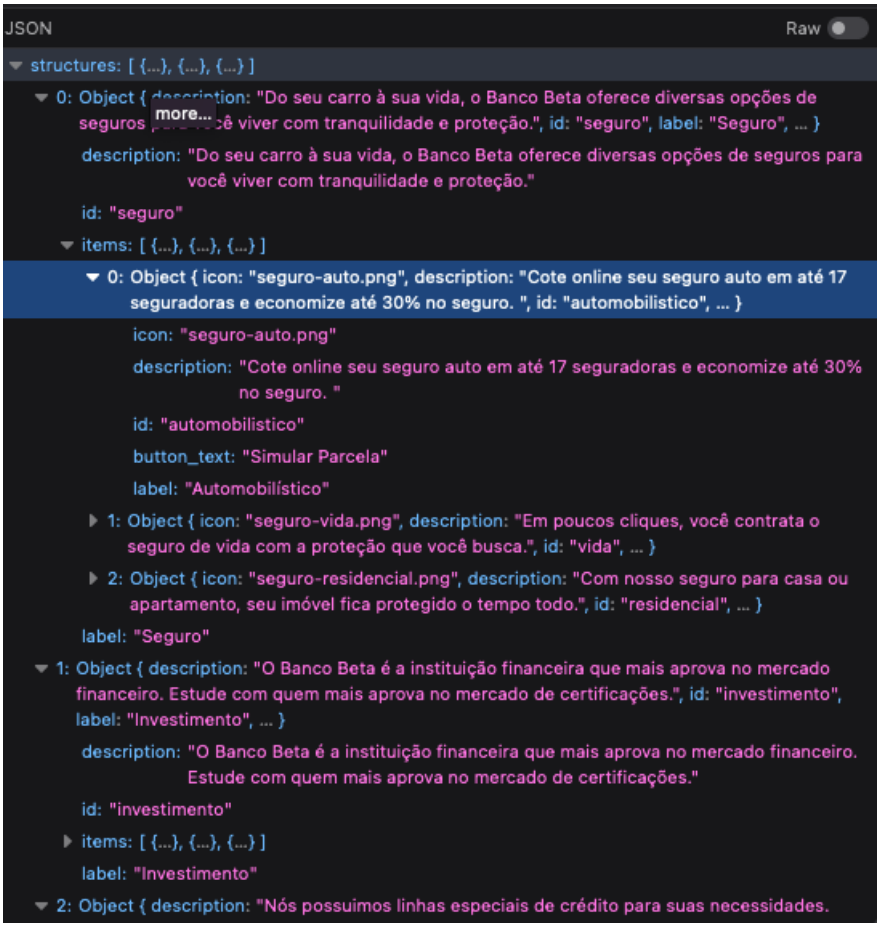

Figura 4.24: Exemplo da resposta para apresentar a IU adaptativa
Fonte: Elaborada pelo autor

Essa primeira requisição tem uma resposta que contém as aplicações que devem ser apresentadas, a ordem de apresentação e a informação necessária para preencher os componentes. Porém, essa informação está armazenada em um banco de dados, preenchida pelo motor de adaptação. Isto é, cada clique que o usuário realiza na tela, é enviado para o *backend*, de forma assíncrona, via comunicação *websocket*. O microsserviço (*AWS Lambda*) de eventos recupera e armazena esse novo evento em um banco de dados do tipo chave-valor. Esse banco de dados, no que lhe concerne, ao armazenar as informações, envia um evento para o microsserviço, que inicia a execução do motor de adaptação. Caso o motor detecte que uma nova estrutura deve ser apresentada ao usuário, de forma assíncrona, será enviada essa informação também via *websocket* para o *frontend*. Por fim, o *frontend* apresentará a nova estrutura para o usuário, de maneira automática.

O motor de adaptação é composto por três passos:

1. pré-processamento: recupera os dados do banco de dados, com *queries* GraphQL, processa e transforma os dados para estarem no formato esperado pela execução do

algoritmo;

2. processamento: executa o algoritmo explicado na Seção 4.2. A execução deve ocorrer de forma paralela, assim, será possível obter múltiplos resultados, que serão avaliados no passo seguinte;

3. pós-processamento: recebe múltiplos resultados da execução em paralelo e escolhe a adaptação que tenha a melhor recompensa. Por último, salva a nova adaptação no banco de dados e envia para usuário final de forma assíncrona.

O motor de adaptação utiliza um modelo pré-treinado para executar o algoritmo, a arquitetura também prevê o fluxo de treinamento e implantação do modelo e do algoritmo. Esse fluxo também pode ser observado na Figura 4.23, descrito como: treinamento do modelo e geração de nova imagem. O fluxo de treinamento é iniciado por um evento agendado, que aciona o processamento (*AWS Batch*). Esse processamento recupera os dados requeridos, realiza o treinamento e salva o modelo no Amazon S3. Este último, no que lhe concerne, envia um evento que aciona o fluxo de implantação. Esse fluxo recupera o modelo do Amazon S3, gera uma nova imagem de contêiner e implanta essa nova versão.

Como já definido, cada *micro-frontend* deve ter o seu próprio *backend*, que deve ser composto por um conjunto de microsserviços. No mesmo sentido, cada microsserviço deve ter o seu próprio banco de dados, de acordo com o padrão banco de dados por microsserviço, como mostra a Figura 4.25.

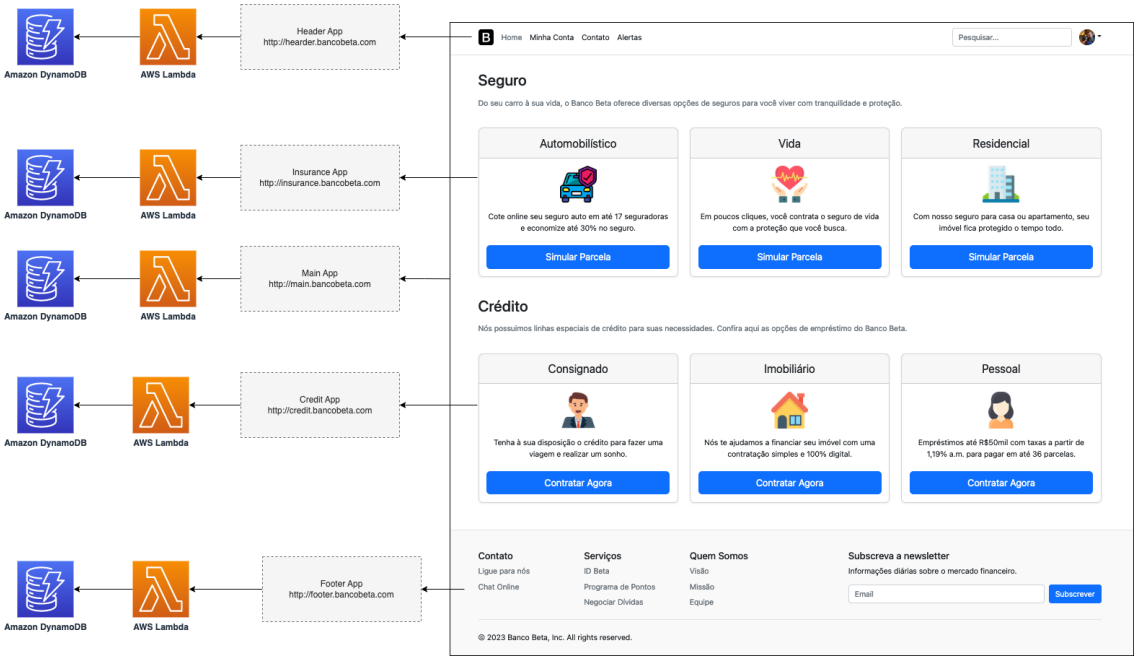

Figura 4.25: Padrão Banco de dados por Microsserviço
Fonte: Elaborada pelo autor

O baixo acoplamento é a principal característica de uma arquitetura de microsserviços, porque cada microsserviço individual pode armazenar e recuperar informações de seu próprio armazenamento de dados, de forma independente. Ao implantar o padrão de banco de dados por serviço, é possível escolher os armazenamentos de dados mais apropriados (por exemplo, bancos de dados relacionais ou não relacionais) para o aplicativo e requisitos de negócios. Isso significa que os microsserviços não compartilham uma camada de dados; as alterações no banco de dados individual de um microsserviço não afetam outros microsserviços; os armazenamentos de dados individuais não podem ser acessados, diretamente, por outros microsserviços; e os dados persistentes são acessados apenas por APIs. A dissociação de armazenamentos de dados também melhora a resiliência de seu aplicativo geral e garante que um único banco de dados não seja um único ponto de falha.

Na arquitetura prevista, é utilizado o padrão de mensagens de desacoplamento, por meio do Amazon SQS. Esse padrão fornece comunicação assíncrona entre microsserviços, por meio de um modelo de pesquisa assíncrono. Quando o sistema de *backend* recebe uma chamada, ele responde, imediatamente, com um identificador de solicitação; e, em seguida, processa a solicitação, de forma assíncrona [140].

A arquitetura prevista na Figura 4.23 utiliza os serviços do provedor de serviços em nuvem *Amazon Web Services* [**?**], descritos como:

- *AWS Lambda*: serviço de computação sem servidor e orientado a eventos, que permite executar código para, praticamente, qualquer tipo de aplicação ou serviço de *backend*, sem provisionar ou gerenciar servidores;
- *AWS AppSync*: cria APIs GraphQL e Pub/Sub com tecnologia sem servidor, que simplificam o desenvolvimento de aplicações, com um único *endpoint* para consultas, atualizações ou publicações de dados, de forma segura;
- *Amazon SQS*: permite que seja enviado, armazenado e recebidas mensagens entre componentes de software, em qualquer volume, sem perder mensagens ou precisar que outros serviços estejam disponíveis;
- *Amazon DynamoDB*: banco de dados de chave-valor NoSQL, sem servidor e totalmente gerenciado, projetado para executar aplicações de alto desempenho, em qualquer escala. Também oferece segurança integrada, *backups* contínuos, replicação multirregional automatizada, armazenamento em cache na memória e ferramentas de importação e exportação de dados;
- *AWS Batch*: permite que desenvolvedores, cientistas e engenheiros executem, com eficiência, milhares de tarefas de computação em lote e em ML enquanto otimizam recursos computacionais;

- *Amazon S3*: serviço de armazenamento de objetos que oferece escalabilidade, disponibilidade de dados, segurança e desempenho líderes do setor. Clientes de todos os portes e setores podem armazenar e proteger qualquer quantidade de dados de, praticamente, qualquer caso de uso, como *data lakes*, aplicações nativas da nuvem e aplicações móveis;
- *AWS ECR*: registro de contêiner totalmente gerenciado, que oferece hospedagem de alto desempenho para implantar imagens e artefatos de aplicações, de forma confiável, em qualquer lugar;
- *AWS Step Functions*: serviço de fluxo de trabalho visual, que ajuda os desenvolvedores a usar os produtos da AWS para desenvolver sistemas distribuídos, automatizar processos, orquestrar microsserviços e criar pipelines de dados e aprendizado de máquina;
- *AWS Code Commit*: serviço de controle de código-fonte totalmente gerenciado, seguro e altamente escalável, que hospeda repositórios privados do Git;
- *AWS Code Build*: permite a integração contínua, totalmente gerenciada, que compila código-fonte, executa testes e produz pacotes de software prontos para implantação;
- *AWS Code Deploy*: permite a implantação totalmente gerenciada que automatiza implantações de software em vários serviços de computação.

4.4 Projeto da Arquitetura

A arquitetura de software é um conjunto de elementos arquiteturais (de dados, de processamento, de conexão) que possuem alguma organização [156]. Os elementos e sua organização são definidos por decisões tomadas para satisfazer objetivos e restrições [156].

Além disso, a arquitetura de software é projetada para atender aos requisitos funcionais e não funcionais do sistema, responsável por definir o que é o sistema em componentes computacionais, os relacionamentos entre estes componentes e os padrões que guiam a sua composição e restrições [157], como explicado na Seção 2.3. Desses relacionamentos, emergem os atributos de funcionalidade e qualidade que os *stakeholders* do sistema exigem: segurança, modificabilidade, desempenho e assim por diante [47]. Quanto maior e mais complexo o sistema, mais crítico é esse particionamento, portanto, a arquitetura. E, como este trabalho trata de sistemas bancários que são, originalmente, mais exigentes, esses atributos de qualidade tornam a arquitetura um tema crítico e essencial.

Diante disso, esta Seção 4.4 apresenta o projeto da arquitetura para aplicação web bancária, que se baseia no roteiro proposto por Rozanski e Woods [117]. A arquitetura proposta usa e estende os modelos definidos por Lewis e Fowler [24], Smet [36], Bonér *et al.* [23], Clements *et al.* [47], Richardson [158], Stopford e Safari [159].

4.4.1 Escopo

O escopo da arquitetura proposta é baseado nos objetivos e nas dificuldades enfrentadas pelos *stakeholders* para desenvolver sistemas robustos, resilientes, flexíveis e ajustados para sustentar as demandas modernas. Como visto no decorrer deste trabalho, a criação de sistemas web bancários que seguem essas características é complexa e faltam guias para a sua implementação e implantação.

Dessa maneira, a arquitetura proposta oferece suporte para os dois componentes principais do sistema: *backend* e *frontend*. O *backend* é responsável por capturar as requisições realizadas pelos usuários por meio do *frontend*, processar e retornar um resultado. Ele também deve conseguir capturar, em tempo real, as interações que o usuário realiza, como cliques do mouse para que esses dados históricos sejam processados, posteriormente, quando não houver uma limitação rígida de tempo. De outra maneira, o *backend* deve seguir as características de sistemas reativos, agir ativamente, sem a necessidade de uma requisição bloqueante, para que o *frontend* se adapte ao usuário, após o algoritmo processar as interações e detectar que a IU deve ser adaptada.

A adaptação do *frontend* acontece em tempo real, isto é, enquanto o usuário estiver navegando. Quando o *backend* enviar uma informação para que a IU seja adaptada, a interface deve realizar as modificações necessárias, de forma automática, natural e não bloqueante [160]. Outras funcionalidades importantes são a captura, a formatação e o envio das interações do usuário para *backend*. Aliado a isso, o *frontend* também deve seguir as características de sistemas reativos.

4.4.2 Interesses

Os interesses da arquitetura proposta por este trabalho incluem componentes para implementação, implantação, manutenção, configuração, métricas de uso, monitoramento e comunicação em sistemas web. A arquitetura dispõe de mecanismos para adaptar, em tempo real, a IU por serviços disponibilizados pelo *backend* e, para isso, tanto as interações dos usuários, quanto a requisição de adaptação vinda do *backend*, devem ser feitas de forma assíncrona e de acordo com uma característica fundamental de um sistema reativo: ser orientado a mensagens Bonér *et al.* [23].

Os componentes internos do *backend* e do *frontend* podem ser implementados, evoluídos, implantados e gerenciados, de forma independente. Assim, componentes com recursos específicos podem ser incluídos ou substituídos, com pouco impacto na aplicação, em sua totalidade. A utilização de microsserviços [24] e *micro-frontends* [36] colabora para que essas características sejam alcançadas, pois, oferecem um mecanismo para que um sistema complexo seja dividido em pequenas partes, que se comunicam entre si, por IU bem definidas.

4.4.3 Princípios

Em um passado recente, as aplicações que precisavam lidar com a alta demanda utilizavam dezenas de servidores, tempo de resposta na casa dos segundos e horas de indisponibilidade para manutenção e gigabytes de dados. Entretanto, a demanda mudou e, atualmente, as aplicações estão disponíveis em diversos locais, simultaneamente, executadas na nuvem em *clusters*, por milhares de processadores *multi-core*, ou seja, devem ser preparadas para lidar com os desafios de sistemas distribuídos. Geralmente, os usuários esperam respostas em milissegundos, com 100% de disponibilidade, e os dados são mensurados em petabytes Bonér *et al.* [23].

Segundo Tanenbaum e Van Steen [161], sistemas distribuídos são uma coleção de sistemas de computador, que utilizam recursos computacionais, em vários pontos centrais diferentes (também chamados nós), para atingir um objetivo comum e compartilhado. Eles dependem dos nós para se comunicar e sincronizar em uma rede comum. Os nós podem representar dispositivos de hardware físicos diferentes, mas também podem ser processos de software diferentes ou outros sistemas encapsulados. Os sistemas distribuídos visam remover gargalos ou pontos centrais de falha de um sistema.

Dessa forma, os princípios de sistemas reativos surgem, a fim de lidar com os desafios do desenvolvimento de sistemas distribuídos, pois, como descrito na Seção 2.4, são responsivos, resilientes, elásticos e orientados a mensagens. Além disso, conseguem reagir às solicitações, resolver falhas, de forma eficiente; manter a capacidade de resposta, independentemente da quantidade de solicitações; e comunicam-se, principalmente, de forma assíncrona por mensagens [162].

4.4.4 Restrições

Como restrição, as ferramentas, bibliotecas e estratégias para implementação devem ser unicamente *Free and Open Source Software* (FOSS), para que a solução possa ser utilizada em órgãos do governo ou empresas privadas, sem as limitações do custo de aquisição. Durante a implantação dos sistemas, é possível escolher entre as principais soluções de

implementação e implantação, como React.js [163], Node.js [164], Python [165], Scikit Learn [113], *Webpack* [137], *Module Federation* [138], Docker [166], MongoDB [167] e GraphQL [139].

Questões a respeito de segurança, acessibilidade, internacionalização e regulação não estão no escopo deste trabalho, porém são sugestões para trabalhos futuros. A segurança dos dados estará a cargo dos administradores da infraestrutura. Por se tratar de um sistema com interface gráfica para o usuário comum, será disponibilizada uma aplicação para realizar a validação da proposta, porém não serão analisados cenários acerca de acessibilidade. A arquitetura proposta prevê sua utilização em organizações públicas e privadas, mas apenas em língua portuguesa. Além disso, a legislação a respeito de regulação não se aplica, porque os dados usados nos experimentos não foram publicados além dos limites legais.

4.4.5 *Stakeholders*

Os *stakeholders* são as partes interessadas na arquitetura proposta. No entanto, não está no escopo deste trabalho prever alguma estratégia para o engajamento dos *stakeholders*. Nesse caso, considera-se que as empresas ou organizações públicas que desejam utilizar a arquitetura proposta estejam comprometidas com a implementação e implantação de sistemas distribuídos, conforme os princípios de sistemas reativos. Para a arquitetura de aplicações web bancárias proposta, os principais *stakeholders* são:

- usuários externos: aqueles que utilizam a aplicação;
- arquitetos de software: aqueles que desenham as soluções da empresa ou organização e conhecem o ambiente e sistemas atuais;
- desenvolvedores de software: aqueles que criam os sistemas;
- administradores de infraestrutura: responsáveis por manter o ambiente do *datacenter*, que incluem: servidores, armazenamento, conexão e banco de dados;
- sistemas legados: aplicações e bancos de dados legados em operação na organização ou fora dela, que necessitam interagir com a aplicação criada, de acordo com arquitetura proposta.

4.4.6 Cenários de Uso

O diagrama de caso de uso que mostra, graficamente, os cenários envolvidos na arquitetura, pode ser conferido na Figura 4.26. Nesse diagrama, estão listadas e delimitadas as

situações típicas de uma aplicação desenvolvida em ambiente corporativo, conforme a arquitetura proposta. Tais cenários descrevem como a aplicação pode ser utilizada. Alguns cenários são independentes, como no caso de Configurar Recursos Computacionais, Criar Banco de Dados e Projetar a Arquitetura da Aplicação.

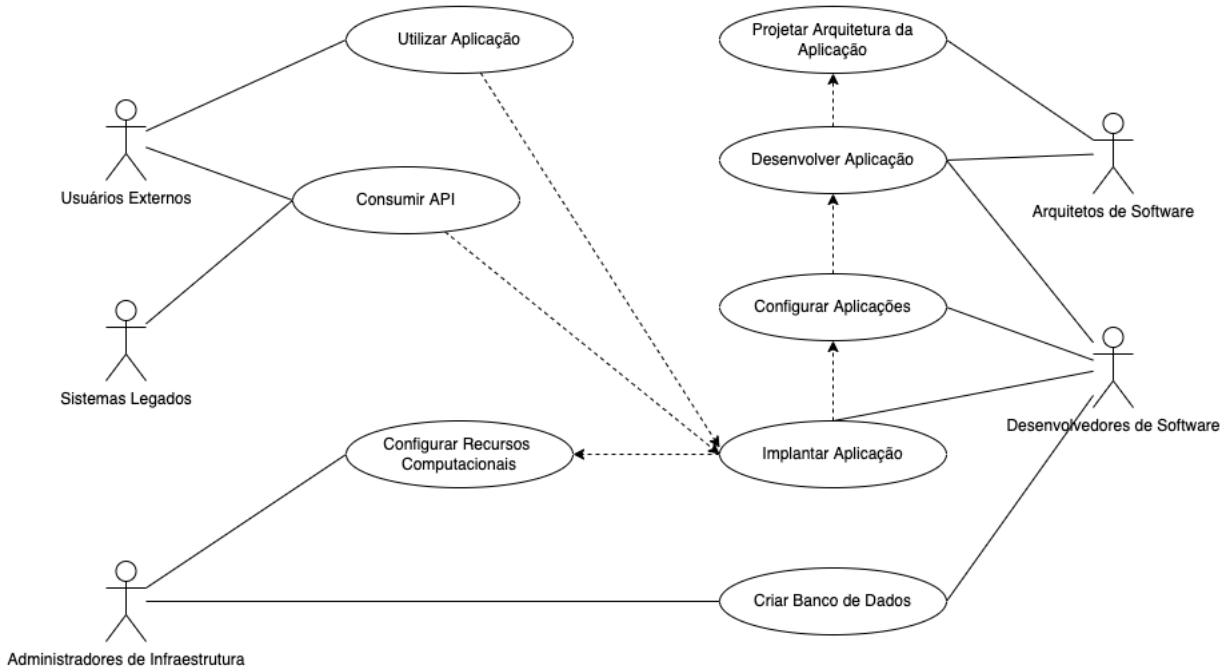

Figura 4.26: Diagrama de caso de uso
Fonte: Elaborada pelo autor

Nesses casos, o usuário pode, simplesmente, executar a ação. Contudo, algumas situações dependem de pré-condições, como o caso de uso Utilizar Aplicação e Consumir, que precisam de uma aplicação desenvolvida, configurada e implantada.

Capítulo 5

Protótipo

O protótipo foi desenvolvido para realizar a validação empíricas com os usuários. Desse modo, seguiu-se, estritamente, a arquitetura proposta por este trabalho.

Para o *frontend*, foi utilizado o *plugin* do *Webpack* [137], chamado *Module Federation* [138]; e o *framework* de Javascript *React*, que está definido na Seção 5.1. Já para os elementos visuais da página, foram utilizados os componentes e módulos do *Bootstrap*, explicados na Seção 5.2. Foram criados três módulos que pertencem ao *Root* da aplicação: *Main*, *Header*, *Footer*, como já explicado na Seção 4.3. Além disso, foram criadas três aplicações do formato de três colunas, previstas na Seção 4.3.3. Para realizar a validação empírica, detalhada na Seção 6.6, foi adicionado um painel de controle, no canto inferior direito da tela. Nesse painel, o usuário conseguiu observar a ação que deveria ser tomada durante a validação. Cada aplicação contém três colunas e cada coluna contém um serviço bancário, composto por: nome do serviço, imagem que o representa e uma descrição, além de um botão que permite o usuário clicar, de modo a acessar a página que representa esse serviço bancário. A Figura 5.1 apresenta a versão final da parte superior da interface do usuário.

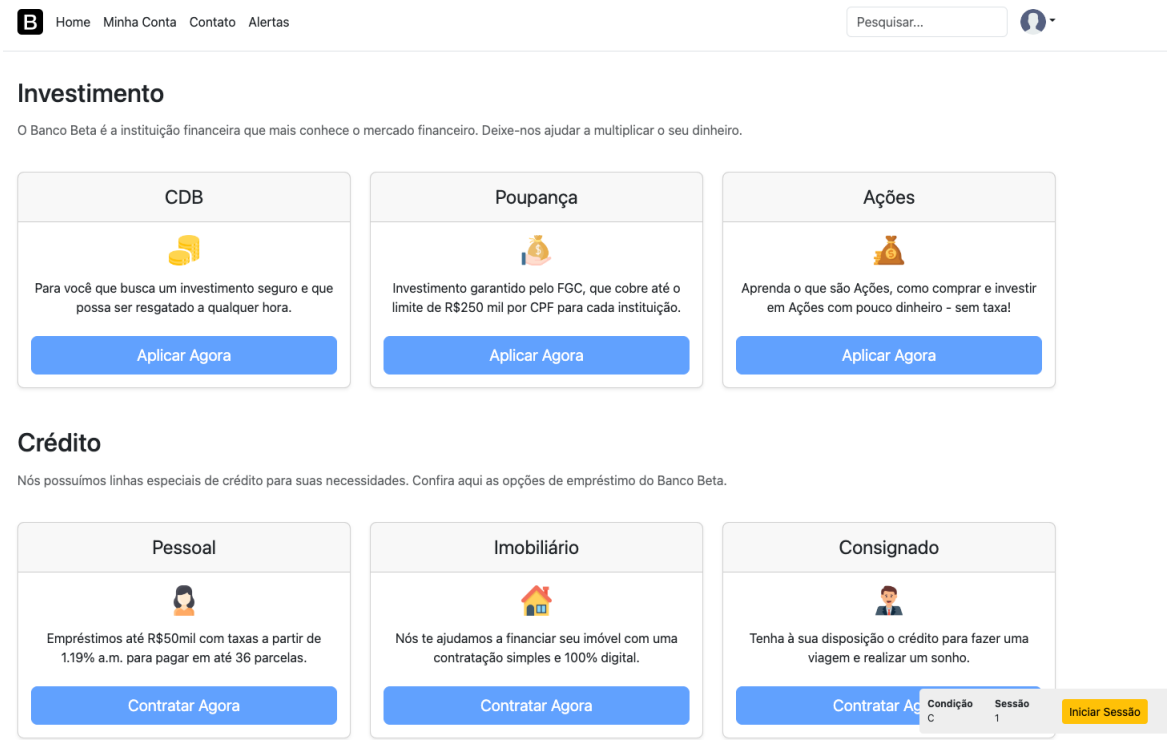

Figura 5.1: IU da parte superior da versão final do protótipo
Fonte: Elaborada pelo autor

Já a Figura 5.2 apresenta a versão final da parte inferior da interface do usuário.

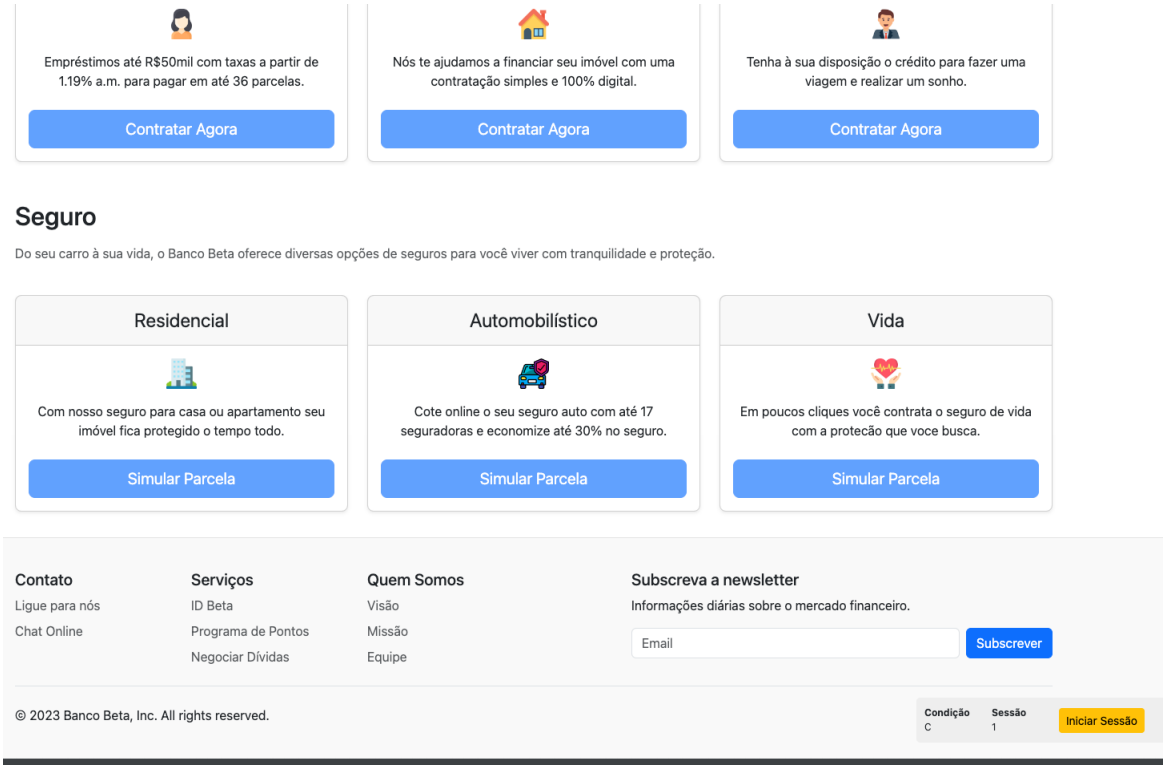

Figura 5.2: IU da parte inferior da versão final do protótipo
Fonte: Elaborada pelo autor

O *backend* e o *frontend* foram implantados no provedor de serviços em nuvem AWS [168]; para gerenciar a infraestrutura, foi utilizada a abordagem *Infra as Code* (IaC), por meio do Kit de desenvolvimento em nuvem (CDK) da AWS. Portanto, a aplicação completa foi implantada com base na tecnologia de *Cloud Computing*. Essa tecnologia é explicada na Seção 4.4.1. Já o CDK é explicado na Seção 5.4.

Os serviços de processamento foram desenvolvidos com a linguagem de programação Javascript, por meio do ambiente de execução Node.js [164]. Somente o motor de adaptação utilizou a linguagem de programação Python [165]. De toda forma, ambos foram implantados no serviço de computação *AWS Lambda* [169].

A comunicação entre o *frontend* e o *backend* foi realizada por *mutations*, *queries* e *subscriptions* GraphQL, como explicado na Seção 5.5. Além disso, as bases de dados não foram acessadas diretamente pelos microsserviços, como previsto na arquitetura, a comunicação entre eles foi providenciada pelo *AWS AppSync* [170], que oferece uma comunicação GraphQL.

5.1 *React*

Para o desenvolvimento do protótipo, este trabalho utilizou a biblioteca para JavaScript chamada *React*, pois é uma das mais populares do mundo, que possui uma vasta e difundida documentação [171]. Dessa forma, a chance de outro autor que queira reproduzir esse trabalho conseguir utilizá-la é alta. Além disso, é recomendada para aplicações que necessite realizar ações simultâneas [172], como no caso da aplicação desenvolvida neste trabalho.

Esta biblioteca é amplamente reconhecida para a construção de IU. Introduzida em 2011, pela equipe do Facebook, surgiu com o propósito de aprimorar a atualização e a sincronização de atividades simultâneas no feed de notícias da rede social. Isso incluía elementos como chat, status, listagem de contatos, entre outros. Inicialmente, essas atividades e requisições de estados eram descritas de maneira complexa. No entanto, o *React* trouxe simplicidade, tanto na descrição desses estados quanto na conexão entre HTML, Folhas de estilo em cascata (CSS) e JavaScript, além de simplificar o gerenciamento de todos os componentes de uma página [172].

Devido à sua eficiência comprovada, o *React* foi incorporado à interface de outras redes sociais do grupo nos anos seguintes, como o Instagram. Em 2013, o código do *React* foi disponibilizado à comunidade, o que impulsionou sua popularização. A biblioteca pretende simplificar a vida dos desenvolvedores, ao criar interfaces de usuário e oferecer diversas funcionalidades que aumentam a produtividade e auxiliam no processo de desenvolvimento. A biblioteca divide a tela em vários componentes, o que permite desenvolvê-los de forma individualizada. Esses componentes são utilizados para reutilização de código e padronização da interface, a fim de proporcionar uma abordagem modular e escalável [173].

5.2 *Bootstrap*

Para o desenvolvimento dos elementos gráficos e visuais do protótipo, este trabalhou utilizou o *Bootstrap*, pois é a biblioteca HTML, CSS e JavaScript mais popular para desenvolver um site responsivo e compatível com múltiplos dispositivos [174]. Além disso, é uma estrutura de desenvolvimento *frontend*, gratuita e de código aberto.

Projetado com o intuito de viabilizar o desenvolvimento responsivo de sites orientados para dispositivos móveis, o *Bootstrap* oferece uma coleção abrangente de sintaxes para a criação de designs de template. Como *framework*, o *Bootstrap* incorpora os elementos básicos necessários para o desenvolvimento web responsivo, e permite que os desenvolve-

dores utilizem uma grade pré-definida para inserir o código. Essa estrutura é construída sobre as bases da linguagem de marcação HTML, das CSS e do JavaScript [175].

A utilização do *Bootstrap* pelos desenvolvedores web proporciona significativa aceleração no processo de criação de sites e elimina a necessidade de se preocuparem com comandos e funções básicas. O *Bootstrap* é responsável por concretizar o design web responsivo; tornar possível a uma página web ou aplicativo identificar, automaticamente, o tamanho e a orientação da tela do visitante; e adaptar a exibição de forma adequada. A abordagem *mobile-first* adotada pressupõe que smartphones, tablets e aplicativos móveis dedicados a tarefas específicas são as principais ferramentas dos usuários para realizar suas atividades. O *Bootstrap* atende a esses requisitos no *design* e fornece componentes de interface de usuário, layouts, ferramentas JavaScript e a estrutura de implementação necessária. O software está disponível em versões pré-compiladas ou como código-fonte, de modo a proporcionar flexibilidade aos desenvolvedores [176].

5.3 *Cloud Computing*

A computação em nuvem (*Cloud Computing*) é um modelo de fornecimento de recursos de tecnologia da informação, sob demanda, por meio da internet, com preços pré-estabelecidos [177]. Em vez de adquirir, possuir e manter infraestruturas físicas, como *data centers* e servidores, é possível acessar serviços tecnológicos, como capacidade de processamento, armazenamento e bancos de dados, conforme necessário, por um provedor de serviços em nuvem, como a *Amazon Web Services* (AWS).

Organizações de diversos segmentos, tamanhos e indústrias adotam a computação em nuvem para uma ampla gama de casos de uso, que incluem: backup de dados, recuperação de desastres, e-mail, desktops virtuais, desenvolvimento e teste de software, análise de big data e aplicativos web voltados para o cliente. Por exemplo, empresas do setor de saúde utilizam a nuvem para desenvolver tratamentos mais personalizados aos pacientes. Instituições financeiras aproveitam a nuvem para fortalecer a detecção e prevenção de fraudes em tempo real. Já os desenvolvedores de jogos eletrônicos utilizam a nuvem para disponibilizar jogos on-line a milhões de jogadores ao redor do mundo [177].

Ao adotar a computação em nuvem, as organizações se beneficiam com maior agilidade, escalabilidade e flexibilidade, visto que podem dimensionar seus recursos de forma rápida e eficiente, conforme as necessidades do negócio. Além disso, a nuvem permite o acesso a tecnologias de ponta e recursos avançados, sem a necessidade de investimentos significativos em infraestrutura própria. Com isso, as empresas podem focar em suas atividades principais, impulsionar a inovação e obter vantagens competitivas no mercado atual, cada vez mais dinâmico e digitalizado [178].

Diante das vantagens citadas, o protótipo foi implantado em um ambiente na nuvem, pois o custo de adquirir um servidor, obter softwares de gerenciamento de hardware e instalar as aplicações necessárias invisibilizaria a realização deste trabalho. Não houve custo para os serviços utilizados no protótipo, pois a utilização esteve incluída na faixa grátis oferecida pela AWS.

5.4 AWS Cloud Development Kit (CDK)

O AWS *Cloud Development Kit* (CDK) é uma estrutura de código aberto que possibilita a modelagem e provisionamento de recursos da nuvem AWS, com a linguagem de programação de preferência. Com suporte para TypeScript, Python, Java ou .NET, o CDK permite que seja modelada a infraestrutura de aplicativos, de forma eficiente. Essa estrutura utiliza o AWS CloudFormation para provisionar recursos de maneira segura e repetível [179].

O CDK foi utilizado para o desenvolvimento do protótipo, pois é uma das maneiras mais populares para gerenciar infraestruturas de forma ágil e programática [180], além de apresentar as seguintes vantagens:

- integração simplificada com a nuvem: permite aproveitar as habilidades e ferramentas existentes para criar infraestruturas na nuvem. Os desenvolvedores podem utilizar a linguagem de programação de sua escolha e continuar usando seus Ambientes de Desenvolvimento Integrados (IDE) preferidos para escrever aplicativos CDK;
- processo de desenvolvimento ágil: a expressividade das linguagens e recursos de programação, como objetos, laços e condicionais, pode acelerar, significativamente, o processo de desenvolvimento. Além disso, é possível escrever casos de teste de unidade para os componentes de infraestrutura. A capacidade de testar o código da infraestrutura, em nível de unidade, é de grande valor e fortalece a confiança do desenvolvedor, a cada alteração realizada;
- personalizável e compartilhável: permite a extensão dos componentes existentes para criar componentes personalizados, de modo a atender aos requisitos de segurança, conformidade e governança da organização. Esses componentes podem ser facilmente compartilhados na organização, o que favorece iniciar, rapidamente, novos projetos com as melhores práticas já estabelecidas;
- continuidade no fluxo de trabalho: é possível escrever o código de execução e definir os recursos da AWS, com a mesma linguagem de programação, além de ser possível

continuar com o mesmo IDE, tanto para o código de execução quanto para o desenvolvimento da infraestrutura. Adicionalmente, é possível visualizar suas pilhas e recursos de aplicativos CDK.

5.5 GraphQL

GraphQL é uma linguagem de consulta e manipulação de dados de código aberto, projetada para APIs, de modo a fornecer um tempo de execução eficiente para responder a consultas que envolvem conjuntos de dados existentes. Entre os recursos fundamentais da tecnologia, destacam-se as *queries*, as *mutations* e as *subscriptions* [181].

5.5.1 *Queries*

Queries são o meio pelo qual os clientes podem requisitar os dados necessários do servidor. As consultas GraphQL desempenham um papel fundamental e estabelecem uma diferença significativa entre GraphQL e REST. Ao contrário do REST, que possui uma estrutura pré-definida para os dados retornados de um *endpoint* específico, o GraphQL expõe apenas um *endpoint* para todas as requisições, de modo a permitir que o cliente solicite, especificamente, o que precisa. Essa abordagem flexível e granular do GraphQL oferece aos clientes um controle mais preciso sobre os dados solicitados.

O esquema de *queries* utilizado neste protótipo foi:

```
type Query {
   getStructure(userId: String!, pageId: String!): Structure!
   getTest(testId: String!): Test!
}
```

As definições das *queries* apresentadas no código são:

- *getStructure*: recupera no banco de dados a estrutura de *frontend* para determinado usuário;
- *getTest*: recupera no banco de dados o teste criado e iniciado para determinado usuário.

Um exemplo de requisição do tipo *query*, que contempla o esquema *getStructure* pode ser observado na Figura 5.3.

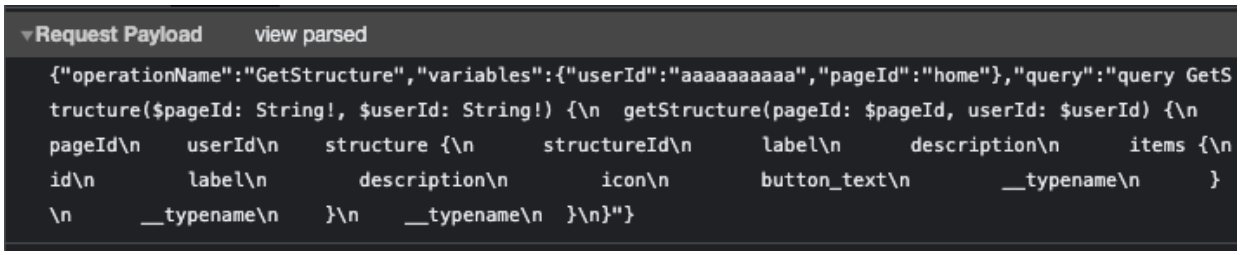

Figura 5.3: Exemplo de requisição do tipo *query*, que contempla o esquema *getStructure*
Fonte: Elaborada pelo autor

5.5.2 *Mutations*

As *mutations* desempenham um papel essencial, ao permitir a modificação dos dados no servidor. Embora apresentem semelhanças, em sua estrutura sintática, com as *queries*, as *mutations* são distintas, ao incorporarem a palavra-chave "mutação". Por meio das *mutations* GraphQL, é possível realizar operações de criação, atualização e exclusão de dados no servidor, o que permite uma capacidade abrangente de manipulação dos recursos disponíveis.

O esquema de *mutations* utilizado neste protótipo foi:

```
type Mutation {
   sendEvent(event: SendEventInput!): Event!
   updateStructure(structure: UpdateStructureInput!): Structure!
   createUser(user: CreateUserInput!): User!
   createTest(test: CreateTestInput!): Test!
   endTest(testId: String!): Test!
   startTest(testId: String!): Test!
   createSession(testId: String!, conditionId: String!, sessionId: Int!): Test!
   endSession(testId: String!, conditionId: String!, sessionId: Int!): Test!
}
```

As definições das *mutations* apresentadas no código são:

- *sendEvent*: o *frontend* envia um evento do tipo clique para o *backend*, e este, no que lhe concerne, salva essa informação no banco de dados;
- *updateStructure*: salva no banco de dados uma nova estrutura de *frontend* para determinado usuário;
- *createUser*: salva no banco de dados um novo usuário;
- *createTest*: salva no banco de dados uma novo teste;

- *endTest*: salva no banco de dados o horário em que um teste foi finalizado;
- *startTest*: salva no banco de dados o horário em que um teste foi iniciado;
- *startTest*: salva no banco de dados o horário em que uma nova sessão de teste foi iniciada;
- *startTest*: salva no banco de dados o horário em que uma sessão de teste foi finalizada.

Um exemplo de requisição do tipo *mutation*, que contempla o esquema *sendEvent*, pode ser observado na Figura 5.4.

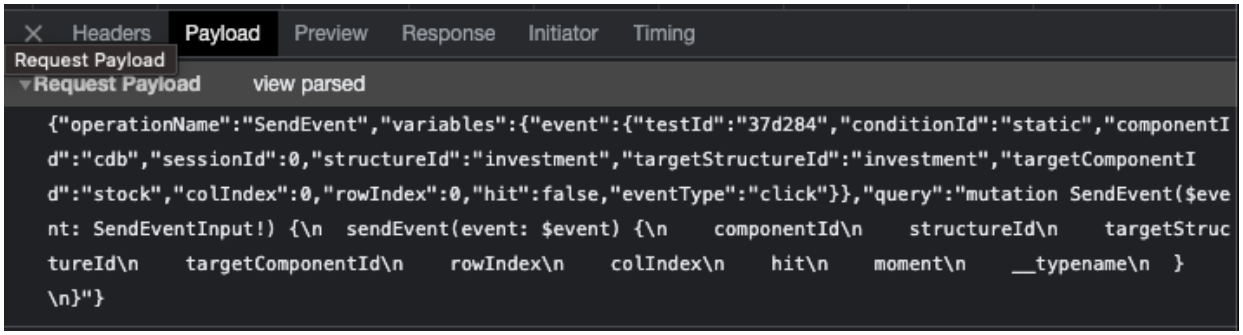

Figura 5.4: Exemplo de requisição do tipo *mutation*, que contempla o esquema *sendEvent*
Fonte: Elaborada pelo autor

5.5.3 *Subscriptions*

As *subscriptions* permitem o envio de dados do servidor para os clientes que optam por receber atualizações em tempo real. Essencialmente, as assinaturas do GraphQL permitem estabelecer e manter uma conexão em tempo real com o servidor. Isso possibilita que os clientes se inscrevam em eventos específicos no servidor, de modo que, sempre que ocorrer um evento correspondente, o servidor responda ao cliente com os dados relevantes em tempo real. Assim como as *queries*, as assinaturas especificam um conjunto de campos a serem entregues ao cliente, quando um evento ocorre no servidor.

Um exemplo de requisição do tipo *subscription*, que contempla o esquema *onUpdateStructure*, pode ser observado na Figura 5.5.

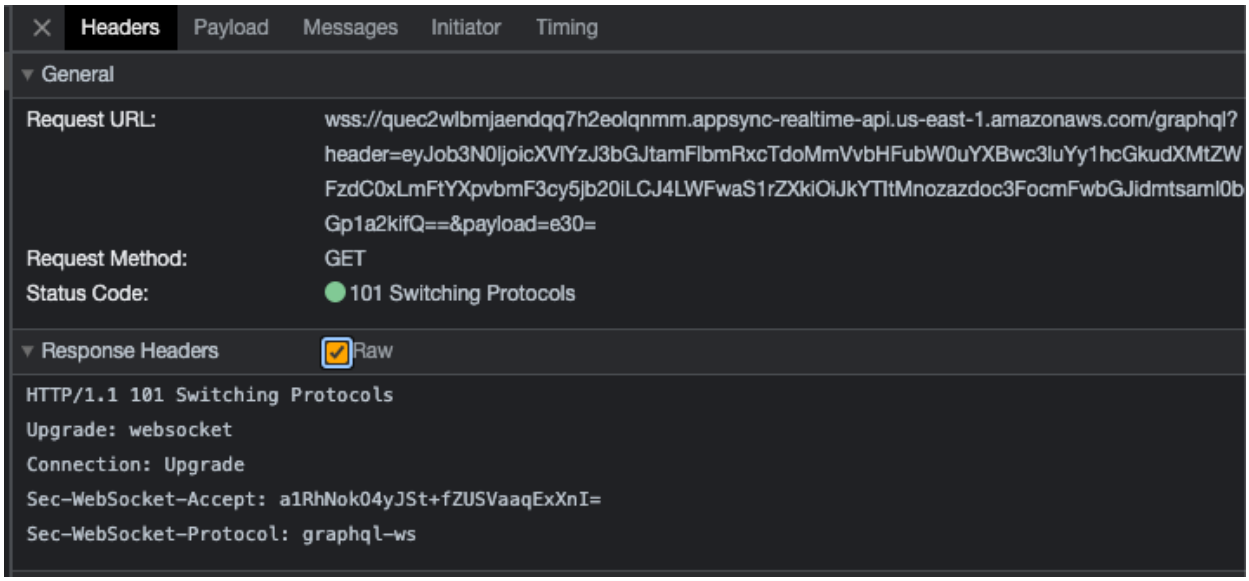

Figura 5.5: Exemplo de requisição do tipo *subscription*, que contempla o esquema *onUpdateStructure*

Fonte: Elaborada pelo autor

A *subscription onUpdateStructure* foi utilizada pelo *frontend* para reagir a eventos de mudanças de estrutura enviados pelo *backend*, isto é, após o processamento dos dados do usuário, caso haja alguma mudança na estrutura do *frontend*, o *backend* envia um evento ao *frontend*, e este, no que lhe concerne, utiliza essa *subscription* para estes eventos. O esquema de *subscriptions* utilizado neste protótipo foi:

```
type Subscription {
   onUpdateStructure(topic: ID): Structure
      @aws_subscribe(mutations: ["updateStructure"])
}
```

Capítulo 6

Validação Empírica

O principal objetivo da avaliação consiste em testar a viabilidade da abordagem apresentada por este projeto. O processamento de adaptações deve ocorrer em tempo real, de modo a organizar os *micro-frontends* dos aplicativos, sem causar impactos negativos para o usuário. Além disso, é essencial que essa composição e reordenação não sobrecarreguem o navegador e transmitam a percepção de que a aplicação não responde às requisições, de forma adequada, em um tempo aceitável. A satisfação do usuário também pode ser prejudicada, caso o *backend* não processe as requisições ou gaste um tempo inaceitável para concluí-las, portanto, é fundamental validar também o comportamento do *backend*.

Por fim, a abordagem de planejamento da MCTS aplicada à grade de aplicativos demonstra melhorias de desempenho, em comparação com as grades estáticas e com a conhecida abordagem adaptativa baseada em frequência. No MCTS, as aplicações se adaptam, após cada bloco por planejamento de adaptações. Na abordagem estática, as aplicações não se adaptam temporalmente. Por último, na abordagem baseada em frequência, as adaptações são realizadas com base na frequência de cliques nos serviços bancários. Para esse propósito, foi conduzida uma validação empírica a qual os participantes concluíram tarefas de seleção de serviços, conforme as seguintes condições: estática, frequência e MCTS. A validação ocorreu de forma verbal e de caráter pontual, realizada por meio de metodologia específica, a qual o participante foi convidado a expressar sua preferência, avaliação ou o sentido que atribui a aplicação desenvolvida neste trabalho. As respostas dos participantes não foram vinculadas com sua identificação, portanto, não há a possibilidade de identificação dos mesmos.

6.1 Materiais

Para a condução do experimento, a ordenação dos aplicativos e dos componentes dos aplicativos foram geradas de forma randômica, por meio do algoritmo *xorshift128+* [182].

No total, foram considerados três aplicativos, com três serviços em cada aplicativo. Os nomes dos serviços foram selecionados a partir de uma observação da página inicial de cinco grandes bancos que atuam no Brasil. Para cada participante, foi criada uma grade de aplicações para cada uma das três condições, em um total de três composições exclusivas.

Dentro de cada condição, utilizou-se também o algoritmo *xorshift128+* para controlar a distribuição de frequência dos serviços, que deveriam ser clicados pelos usuários como estímulo durante as tentativas. A mesma abordagem foi aplicada em todas as três condições para cada participante. Para considerar a variação dos interesses dos usuários, foram geradas grades de serviços exclusivas para cada participante.

O algoritmo *xorshift128+* é um gerador de números pseudoaleatórios. Ele opera em uma estrutura de 128 bits inicializada com uma semente (*seed*) para produzir sequências aparentemente aleatórias de números de 64 bits [182].

6.2 Participantes

O teste contou com a participação de 31 participantes; destes, 58.1% se declararam do gênero feminino e 41.9% se declararam do gênero masculino. Em relação à idade, foi possível notar que a maioria dos participantes se encontra na faixa etária de 28 anos (19,4%), tendo diferentes níveis de escolaridade e foram recrutados de maneira ad hoc. A Figura 6.1 (A) mostra que 96.8% dos participantes acessam o banco por meio da internet.

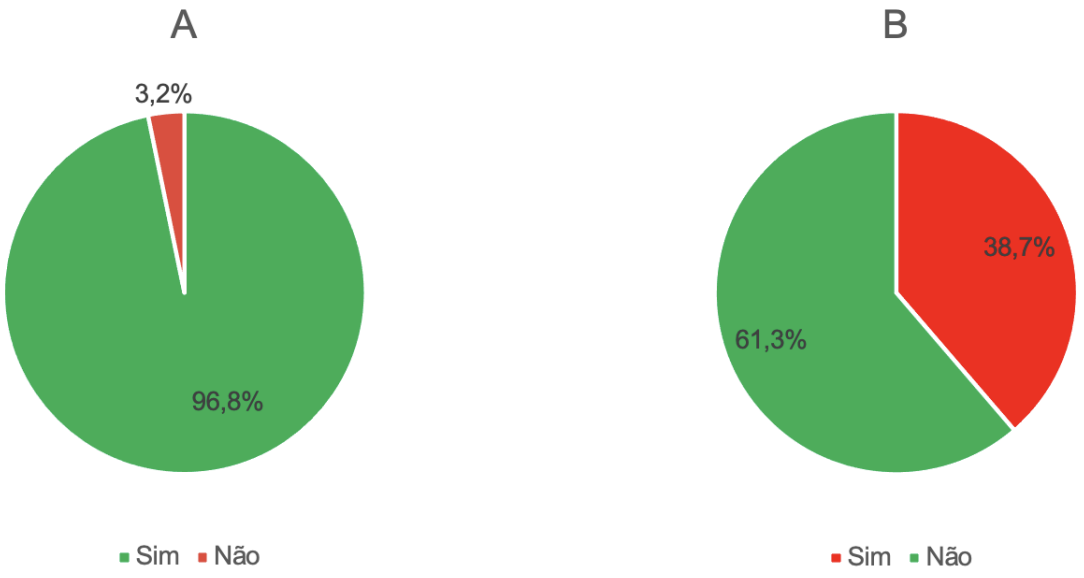

Figura 6.1: Distribuição dos participantes que acessam aplicações financeiras pela internet (A) e os que acessam aplicações financeiras pelo computador (B)

Fonte: Elaborada pelo autor

Por fim, a Figura 6.1 (B) mostra que 61.3% dos participantes não acessam aplicações financeiras por um computador e 38.7% acessam. Essa pergunta foi realizada aos participantes, pois este trabalho tem seu foco em aplicações web acessadas via navegador de computador, por isso é necessário checar o perfil dos participantes. Como pode ser visto, 61.3% dos participantes não acessam aplicações financeiras por um computador, por isso há uma recomendação para trabalhos futuros ampliarem o teste realizado neste trabalho.

6.3 Aparatos

O experimento foi conduzido em um dispositivo Dell Inspiron N4050, com sistema operacional Linux Ubuntu e com uma tela de 14 polegadas (35,56 centímetros). Para as tarefas de seleção, foi utilizado um mouse MasterKeys Lite L, configurado com a velocidade de rastreamento padrão. A IU do estudo foi desenvolvida com HTML e JavaScript e apresentada em uma janela do navegador. Todos os cliques do mouse, que incluem data e hora foram registrados e devidamente documentados.

6.4 Roteiro

O experimento começou com um vídeo introdutório, que explicou como seria o experimento e quais eram as opções que os usuários poderiam selecionar. Após assistir ao vídeo, participantes que consentiram em prosseguir, primeiramente, responderam a um questionário inicial e começaram o teste, o Anexo I apresenta a lista ordenada contendo as atividades realizadas pelos participantes. O formulário inicial continha as seguintes perguntas:

1. Qual o seu nome?
2. Qual a sua idade?
3. Qual o seu gênero?
4. Qual o seu nível de escolaridade?
5. Você acessa serviços bancários por meio da internet?
6. Você acessa banco por meio de um computador?

A primeira tela continha um botão com a frase "Iniciar Teste", conforme apresentada na Figura 6.2.

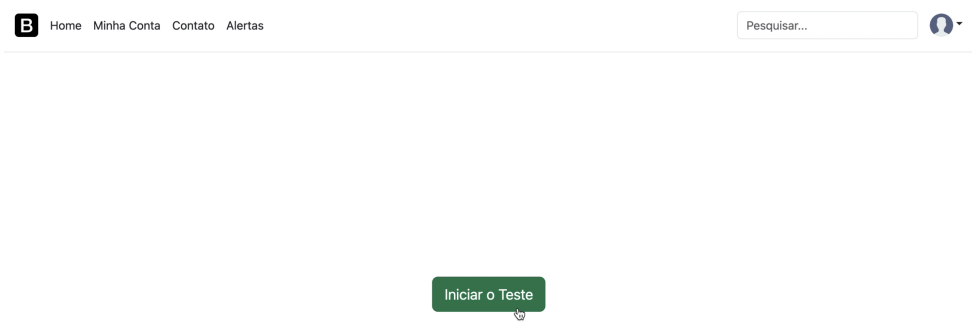

Figura 6.2: *Frontend* como botão "Iniciar Teste"
Fonte: Elaborada pelo autor

Após clicar no botão "Iniciar Teste", o teste se iniciava, conforme apresentado na Figura 6.3.

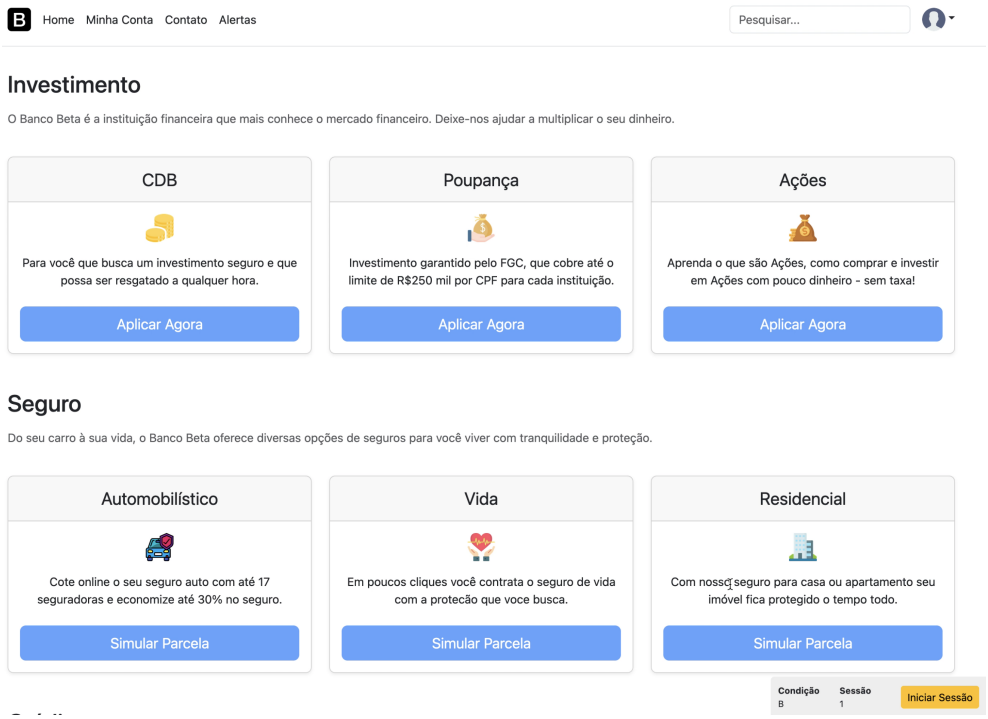

Figura 6.3: *Frontend* da página inicial de um teste
Fonte: Elaborada pelo autor

Cada participante testou as três condições (Estática, Frequência, MCTS), sequencialmente. A ordem das condições foi contrabalanceada entre os participantes, por meio do algoritmo *xorshift128+*. Durante cada condição, o participante interagia com uma grade de serviços bancários, durante três sessões, conforme apresentado na Figura 6.3.

Dentro de uma sessão, os serviços a serem clicados apareciam no canto inferior direito, até que a sessão se encerrasse. Uma mensagem de confirmação aparecia no canto superior direito da tela, quando o usuário clicava em um serviço. Caso o item clicado fosse o item solicitado, então, a mensagem "Parabéns, você acertou!"era mostrada; caso contrário, aparecia: "Você errou, tente novamente!".

A mensagem de erro ou acerto tem como objetivo indicar se o usuário foi capaz ou não de selecionar o serviço bancário indicado pelo teste. A ação é uma simulação do interesse do usuário em determinado serviço bancário, utilizada para medir o tempo de seleção dos componentes na tela. Esse tempo é utilizado no presente trabalho para identificar se o usuário encontrou o componente mais rapidamente (menos tempo entre uma seleção e outra), ou menos rapidamente (mais tempo entre uma seleção e outra). A Figura 6.4 apresenta esse comportamento.

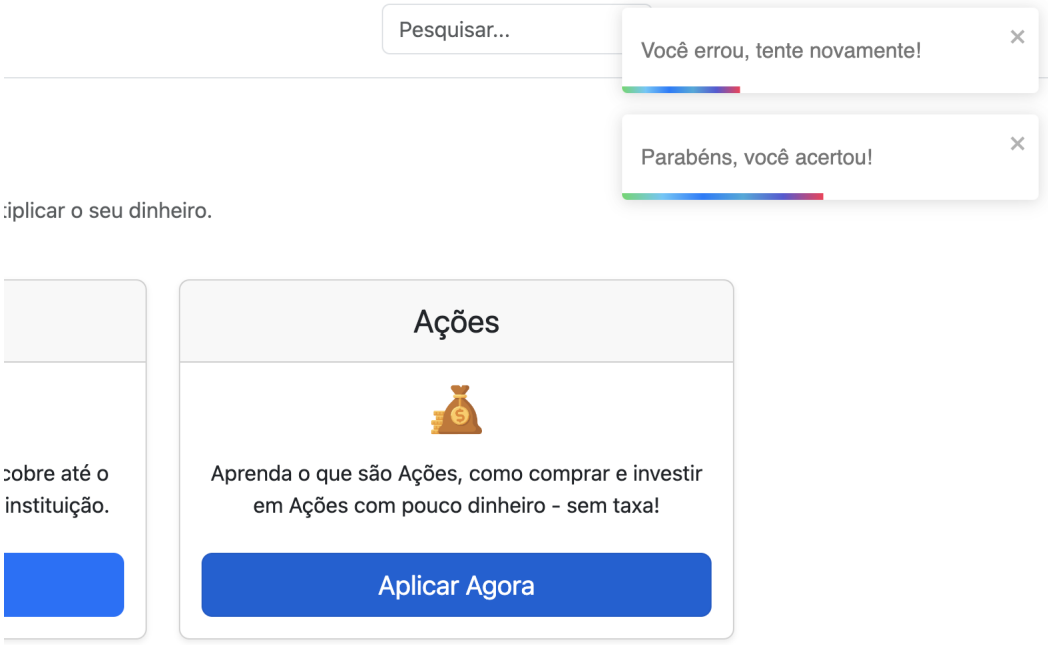

Figura 6.4: Mensagem de confirmação mostrada a cada clique do usuário
Fonte: Elaborada pelo autor

Para cada menu, 20 tarefas de seleção (tentativas) foram concluídas. Essa abordagem foi realizada para refletir as várias sessões que os usuários executam durante uma navegação cotidiana. Os participantes fizeram intervalos obrigatórios de 45 segundos, entre duas sessões consecutivas. Esse é o tempo máximo que o *backend* necessita para processar os dados do usuário e indicar ou não uma alteração na estrutura do *frontend*. Além disso, durante o período, o participante respondeu a um conjunto de perguntas produzidas para serem realizadas entre uma sessão e outra. O questionário é composto pelas seguintes perguntas:

1. Você notou alguma alteração na tela durante o uso?
2. Se sim, você julga que melhorou a utilização?
3. Por que melhorou ou piorou?

Após as sessões de todas as condições serem finalizadas, um botão com a frase "Encerrar Teste"foi mostrado para o usuário. Por fim, o usuário respondeu ao último conjunto de perguntas:

1. De 1 a 5, qual nota você atribui para a aparência do *frontend*? Cor, imagens, tamanho.
2. De 1 a 5, qual nota você atribui para a facilidade de uso do sistema?
3. De 1 a 5, qual nota você atribui para o tempo de resposta entre os cliques?
4. De 1 a 5, qual nota você atribui para a disposição dos elementos? Isto é, organização dos elementos no *frontend*.
5. De 1 a 5, qual nota você atribui para a sua satisfação em utilizar esse site, que representa a página inicial de um sistema bancário?
6. Use este espaço para apresentar os comentários acerca do sistema.

Foi explicado aos participantes a seguinte correspondência entre números e categoria: 1: Péssimo, 2: Ruim, 3: Indiferente, 4: Bom, 5: Ótimo.

Em resumo, o projeto contou com 31 participantes; com 3 condições; em 3 sessões, com 20 seleções, que totalizam 5.580 tentativas. O teste empírico teve uma duração média de 23 minutos e 33 segundos por participante. Cada condição teve uma duração média de 7,77 minutos e cada sessão teve uma duração média de 2,59 minutos. Não houve um critério para a seleção dos participantes, ou seja, foram selecionados de maneira ad hoc durante uma semana, logo após a finalização do desenvolvimento do protótipo. A

oportunidade ocorria quando o autor deste trabalho se encontrava em ambientes em que havia internet e algum cômodo ou mesa isolada das demais, como, por exemplo, sala de escritório. Dessa forma, o provável participante era introduzido ao tema, como descrito no primeiro parágrafo desta Seção 6.4. Nenhum participante selecionado se negou a realizar o teste, de maneira tal que 31 pessoas foram abordadas e todas decidiram participar do experimento.

Para efeito de identificação, os participantes receberam, de maneira aleatória, uma numeração sequencial (1-31). Com isso, foi possível realizar a citação dos participantes da seguinte forma: Participante 1, Participante 2 etc.

6.5 Resultados Quantitativos

Nesta seção, será apresentada a classificação do método científico que utiliza técnicas estatísticas para quantificar os resultados obtidos na validação empírica.

Um modelo de efeito misto para análise de variância de medidas repetidas (ANOVA unidirecional), com tempo de seleção (em milissegundos) como variável dependente, condições (estática, frequência, MCTS); como variável independente fixa; e ID do participante e a estrutura da página como variáveis aleatórias.

A condição teve um efeito estatisticamente significativo no tempo de seleção: $F = 6{,}50$ e $p < 0{,}05$, com as seguintes médias globais: estática = 3.969 ms, frequência = 3.937 ms e MCTS = 3.866 ms, como pode ser observado na Figura 6.5.

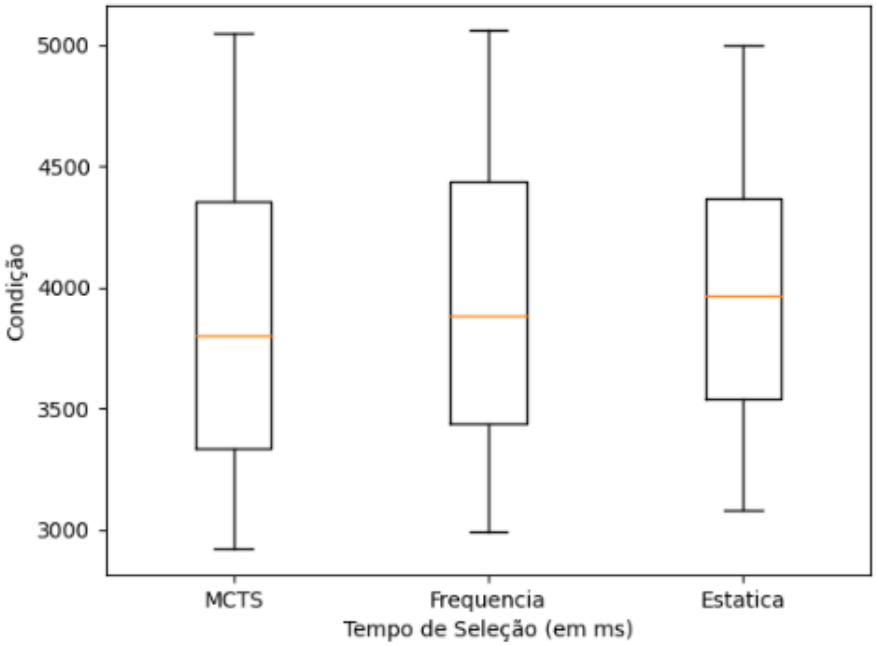

Figura 6.5: Relação entre condição e tempo de seleção
Fonte: Elaborada pelo autor

O teste *post-hoc*, com Tukey HSD, revelou que a MCTS (3.866 ms) foi significativamente mais rápida (menor tempo de seleção) do que a abordagem utilizando frequência (3.937 ms) e estática (3.969 ms); porém, a diferença entre estática e frequência não foi estatisticamente significativa. Esse resultado sugere que, com a abordagem utilizada neste trabalho (MCTS), houve uma melhoria na UX; e o Objetivo Geral, descrito na Seção 1.3.1, foi cumprido, pois o usuário, após as adaptações, foi capaz de selecionar os serviços bancários mais rapidamente.

6.6 Resultados Qualitativos

Durante o estudo, os participantes não foram informados com antecedência, a respeito das adaptações. Após cada sessão, foi perguntado se notaram alterações na página durante o uso e suas opiniões acerca de tais alterações. Diante disso, 61,3% dos participantes comentaram que perceberam mudanças na condição de frequência, mas apenas 39% perceberam como essas mudanças estavam ocorrendo.

Quatro participantes (Participante 9, Participante 6, Participante 22 e Participante 25) comentaram que o reordenamento era confuso e isso os impediram de lembrar a localização dos itens. O Participante 19 comentou: “[...] como eu estava acostumado com a disposição dos elementos, conseguia selecionar os itens de maneira automática, pois já

estava na memória. Então, quando houve mudanças, eu fui forçado a buscar e a memorizar os elementos novamente, tornando a navegação mais confusa". Essa observação indica que alguns usuários não são favoráveis a adaptações, pois preferem utilizar a memória visual para encontrar os itens de que precisam.

Na condição estática, os participantes podiam utilizar a memória para acessarem, diretamente, alguns itens; 6% dos participantes disseram que perceberam mudanças nessa condição, o que sugere a ausência de viés. Na condição MCTS, 80% dos participantes perceberam que houve adaptação na página, e 66% disseram que gostaram das mudanças efetuadas. O Participante 25 comentou: "Os itens mais clicados estão mais perto um do outro, diminuindo o tempo de busca".

Em relação às perguntas finais, para as quais os participantes deveriam dar uma nota de um a cinco (em que 1 é Péssimo; 2, Ruim; 3, Indiferente; 4, Bom; e 5, Ótimo), foi solicitado avaliar a aparência do *frontend*: cor, imagens, tamanho. Nesta situação, 54,8% dos participantes avaliaram como 5 (Ótimo); 29%, como 4 (Bom); e 16,1%, deram a nota de 3 (Indiferente), como mostra a Figura 6.6 (A).

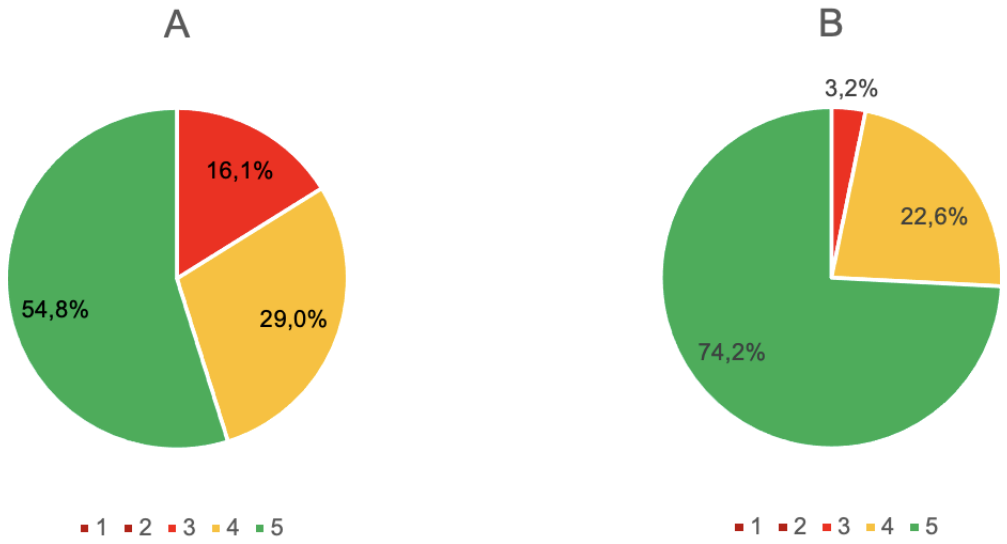

Figura 6.6: Avaliação relação à aparência do *frontend* (A) e em relação ao tempo de resposta entre os cliques (B)

Fonte: Elaborada pelo autor

A mesma pergunta foi realizada para os participantes em relação ao tempo de resposta entre os cliques; 74,2% avaliaram como 5 (Ótimo), e 22,6%, como 4 (Bom), conforme apresenta a Figura 6.6 (B).

Também foi realizada uma pergunta aberta para os usuários, sendo que 12 usuários responderam essa pergunta, e, de maneira geral, foram respostas curtas. A seguir, estão descritas algumas respostas relevantes.

1. Participante 1: "Boa logística, quanto mais uso mais fica fácil, pois o site está entendendo o que estou fazendo e acompanhando meu gosto".

2. Participante 17: "A mudança dos elementos causa confusão na mente".

3. Participante 31: "Conforme a quantidade de cliques nos elementos, mais eles apareciam próximos para mim, facilitando o manuseio".

Os resultados apresentados na Figura 6.6 (A) e na Figura 6.6 (B) sugerem que os participantes avaliaram de forma ótima ou boa a aparência geral do *frontend*; e também quanto ao tempo de resposta aos cliques. Esse resultado indica que a implementação gerou a sensação desejada por este trabalho nos participantes, isto é, pressupõe-se que as decisões de *design* tomadas foram adequadas e atingiram o público como esperado. Além disso, os resultados quantitativos revelaram que a abordagem MCTS (3.866 ms) foi significativamente mais rápida que a abordagem de frequência (3.937 ms) e a abordagem estática (3.969 ms). O conjunto de resultados sugere que o Objetivo Geral, apresentado na Seção 1.3.1, foi cumprido, pois, a experiência dos usuários foi melhorada a partir de uma implementação baseada na arquitetura proposta. Em relação a comentários similares ao do Participante 17, foi percebido durante o teste que são frutos de impressões pessoais e não estão relacionados à faixa etária, gênero ou nível de escolaridade. Este foi o primeiro teste, outros experimentos podem ser realizados, de modo a reforçar os resultados apresentados.

Capítulo 7

Conclusão

No Brasil, as instituições financeiras oferecem cada vez mais serviços por meio da internet, com milhões de usuários que utilizam esses serviços para realizar operações financeiras no dia a dia. Só em 2020, 66,3% das transações executadas em bancos brasileiros foram realizadas de forma on-line [3], e nove em cada dez contratações de crédito são realizadas em canais digitais. Esses dados mostram que as pessoas estão cada vez mais dependentes dos canais digitais para realizar suas operações financeiras e contam com a disponibilidade desses serviços, de forma ininterrupta. Dessa forma, falhas podem gerar grandes prejuízos financeiros para ambos os lados: para as instituições, que deixam de vender serviços e perdem a confiança dos clientes; e para os usuários, que podem perder oportunidades pontuais ou precisarão pagar juros ao atrasar um pagamento, por exemplo. Além disso, de 2017 a 2019, o investimento em bancos digitais cresceu mais de 100%, enquanto o número de negócios (contratação de serviços bancários) aumentou 28%, o que demonstra a força de crescimento dessas instituições. Assim, a concorrência é cada vez mais acirrada no setor bancário. Por essa razão, é necessária uma arquitetura robusta para lidar com os critérios de qualidade requeridos por esse tipo de sistema.

Este trabalho apresentou uma proposta de arquitetura para aplicações web bancárias, a fim de melhorar a experiência do usuário, por meio de uma interface adaptativa. A arquitetura foi viabilizada com base na utilização de métodos, técnicas e tecnologias que atraem a indústria e a academia, a saber: sistemas reativos, microsserviços, *micro-frontends*, algoritmo de aprendizado de máquina e GraphQL. A arquitetura proposta compreende não somente as camadas mais básicas, mas também a descrição e a integração dos componentes essenciais. Com essa arquitetura, em um curto espaço de tempo, é possível desenvolver aplicações sem passar por fases como pesquisa e teste de ferramentas.

Dessa forma, o presente trabalho descreveu as funcionalidades e as propostas de soluções para a área de arquitetura de software. Como resultado, foi formalizada uma descrição arquitetural com as técnicas específicas de criação de sistemas na forma de um

catálogo de tecnologias e técnicas que podem ser usados para implantar novas aplicações. As características dos sistemas construídos a partir da arquitetura proposta e verificadas no estudo de caso incluem experiência de utilização, alta performance, escalabilidade, modularidade, baixo acoplamento, reusabilidade da arquitetura e uso de melhores práticas. Essas características positivas foram validadas por meio de um estudo empírico.

A arquitetura distribuída de microsserviços que foi utilizada neste trabalho traz uma série de vantagens para aplicações bancárias. Em primeiro lugar, a divisão do sistema em microsserviços independentes permite que diferentes equipes trabalhem em módulos separados, o que favorece a escalabilidade e a manutenção. Isso também facilita a adoção de tecnologias específicas para cada serviço, ao permitir atualizações e inovações mais rápidas e eficientes. A abordagem de *micro-frontends*, em conjunto com interfaces adaptativas, também é benéfica para aplicações bancárias, especialmente quando se trata de oferecer uma experiência do usuário mais personalizada e ágil. Ao dividir a interface do usuário em componentes menores e independentes, as equipes de desenvolvimento podem trabalhar de forma paralela e iterativa, o que oportuniza a implementação de novas funcionalidades e melhorias sem comprometer a estabilidade geral da aplicação.

Quanto à comunicação com bases de dados heterogêneas, foi utilizado o *GraphQL*. Essa tecnologia oferece uma camada de abstração que simplifica a interação com diferentes sistemas de armazenamento de dados. Isso é particularmente útil em aplicações bancárias em que há uma variedade de sistemas legados e bases de dados distintas. O *GraphQL* permite que os serviços solicitem apenas os dados necessários para suas operações, evita excesso de busca de dados desnecessários e melhora a eficiência geral do sistema. Além disso, o *GraphQL* oferece uma documentação clara dos tipos de dados disponíveis (esquemas), o que favorece aos desenvolvedores entender e usar as informações fornecidas pelos serviços em suas aplicações.

Para que as adaptações de uma aplicação web seja capaz de melhorar a experiência do usuário, é necessário investigar, treinar e testar diversos algoritmos de ML existentes. Esse passo poderia inviabilizar a execução deste trabalho pelo tempo a ser despendido e alto custo; porém, com a execução da RSL, foi possível realizar buscas em bases científicas sólidas e encontrar o algoritmo que se mostrou eficiente para o caso proposto. Por meio dessa abordagem, foi possível identificar, analisar e comparar os diferentes algoritmos utilizados em contextos similares, bem como suas potencialidades e limitações. Como a análise foi imparcial, isso permitiu que a escolha do algoritmo fosse fundamentada em evidências, o que aumentou as chances de sucesso. Além disso, a RSL evita viés de seleção, otimiza o tempo de pesquisa e viabiliza a identificação de resultados baseados nas últimas tendências e avanços da área. Assim, garante uma abordagem mais informada e inovadora para atender às necessidades específicas dos projetos.

O desenvolvimento do protótipo, de acordo com a arquitetura proposta por este trabalho, apresentou desafios significativos, pois a complexidade da arquitetura distribuída empreendeu uma abordagem cuidadosa no desenho do projeto e na comunicação entre os serviços. Orquestrar as interações entre os microsserviços, garantir a consistência dos dados e lidar com situações de falha requerem uma compreensão profunda de padrões de projeto e boas práticas em arquiteturas distribuídas. Nesse sentido, a utilização do *GraphQL* simplificou o gerenciamento da consistência dos dados; e a implantação com o uso de serviços da nuvem facilitou o controle das situações de falhas. Além disso, a integração harmoniosa entre *micro-frontends* exigiu uma estratégia eficiente para o versionamento e implantação, bem como para o controle de dependências; porém, a utilização do framework *Module Federation* facilitou esse trabalho, pois diminuiu a complexidade de implementação, além de aumentar a performance no carregamento dos *micro-frontends* que compõem a aplicação web.

A adequação a conceitos de sistemas reativos foi um fator fundamental que contribuiu para a definição de uma arquitetura adequada para aplicações bancárias, pois sistemas reativos residem na capacidade de atender às demandas cada vez mais rigorosas e dinâmicas do setor financeiro. Com a crescente quantidade de transações, usuários e requisitos regulatórios, é essencial que as aplicações bancárias sejam altamente responsivas, resilientes e escaláveis. Portanto, os sistemas reativos permitem que essas aplicações lidem com uma grande carga de trabalho e eventos em tempo real, além de garantir que as transações sejam processadas rapidamente e que os clientes obtenham respostas imediatas.

Destarte, as contribuições desta dissertação são:

- formalização da arquitetura para aplicações web bancárias, que foca na experiência do usuário e utiliza técnicas e ferramentas para atingir esse objetivo;
- catálogo de técnicas, tecnologias e ferramentas para o desenvolvimento e implantação de sistemas reativos, com o mapeamento entre as funcionalidades desejadas e a forma de sua implementação prática;
- RSL que captura os algoritmos frequentemente utilizados para o desenvolvimento de aplicações web adaptativas.

O experimento empírico mostrou que a arquitetura alcançou os objetivos estabelecidos, e o método de adaptação demonstrou superar tanto uma política não adaptativa, quanto uma política baseada em frequência para uma página web que oferece diversos serviços bancários. Isso porque os resultados quantitativos revelaram que a abordagem MCTS (3.866 ms) foi significativamente mais rápida que a abordagem de frequência (3.937 ms) e que a abordagem estática (3.969 ms), sugerindo que a experiência dos usuários foi melhorada e que o Objetivo Geral, apresentado na Seção 1.3.1, foi cumprido.

Como perspectivas futuras, a pesquisa pode progredir ao abordar um conjunto mais abrangente de variáveis em seu algoritmo, tais como: sensores externos de Internet das Coisas; posição do cursor; e rastreamento ocular, de modo a oferecer uma adaptação ainda mais personalizada aos usuários. Além disso, seria benéfico considerar dispositivos adicionais, como *tablets*, *smartphones* e *smart TVs*. Outra melhoria seria expandir as opções de adaptação disponíveis na arquitetura, como: cores, tamanho de componentes e tamanho de fonte. Por fim, este trabalho realizou um primeiro teste empírico, em relação à arquitetura proposta, porém este pode ser ampliado, para abranger mais participantes e cenários, assim, os resultados da avaliação podem ser tornar ainda mais confiáveis.

Por fim, a arquitetura proposta por este trabalho pode ser aplicada em instituições financeiras de maneira gradual, com a criação de um plano ao qual resultado em uma substituição total das aplicações legadas para a nova arquitetura, desta forma, é possível diminuir o risco de inconsistência sistêmica. Ao desenvolver uma nova aplicação seguindo esta arquitetura, as aplicações legadas, ao invés de realizar uma lógica internamente ou requisitar outra aplicação legada, requisita a nova aplicação, até uma substituição total. Da mesma forma para a implantação do GraphQL, as aplicações ao contrário de requisitar fontes de dados diretamente, podem ser adaptadas para requisitar o GraphQL e este requisita a fonte de dados, isso deve ocorrer até que todas as aplicações utilizem o GraphQL. Este projeto teve o foco em instituições financeiras, porém, pode ser adaptado para ser utilizado em outras categorias de negócio, principalmente em empresas de grande porte em relação a funcionários e clientes.

Referências

[1] ONU. 2.9 billion people still offline. *New data from ITU suggest 'COVID connectivity boost' – but world's poorest being left far behind*, Nov 2021. URL https://www.itu.int/en/mediacentre/Pages/PR-2021-11-29-FactsFigures.aspx. 1

[2] Keith H. Bennett e Václav T. Rajlich. Software maintenance and evolution: A roadmap. In *Proceedings of the Conference on The Future of Software Engineering*, ICSE '00, page 73–87, New York, NY, USA, 2000. Association for Computing Machinery. ISBN 1581132530. doi: 10.1145/336512.336534. URL https://doi.org/10.1145/336512.336534. 1

[3] Deloitte. Pesquisa febraban de tecnologia bancária 2021, 2021. URL https://cmsarquivos.febraban.org.br/Arquivos/documentos/PDF/pesquisa-febraban-relatorio.pdf. 1, 109

[4] Banco Central do Brasil. Estatísticas do pix, 2022. URL https://www.bcb.gov.br/estabilidadefinanceira/estatisticaspix. 1

[5] BIS. *The dawn of fintech in Latin America: landscape, prospects and challenges*. Number 112 in BIS Papers. Bank for International Settlements, April 2020. ISBN ARRAY(0x588116f8). URL https://ideas.repec.org/b/bis/bisbps/112.html. 1

[6] Febraban. Brasil é maior mercado de fintechs na américa latina, diz estudo, 2020. URL https://noomis.febraban.org.br/blog/brasil-e-maior-mercado-de-fintechs-na-america-latina-diz-estudo. 1

[7] Akamai. Mais que dobra número de usuários de bancos digitais, mas a segurança ainda preocupa, 2021. URL https://cantarinobrasileiro.com.br/blog/mais-que-dobra-numero-de-usuarios-de-bancos-digitais-mas-a-seguranca-ainda-preocupa/. 2

[8] Poder360. Maiores bancos digitais têm 82 milhões de contas em 2021, 2021. URL https://www.poder360.com.br/economia/maiores-bancos-digitais-tem-82-milhoes-de-contas-em-2021/. 2

[9] Laís Pâmela e Márcia D'Souza. Desempenho de bancos digitais brasileiros: um estudo sob o enfoque da análise envoltória de dados (dea). In *18º Congresso USP de Iniciação científica em Contabilidade*, 08 2021. 2

[10] Francisco Liébana-Cabanillas, Juan Sánchez-Fernández, e Francisco Muñoz-Leiva. The moderating effect of experience in the adoption of mobile payment tools in

virtual social networks: The m-payment acceptance model in virtual social networks (mpam-vsn). *International Journal of Information Management*, 34:151–166, 04 2014. doi: 10.1016/j.ijinfomgt.2013.12.006. 2, 52

[11] Don Norman e Jakob Nielsen. The definition of user experience (ux), 2006. URL `https://www.nngroup.com/articles/definition-user-experience/`. 2, 16

[12] Jaehyun Park, Sung Han, Hyun K Kim, Youngseok Cho, e Wonkyu Park. Developing elements of user experience for mobile phones and services: Survey, interview, and observation approaches. *Human Factors and Ergonomics in Manufacturing & Service Industries*, 23, 07 2013. doi: 10.1002/hfm.20316. 2, 16, 52

[13] Jaehyun Park, Sung Han, Hyun K Kim, Oh Seunghwan, e Heekyung Moon. Modeling user experience: A case study on a mobile device. *International Journal of Industrial Ergonomics*, 43:187–196, 03 2013. doi: 10.1016/j.ergon.2013.01.005. 2, 52

[14] Carine Lallemand, Guillaume Gronier, e Vincent Koenig. User experience: A concept without consensus? exploring practitioners' perspectives through an international survey. *Computers in Human Behavior*, 43:35–48, 02 2015. doi: 10.1016/j.chb.2014.10.048. 2, 16, 52

[15] Jakob Nielsen e Thomas K. Landauer. A mathematical model of the finding of usability problems. In *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems*, CHI '93, page 206–213, New York, NY, USA, 1993. Association for Computing Machinery. ISBN 0897915755. doi: 10.1145/169059.169166. URL `https://doi.org/10.1145/169059.169166`. 2, 15

[16] Ahmed Seffah e Eduard Metzker. The obstacles and myths of usability and software engineering. *Commun. ACM*, 47:71–76, 12 2004. doi: 10.1145/1035134.1035136. 2

[17] M. Fishbein e Icek Ajzen. *Belief, attitude, intention and behaviour: An introduction to theory and research*, volume 27. Addison-Wesley, 05 1975. 2

[18] Pratibha Dabholkar e Xiaojing Sheng. The role of perceived control and gender in consumer reactions to download delays. *Journal of Business Research*, 62:756–760, 07 2009. doi: 10.1016/j.jbusres.2008.06.001. 2

[19] Anol Bhattacherjee. An empirical analysis of the antecedents of electronic commerce service continuance. *Decision Support Systems*, 32(2):201–214, 2001. ISSN 0167-9236. doi: https://doi.org/10.1016/S0167-9236(01)00111-7. URL `https://www.sciencedirect.com/science/article/pii/S0167923601001117`. Decision Support Issues in Customer Relationship Management and Interactive Marketing for E-Commerce. 2

[20] Lina Zhou, Liwei Dai, e Dongsong Zhang. Online shopping acceptance model - a critical survey of consumer factors in online shopping. *Journal of Electronic Commerce Research*, 8, 01 2007. 2

[21] Laura Carvajal, Ana Moreno, Maria-Isabel Sanchez-Segura, e Ahmed Seffah. Usability through software design. *IEEE Transactions on Software Engineering*, 39, 11 2013. doi: 10.1109/TSE.2013.29. 2

[22] Len Bass e Bonnie John. Linking usability to software architecture patterns through general scenarios. *Journal of Systems and Software*, 66:187–197, 06 2003. doi: 10.1016/S0164-1212(02)00076-6. 2, 16, 52

[23] Jonas Bonér, Dave Farley, Roland Kuhn, e Martin Thompson. The reactive manifesto, v2. 0, 2014. URL `https://www.reactivemanifesto.org`. 3, 12, 51, 52, 85, 86

[24] James Lewis e Martin Fowler. Microservices: a definition of this new architectural term, 2014. URL `https://martinfowler.com/articles/microservices.html`. 3, 85, 86

[25] Jonas Bonér. *Reactive Microservices Architecture*. O'Reilly Media, Inc., 1st edition edition, 2016. ISBN 1-4919-7566-0. 3

[26] Caifang Yang, Chuanchang Liu, e Zhiyuan Su. Research and application of micro frontends. *IOP Conference Series: Materials Science and Engineering*, 490:062082, apr 2019. doi: 10.1088/1757-899x/490/6/062082. URL `https://doi.org/10.1088/1757-899x/490/6/062082`. 3, 17

[27] Saeed Raheel. Improving the user experience using an intelligent adaptive user interface in mobile applications. In *2016 IEEE International Multidisciplinary Conference on Engineering Technology (IMCET)*, pages 64–68, 2016. doi: 10.1109/IMCET.2016.7777428. 3

[28] Rui Chen, Shanshan Li, e Zheng (Eddie) Li. From monolith to microservices: A dataflow-driven approach. In *From Monolith to Microservices: A Dataflow-Driven Approach*, pages 466–475, 12 2017. doi: 10.1109/APSEC.2017.53. 4

[29] Chris Richardson. What are microservices?, 2019. URL `http://microservices.io/index.html`. 4

[30] Paolo Francesco, Patricia Lago, e Ivano Malavolta. Migrating towards microservice architectures: An industrial survey. In *Migrating Towards Microservice Architectures: An Industrial Survey*, pages 29–2909, 04 2018. doi: 10.1109/ICSA.2018.00012. 4

[31] B. Hayes-Roth, K. Pfleger, P. Lalanda, P. Morignot, e M. Balabanovic. A domain-specific software architecture for adaptive intelligent systems. *IEEE Transactions on Software Engineering*, 21(4):288–301, 1995. doi: 10.1109/32.385968. 4

[32] Nenad Medvidovic e Richard N. Taylor. Software architecture: foundations, theory, and practice. In *2010 ACM/IEEE 32nd International Conference on Software Engineering*, volume 2, pages 471–472, 2010. doi: 10.1145/1810295.1810435. 4

[33] Neeraj Sangal, Ev Jordan, Vineet Sinha, e Daniel Jackson. Using dependency models to manage complex software architecture. In *Proceedings of the 20th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications*, OOPSLA '05, page 167–176, New York, NY, USA, 2005. Association for Computing Machinery. ISBN 1595930310. doi: 10.1145/1094811.1094824. URL https://doi.org/10.1145/1094811.1094824. 4

[34] Martin Fowler. Technical debt, 2003. URL https://martinfowler.com/bliki/TechnicalDebt.html. 4

[35] Joshua Garcia, Daniel Popescu, George Edwards, e Nenad Medvidovic. Identifying architectural bad smells. In *2009 13th European Conference on Software Maintenance and Reengineering*, pages 255–258, 2009. doi: 10.1109/CSMR.2009.59. 4

[36] Tim Smet. Micro frontend architecture for cross framework reusability in practice. Master's thesis, Universiteit Ghent, Campus Aula, Department of European, Public and International Law Universiteitstraat 4, 9000 Gent, Belgium, 2020. URL https://libstore.ugent.be/fulltxt/RUG01/002/946/106/RUG01-002946106_2021_0001_AC.pdf. 5, 85, 86

[37] Azuel Silva Neto. Desenvolvimento de uma aplicação pwa que comporte outras aplicações usando arquitetura de micro frontend. Master's thesis, PUC Goiás, Av, Universitária 1.440, Setor Universitário, 2020. URL https://repositorio.pucgoias.edu.br/jspui/handle/123456789/1037. 5

[38] Taylor Rodrigues Lopes. Método de migração de sistemas monolíticos legados para a arquitetura de microsserviços. Master's thesis, UnB, Campus Universitário Darcy Ribeiro, Brasilia-DF, 2021. URL https://repositorio.unb.br/handle/10482/41178. 5

[39] Andrey Pavlenko, Nursultan Askarbekuly, Swati Megha, e Manuel Mazzara. Microfrontends: application of microservices to web front-ends. *Journal of Internet Services and Information Security (JISIS)*, 10(2), 05 2020. doi: 10.22667/JISIS.2020.05.31.049. 5

[40] Manel Mena, Antonio Corral, Luis Iribarne, e Javier Criado. A progressive web application based on microservices combining geospatial data and the internet of things. *IEEE Access*, 7:104577–104590, 2019. doi: 10.1109/ACCESS.2019.2932196. 5

[41] Antonio Jesus Fernandez Garcia, Luis Iribarne, Antonio Corral, Javier Criado, e James Wang. A microservice-based architecture for enhancing the user experience in cross-device distributed mashup uis with multiple forms of interaction. *Universal Access in the Information Society*, 18, 11 2019. doi: 10.1007/s10209-017-0606-0. 5

[42] Kashyap Todi, Gilles Bailly, Luis Leiva, e Antti Oulasvirta. Adapting user interfaces with model-based reinforcement learning. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI '21, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450380966. doi:

10.1145/3411764.3445497. URL https://doi.org/10.1145/3411764.3445497. 5, 8, 9, 40, 41, 42, 45, 48, 49, 53, 54, 55, 56, 57, 59, 60, 64, 65, 66, 67

[43] Shweta Bhatt. Reinforcement learning 101, 2018. URL https://towardsdatascience.com/reinforcement-learning-101-e24b50e1d292. 8, 9

[44] Maciej Świechowski, Konrad Godlewski, Bartosz Sawicki, e Jacek Mańdziuk. Monte carlo tree search: A review of recent modifications and applications, 2021. URL https://arxiv.org/abs/2103.04931. 10

[45] Paolo Ciancarini e Gian Piero Favini. Monte carlo tree search in kriegspiel. *Artificial Intelligence*, 174(11):670–684, 2010. ISSN 0004-3702. doi: https://doi.org/10.1016/j.artint.2010.04.017. URL https://www.sciencedirect.com/science/article/pii/S0004370210000536. 10

[46] Rémi Coulom. Efficient selectivity and backup operators in monte-carlo tree search. In H. Jaap van den Herik, Paolo Ciancarini, e H. H. L. M. (Jeroen) Donkers, editors, *Computers and Games*, pages 72–83, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. 11

[47] P. Clements, F. Bachmann, L. Bass, D. Garlan, J. Ivers, R. Little, P. Merson, R. Nord, e J. Stafford. *Documenting Software Architectures: Views and Beyond*. SEI Series in Software Engineering. Pearson Education, 2010. ISBN 9780132488594. URL https://books.google.com.br/books?id=UTZbsrA4qAsC. 11, 84, 85

[48] ISO/IEC/IEEE. Systems and software engineering – architecture description. *ISO/IEC/IEEE 42010:2011(E) (Revision of ISO/IEC 42010:2007 and IEEE Std 1471-2000)*, pages 1 –46, 1 2011. doi: 10.1109/IEEESTD.2011.6129467. 12

[49] B. Tekinerdogan e H. Sozer. Chapter 4 - an architecture viewpoint for modeling dynamically configurable software systems. In Ivan Mistrik, Nour Ali, Rick Kazman, John Grundy, e Bradley Schmerl, editors, *Managing Trade-Offs in Adaptable Software Architectures*, pages 79–97. Morgan Kaufmann, Boston, 2017. ISBN 978-0-12-802855-1. doi: https://doi.org/10.1016/B978-0-12-802855-1.00004-6. URL https://www.sciencedirect.com/science/article/pii/B9780128028551000046. 12

[50] Soufiane Maguerra, Azedine Boulmakoul, Lamia Karim, Badir Hassan, e Ahmed Lbath. Towards a reactive system for managing big trajectory data. *Journal of Ambient Intelligence and Humanized Computing*, 11, 10 2020. doi: 10.1007/s12652-019-01625-3. 13

[51] Nuha Alshuqayran, Nour Ali, e Roger Evans. A systematic mapping study in microservice architecture. In *2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA)*, pages 44–51, 2016. doi: 10.1109/SOCA.2016.15. 13

[52] Giovanni Toffetti, Sandro Brunner, Martin Blöchlinger, Florian Dudouet, e Andrew Edmonds. An architecture for self-managing microservices. In *Proceedings of the 1st International Workshop on Automated Incident Management in*

Cloud, AIMC '15, page 19–24, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450334761. doi: 10.1145/2747470.2747474. URL https://doi.org/10.1145/2747470.2747474. 13

[53] Eric Anderson, Xiaozhou Li, Mehul A. Shah, Joseph Tucek, e Jay J. Wylie. What consistency does your key-value store actually provide? In *Proceedings of the Sixth International Conference on Hot Topics in System Dependability*, HotDep'10, page 1–16, USA, 2010. USENIX Association. 13

[54] Amazon. Amazon key value, 2023. URL https://aws.amazon.com/nosql/key-value/. 14

[55] Redis. Redis key value, 2023. URL https://redis.com/nosql/key-value-databases/. 14

[56] Florian Sarodnick e Henning Brau. *Methoden der Usability Evaluation: Wissenschaftliche Grundlagen und praktische Anwendung*. Huber, Bern, 1 edition, 2006. ISBN 3456842007. 14

[57] B. Shackel. *Man-computer Interaction: Human Factors Aspects of Computers & People*. NATO ASI series: Series E : applied sciences. Springer, 1981. ISBN 9789028609105. URL https://books.google.com.br/books?id=G68mAAAAMAAJ. 14

[58] Nigel Bevan, Jurek Kirakowski, e Jonathan Maissel. What is usability? In *Proceedings of the 4th International Conference on HCI*, 01 1991. 14

[59] Ergonomics of human-system interaction. Ergonomics of human-system interaction — part 210: Human-centred design for interactive systems. Standard ISO 9241-210:2010, International Organization for Standardization, Geneva, CH, 2010. URL https://www.iso.org/standard/52075.html. 15, 16

[60] Noam Tractinsky. The usability construct: A dead end? *Human–Computer Interaction*, 33(2):131–177, 2018. doi: 10.1080/07370024.2017.1298038. URL https://doi.org/10.1080/07370024.2017.1298038. 15

[61] Kasper Hornbæk e Effie Lai-Chong Law. Meta-analysis of correlations among usability measures. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '07, page 617–626, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 9781595935939. doi: 10.1145/1240624.1240722. URL https://doi.org/10.1145/1240624.1240722. 15

[62] Jürgen Sauer, Andreas Sonderegger, e Sven Schmutz. Usability, user experience and accessibility: towards an integrative model. *Ergonomics*, 63:1–23, 05 2020. doi: 10.1080/00140139.2020.1774080. 15

[63] Pieter Desmet e Paul Hekkert. Framework of product experience. *International Journal of Design*, 1:57–66, 04 2007. 15

[64] Nigel Bevan. Classifying and selecting ux and usability measures. *International Workshop on Meaningful Measures: Valid Useful User Experience Measurement*, 01 2008. 16

[65] Gitte Lindgaard e Cathy Dudek. What is this evasive beast we call user satisfaction? *Interacting with Computers*, 15:429–452, 06 2003. doi: 10.1016/S0953-5438(02)00063-2. 16

[66] Invillia Adademy Insights. Agilizando o desenvolvimento de apps modernas com microfrontends, 2022. URL `https://micro-frontends.org`. 17

[67] Micro Frontends. Micro frontends, 2022. URL `https://insights.invillia.com/pt/desenvolvimento-de-apps-modernas-com-microfrontends/`. 17

[68] Hazem M El-Bakry, Alaa M Riad, Mohamed Abu-Elsoud, Samaa Mohamed, Ahmed E Hassan, Mahmoud S Kandel, e Nikos Mastorakis. Adaptive user interface for web applications. In *Recent Advances in Business Administration: Proceedings of the 4th WSEAS International Conference on Business Administration (ICBA'10)*, pages 20–22, 2010. 17, 18

[69] Pat Langley. User modeling in adaptive interface. In Judy Kay, editor, *UM99 User Modeling*, pages 357–370, Vienna, 1999. Springer Vienna. ISBN 978-3-7091-2490-1. 18

[70] P. Langley e M.B. Morgan. *Elements of Machine Learning*. Machine Learning Series. Morgan Kaufmann, 1996. ISBN 9781558603011. URL `https://books.google.com.br/books?id=TNg5qVoqRtUC`. 18

[71] Pramila P. Shinde e Seema Shah. A review of machine learning and deep learning applications. In *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, pages 1–6, 2018. doi: 10.1109/ICCUBEA.2018.8697857. 19

[72] Lei Zhang, Shuai Wang, e Bing Liu. Deep learning for sentiment analysis : A survey, 2018. URL `https://arxiv.org/abs/1801.07883`. 19

[73] Barbara Kitchenham e Stuart Charters. Guidelines for performing systematic literature reviews in software engineering. *Journal of Software Engineering and Applications*, 2, 01 2007. 20, 21, 31

[74] D.J. Cook, Cynthia Mulrow, e bhaynes@mcmaster.ca Haynes. Systematic reviews: Synthesis of best evidence for clinical decisions. *Annals of internal medicine*, 126: 376–80, 04 1997. doi: 10.7326/0003?4819?126?5?199703010?00006. 20

[75] Matthias Egger e G Smith. Meta-analysis. potentials and promise. *BMJ (Clinical research ed.)*, 315:1371–4, 12 1997. doi: 10.1136/bmj.315.7119.1371. 20

[76] Henry Beecher. The powerful placebo. *Journal of the American Medical Association*, 159:1602–6, 12 1955. doi: 10.1001/jama.1955.02960340022006. 20

[77] MW Enkin, MJ Keirse, MJ Renfrew, e JP Neilson. *Effective care in pregnancy and childbirth*. Oxford University Press, 1st edition edition, 1989. 20

[78] Alexander Cordeiro, Glória Oliveira, Juan Renteria, e Carlos Alberto Guimaraes. Revisão sistemática: uma revisão narrativa. *Revista do Colégio Brasileiro de Cirurgiões*, 34, 12 2007. doi: 10.1590/S0100-69912007000600012. 20

[79] Diego DERMEVAL, Jorge A. P. de M. COELHO, e Ig I. BITTENCOURT. Mapeamento sistemático e revisão sistemática da literatura em informática na educação. In *Metodologia de Pesquisa Científica em Informática na Educação: Abordagem Quantitativa*, chapter 3. SBC, 2020. 21, 23

[80] Barbara Kitchenham. Procedures for performing systematic reviews. *Keele, UK, Keele Univ.*, 33, 08 2004. 21, 24, 27, 31

[81] Sômulo Nogueira Mafra e Guilherme Travassos. Estudos pirmários e secundários apoiando a busca por evidênica em engenharia de software. Technical Report 687, COPPE/UFRJ, 03 2006. 21

[82] Jorge Biolchini, P. Gomes Mian, A. Candida Cruz Natali, e G. Horta Travassos. Systematic review in software engineering. *System Engineering and Computer Science Department COPPE/UFRJ, Technical Report ES*, 679(05), 2005. 22

[83] Justus Randolph. A guide to writing the dissertation literature review. *Practical Assessment, Research, & Evaluation*, 14, 01 2007. 22

[84] Lindsay Uman. Systematic reviews and meta-analyses. *Journal of the Canadian Academy of Child and Adolescent Psychiatry = Journal de l'Académie canadienne de psychiatrie de l'enfant et de l'adolescent*, 20:57–9, 02 2011. doi: 10.1002/9781444311723.ch8. 24, 25, 27

[85] Evelina Tacconelli. Crd's guidance for undertaking reviews in health care. *Lancet Infectious Diseases - LANCET INFECT DIS*, 10:226–226, 04 2010. doi: 10.1016/S1473-3099(10)70065-7. 25

[86] Abi Methley, Stephen Campbell, Carolyn Chew-Graham, Rosalind McNally, e Sudeh Cheraghi-Sohi. Pico, picos and spider: A comparison study of specificity and sensitivity in three search tools for qualitative systematic reviews. *BMC health services research*, 14:579, 11 2014. doi: 10.1186/s12913-014-0579-0. 25

[87] Barbara Kitchenham, O. Pearl Brereton, David Budgen, Mark Turner, John Bailey, e Stephen Linkman. Systematic literature reviews in software engineering – a systematic literature review. *Information and Software Technology*, 51(1):7–15, 2009. ISSN 0950-5849. doi: https://doi.org/10.1016/j.infsof.2008.09.009. URL `https://www.sciencedirect.com/science/article/pii/S0950584908001390`. Special Section - Most Cited Articles in 2002 and Regular Research Papers. 25

[88] Joanna Lumsden. *Human-computer interaction and innovation in handheld, mobile and wearable technologies*. IGI Global, United States, April 2011. ISBN 1-60960-499-7. doi: 10.4018/978-1-60960-499-8. This book is only available for purchase. Please follow the link to the publisher. 26

[89] Anders Kofod-Petersen. How to do a structured literature review in computer science. *Ver. 0.1. October*, 1, 2012. 28

[90] Laboratório de Pesquisa em Engenharia de Software. Tool to help students to generate a slr, 2022. URL `http://lapes.dc.ufscar.br/tools/start_tool`. 30, 31, 36

[91] Aline Dresch, Daniel Lacerda, e Junico Antunes. *Design Science Research: Método de Pesquisa para Avanço da Ciência e Tecnologia*. Bookman, 01 2015. ISBN 978-85-8260-298-0. doi: 10.13140/2.1.2264.2885. 31

[92] Maranke Wieringa. What to account for when accounting for algorithms: A systematic literature review on algorithmic accountability. In *What to Account for When Accounting for Algorithms: A Systematic Literature Review on Algorithmic Accountability*, FAT* '20, page 1–18, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450369367. doi: 10.1145/3351095.3372833. URL `https://doi-org.ez54.periodicos.capes.gov.br/10.1145/3351095.3372833`. 32

[93] J. Kong, W.Y. Zhang, N. Yu, e X.J. Xia. Design of human-centric adaptive multimodal interfaces. *International Journal of Human-Computer Studies*, 69(12):854–869, 2011. ISSN 1071-5819. doi: https://doi.org/10.1016/j.ijhcs.2011.07.006. URL `https://www.sciencedirect.com/science/article/pii/S1071581911001017`. 38, 41, 47, 49

[94] Rebai Rim, Abid Aymen, Maalej Mohamed Amin, e Mahfoudhi Adel. Cloud computing for an adaptive web interface. In *2015 5th International Conference on Information & Communication Technology and Accessibility (ICTA)*, pages 1–3, 2015. doi: 10.1109/ICTA.2015.7426933. 38, 42, 46, 49

[95] Pierre A. Akiki, Arosha K. Bandara, e Yijun Yu. Engineering adaptive model-driven user interfaces. *IEEE Transactions on Software Engineering*, 42(12):1118–1147, 2016. doi: 10.1109/TSE.2016.2553035. 38, 41, 43, 44, 47, 48, 49

[96] Harold Soh, Scott Sanner, Madeleine White, e Greg Jamieson. Deep sequential recommendation for personalized adaptive user interfaces. In *Proceedings of the 22nd International Conference on Intelligent User Interfaces*, IUI '17, page 589–593, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450343480. doi: 10.1145/3025171.3025207. URL `https://doi.org/10.1145/3025171.3025207`. 38, 41, 43, 44, 49

[97] Francesca Gullà, Silvia Ceccacci, Roberto Menghi, Lorenzo Cavalieri, e Michele Germani. Adaptive interface for smart home: A new design approach. In *Adaptive Interface for Smart Home: A New Design Approach*, pages 107–115, 04 2017. ISBN 978-3-319-54282-9. doi: 10.1007/978-3-319-54283-6_8. 39, 41, 46, 49

[98] Çağla Çığ Karaman e Tevfik Metin Sezgin. Gaze-based predictive user interfaces: Visualizing user intentions in the presence of uncertainty. *International Journal of Human-Computer Studies*, 111:78–91, 2018. ISSN 1071-5819. doi: https://doi.org/10.1016/j.ijhcs.2017.11.005. URL `https://www.sciencedirect.com/science/article/pii/S1071581917301611`. 39, 42, 44, 47, 48, 49

[99] Antonio Jesus Fernandez Garcia, Luis Iribarne, Antonio Corral, Javier Criado, e James Wang. A microservice-based architecture for enhancing the user experience in cross-device distributed mashup uis with multiple forms of interaction. *Universal Access in the Information Society*, 18, 11 2019. doi: 10.1007/s10209-017-0606-0. 39, 41, 44, 47

[100] Nilanka Rathnayake, Dulani Meedeniya, Indika Perera, e Anuradha Welivita. A framework for adaptive user interface generation based on user behavioural patterns. In *2019 Moratuwa Engineering Research Conference (MERCon)*, pages 698–703, 2019. doi: 10.1109/MERCon.2019.8818825. 40, 41, 43, 45, 48, 49

[101] Kashyap Todi, Jussi Jokinen, Kris Luyten, e Antti Oulasvirta. Individualising graphical layouts with predictive visual search models. *ACM Trans. Interact. Intell. Syst.*, 10(1), aug 2019. ISSN 2160-6455. doi: 10.1145/3241381. URL `https://doi.org/10.1145/3241381`. 40, 41, 42, 46, 49

[102] T. Escovedo e A. Koshiyama. *Introdução a Data Science: Algoritmos de Machine Learning e métodos de análise*. Casa do Código, 2020. ISBN 9788572540551. URL `https://books.google.com.br/books?id=cL7TDwAAQBAJ`. 41

[103] Anis Elbahi e Mohamed Nazih Omri. Conditional random fields for web user task recognition based on human computer interaction. In *Conditional Random Fields For Web User Task Recognition Based On Human Computer Interaction*, 06 2015. doi: 10.13140/RG.2.1.4219.2487. 41, 43

[104] John S. Breese, David Heckerman, e Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, UAI'98, page 43–52, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc. ISBN 155860555X. 41, 43

[105] Neil Heffernan e Cristina Heffernan. The assistments ecosystem: Building a platform that brings scientists and teachers together for minimally invasive research on human learning and teaching. *International Journal of Artificial Intelligence in Education*, 24, 12 2014. doi: 10.1007/s40593-014-0024-x. 41

[106] Mozilla. Websocket - apis da web, 2021. URL `https://developer.mozilla.org/pt-BR/docs/Web/API/WebSockets_API`. 43

[107] Armin Balalaie, Abbas Heydarnoori, e Pooyan Jamshidi. Microservices architecture enables devops: Migration to a cloud-native architecture. *IEEE Software*, 33(3):42–52, 2016. doi: 10.1109/MS.2016.64. 43

[108] Cameron B. Browne, Edward Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, e Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):1–43, 2012. doi: 10.1109/TCIAIG.2012.2186810. 45

[109] Kristina P. Sinaga e Miin-Shen Yang. Unsupervised k-means clustering algorithm. *IEEE Access*, 8:80716–80727, 2020. doi: 10.1109/ACCESS.2020.2988796. 45

[110] Derya Birant e Alp Kut. St-dbscan: An algorithm for clustering spatial-temporal data. *Data Knowl. Eng.*, 60(1):208–221, jan 2007. ISSN 0169-023X. doi: 10.1016/j.datak.2006.01.013. URL https://doi.org/10.1016/j.datak.2006.01.013. 45, 49

[111] S. Lemeshow D. W. Hosmer e R. X. Sturdivant. *Applied logistic regression*, volume 398. John Wiley & Sons, 2013. 45, 49

[112] Robert E. Schapire. *Explaining AdaBoost*, pages 37–52. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. ISBN 978-3-642-41136-6. doi: 10.1007/978-3-642-41136-6_5. URL https://doi.org/10.1007/978-3-642-41136-6_5. 45, 49

[113] Scikit-learn. Scikit-learn: machine learning in python - 1.1.0, 2022. URL https://developer.mozilla.org/pt-BR/docs/Web/API/WebSockets_API. 45, 87

[114] Rebai Rim, Mohamed Amine Maalej, Adel Mahfoudhi, e Mohamed Abid. Bayesian user modeling: evaluation metrics of an adaptive user interface. In *Bayesian user modeling: evaluation metrics of an adaptive user interface*, page 103412F, 03 2017. doi: 10.1117/12.2268568. 46

[115] Francesca Gullà, Lorenzo Cavalieri, Silvia Ceccacci, e Michele Germani. A bbn-based method to manage adaptive behavior of a smart user interface. *Procedia CIRP*, 50:535–540, 2016. ISSN 2212-8271. doi: https://doi.org/10.1016/j.procir.2016.04.162. URL https://www.sciencedirect.com/science/article/pii/S2212827116304024. 26th CIRP Design Conference. 46

[116] Leslie Pack Kaelbling, Michael L. Littman, e Andrew W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4(1):237–285, may 1996. ISSN 1076-9757. 50

[117] N. Rozanski e E. Woods. *Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives*. Addison-Wesley, 2012. ISBN 9780321718334. URL https://books.google.com.br/books?id=ka4QO9kXQFUC. 51, 85

[118] Jamil Hussain, Anees Ul Hassan, Hafiz Bilal, Rahman Ali, Muhammad Afzal, Shujaat Hussain, Jae Bang, Oresti Banos, e Sungyoung Lee. Model-based adaptive user interface based on context and user experience evaluation. *Journal on Multimodal User Interfaces*, 12, 02 2018. doi: 10.1007/s12193-018-0258-2. 52

[119] L. A. Zadeh R. E. Bellman. Decision-making in a fuzzy environment. management science. *Management Science*, 17, 12 1970. doi: 10.1287/mnsc.17.4.B141. 53

[120] Levente Kocsis e Csaba Szepesvári. Bandit based monte-carlo planning. In Johannes Fürnkranz, Tobias Scheffer, e Myra Spiliopoulou, editors, *Machine Learning: ECML 2006*, pages 282–293, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. ISBN 978-3-540-46056-5. 55

[121] Bowen Hui, Grant Partridge, e Craig Boutilier. A probabilistic mental model for estimating disruption. In *Proceedings of the 14th International Conference on Intelligent User Interfaces*, IUI '09, page 287–296, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605581682. doi: 10.1145/1502650.1502691. URL https://doi.org/10.1145/1502650.1502691. 57

[122] Antti Oulasvirta, Niraj Ramesh Dayama, Morteza Shiripour, Maximilian John, e Andreas Karrenbauer. Combinatorial optimization of graphical user interface designs. *Proceedings of the IEEE*, 108(3):434–464, 2020. doi: 10.1109/JPROC.2020.2969687. 58

[123] Jean Vanderdonckt, Sara Bouzit, Gaëlle Calvary, e Denis Chêne. Exploring a design space of graphical adaptive menus: Normal vs. small screens. *ACM Trans. Interact. Intell. Syst.*, 10(1), jul 2019. ISSN 2160-6455. doi: 10.1145/3237190. URL https://doi.org/10.1145/3237190. 59

[124] Stephen Fitchett e Andy Cockburn. Accessrank: Predicting what users will do next. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, page 2239–2242, New York, NY, USA, 2012. Association for Computing Machinery. ISBN 9781450310154. doi: 10.1145/2207676.2208380. URL https://doi.org/10.1145/2207676.2208380. 59

[125] Peter A. Frensch. Composition during serial learning: a serial position effect. *Journal of experimental psychology. Learning, memory, and cognition*, 20 2:423–42, 1994. 59

[126] Camille Gobert, Kashyap Todi, Gilles Bailly, e Antti Oulasvirta. Sam: A modular framework for self-adapting web menus. In *Proceedings of the 24th International Conference on Intelligent User Interfaces*, IUI '19, page 481–484, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450362726. doi: 10.1145/3301275.3302314. URL https://doi.org/10.1145/3301275.3302314. 59

[127] Kashyap Todi, Jussi Jokinen, Kris Luyten, e Antti Oulasvirta. Familiarisation: Restructuring layouts with visual learning models. In *23rd International Conference on Intelligent User Interfaces*, IUI '18, page 547–558, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450349451. doi: 10.1145/3172944.3172949. URL https://doi.org/10.1145/3172944.3172949. 59

[128] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, e Clifford Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009. ISBN 0262033844. 60

[129] Pete Faraday. Visually critiquing web pages. In *Eurographics Multimedia Workshop*, pages 155–166, 1999. 61

[130] Gerhard Fischer, Andreas C. Lemke, Thomas Mastaglio, e Anders I. Morch. Using critics to empower users. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '90, page 337–347, New York, NY, USA, 1990. Association for Computing Machinery. ISBN 0201509326. doi: 10.1145/97243.97305. URL https://doi.org/10.1145/97243.97305. 62

[131] Paul C. Quinn, Candice R. Brown, e Michele L. Streppa. Perceptual organization of complex visual configurations by young infants. *Infant Behavior and Development*, 20(1):35–46, 1997. ISSN 0163-6383. doi: https://doi.org/10.1016/S0163-6383(97)90059-X. URL https://www.sciencedirect.com/science/article/pii/S016363839790059X. 62

[132] Gilles Bailly, Antti Oulasvirta, Duncan P. Brumby, e Andrew Howes. Model of visual search and selection time in linear menus. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, page 3865–3874, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450324731. doi: 10.1145/2556288.2557093. URL https://doi.org/10.1145/2556288.2557093. 62, 64, 65

[133] MICHAEL D. BYRNE. Act-r/pm and menu selection: applying a cognitive architecture to hci. *International Journal of Human-Computer Studies*, 55(1):41–84, 2001. ISSN 1071-5819. doi: https://doi.org/10.1006/ijhc.2001.0469. URL https://www.sciencedirect.com/science/article/pii/S1071581901904690. 62

[134] Andy Cockburn e Carl Gutwin. A predictive model of human performance with scrolling and hierarchical lists. *Human–Computer Interaction*, 24(3):273–314, 2009. doi: 10.1080/07370020902990402. URL https://www.tandfonline.com/doi/abs/10.1080/07370020902990402. 62

[135] K.L. Norman. *The Psychology of Menu Selection: Designing Cognitive Control at the Human/computer Interface*. G - Reference,Information and Interdisciplinary Subjects Series. Ablex Publishing Corporation, 1991. ISBN 9780893915537. URL https://books.google.com.br/books?id=Li9ixNhMwUUC. 62, 64

[136] Andy Cockburn, Carl Gutwin, e Saul Greenberg. A predictive model of menu performance. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '07, page 627–636, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 9781595935939. doi: 10.1145/1240624.1240723. URL https://doi.org/10.1145/1240624.1240723. 64

[137] Webpack. Webpack, 2022. URL https://webpack.js.org/. 69, 87, 89

[138] Webpack. This is often known as micro-frontends, but is not limited to that., 2022. URL https://webpack.js.org/concepts/module-federation/. 70, 87, 89

[139] GraphQL. A query language for your api, 2022. URL https://graphql.org/. 70, 73, 87

[140] Amazon Web Services. Benefits at a glance, 2023. URL https://aws.amazon.com/application-hosting/benefits/. 70, 83

[141] Chaitanya K Rudrabhatla. Comparison of event choreography and orchestration techniques in microservice architecture. *International Journal of Advanced Computer Science and Applications*, 9(8), 2018. 72

[142] Antonio Quiña-Mera, Pablo Fernández-Montes, José María García, Edwin Bastidas, e Antonio Ruiz-Cortés. Quality in use evaluation of a graphql implementation. In Miguel Botto-Tobar, Henry Cruz, Angela Díaz Cadena, e Benjamin Durakovic, editors, *Emerging Research in Intelligent Systems*, pages 15–27, Cham, 2022. Springer International Publishing. ISBN 978-3-030-96043-8. 72

[143] Alex Rodriguez. Restful web services: The basics. *IBM developerWorks*, 33(2008): 18, 2008. 72

[144] IBM. Introdução ao ibm websphere mq, 2022. URL https://www.ibm.com/docs/pt-br/ibm-mq/7.5?topic=ssfksj-7-5-0-com-ibm-mq-pro-doc-q001020--htm. 73

[145] Microsoft. Microsoft msmq, 2022. URL https://docs.microsoft.com/en-us/previous-versions/windows/desktop/msmq/ms711472(v=vs.85). 73

[146] Active MQ. Flexible and powerful open source, 2022. URL https://activemq.apache.org/. 73

[147] Amazon. Amazon eventbridge, 2022. URL https://aws.amazon.com/pt/eventbridge/. 73

[148] Zakir Laliwala e Sanjay Chaudhary. Event-driven service-oriented architecture. In *2008 International Conference on Service Systems and Service Management*, pages 1–6, 2008. doi: 10.1109/ICSSSM.2008.4598452. 73

[149] Grace Jansen e Johanna Saladas. Advantages of the event-driven architecture pattern, 2020. URL https://developer.ibm.com/articles/advantages-of-an-event-driven-architecture/. 73, 74

[150] Alam Rahmatulloh, Fuji Nugraha, Rohmat Gunawan, e Irfan Darmawan. Event-driven architecture to improve performance and scalability in microservices-based systems. In *2022 International Conference Advancement in Data Science, E-learning and Information Systems (ICADEIS)*, pages 01–06, 2022. doi: 10.1109/ICADEIS56544.2022.10037390. 74

[151] Simon Tragatschnig, Srdjan Stevanetic, e Uwe Zdun. Supporting the evolution of event-driven service-oriented architectures using change patterns. *Information and Software Technology*, 100:133–146, 2018. ISSN 0950-5849. doi: https://doi.org/10.1016/j.infsof.2018.04.005. URL https://www.sciencedirect.com/science/article/pii/S0950584916303251. 74

[152] M. Geers. *Micro Frontends in Action*. Manning Publications, 2020. ISBN 9781617296871. URL https://books.google.com.br/books?id=5f1VzQEACAAJ. 75

[153] Cam Jackson. Micro frontends, 2019. URL https://martinfowler.com/articles/micro-frontends.html. 75

[154] Neeraj Sangal, Ev Jordan, Vineet Sinha, e Daniel Jackson. Using dependency models to manage complex software architecture. *SIGPLAN Not.*, 40(10):167–176, oct 2005. ISSN 0362-1340. doi: 10.1145/1103845.1094824. URL https://doi.org/10.1145/1103845.1094824. 76

[155] Pahan Perera. React micro frontends with webpack's module federation, 2021. URL https://dev.to/pahanperera/react-micro-frontends-with-webpacks-module-federation-32ii. 76

[156] Dewayne E. Perry e Alexander L. Wolf. Foundations for the study of software architecture. *ACM SIGSOFT*, 17(4):40–52, oct 1992. ISSN 0163-5948. doi: 10.1145/141874.141884. URL https://doi.org/10.1145/141874.141884. 84

[157] David Garlan e Mary Shaw. An introduction to software architecture. Technical report, School of Computer Science, Carnegie Mellon University, USA, 1994. 84

[158] C. Richardson. *Microservices Patterns: With examples in Java*. Manning, 2018. ISBN 9781638356325. URL https://books.google.com.br/books?id=QTgzEAAAQBAJ. 85

[159] B. Stopford e an O'Reilly Media Company Safari. *Designing Event-Driven Systems*. O'Reilly Media, Incorporated, 2018. URL https://books.google.com.br/books?id=-G49zQEACAAJ. 85

[160] Minyi Zhong e Christos G. Cassandras. Asynchronous distributed optimization with event-driven communication. *IEEE Transactions on Automatic Control*, 55 (12):2735–2750, 2010. doi: 10.1109/TAC.2010.2049518. 85

[161] A.S. Tanenbaum e M. Van Steen. *Sistemas distribuídos: princípios e paradgmas*. Pearson Educación, 2007. ISBN 9788576051428. URL https://books.google.com.br/books?id=r2SGPgAACAAJ. 86

[162] Soufiane Maguerra, Azedine Boulmakoul, Lamia Karim, Badir Hassan, e Ahmed Lbath. Towards a reactive system for managing big trajectory data. *Journal of Ambient Intelligence and Humanized Computing*, 11, 10 2020. doi: 10.1007/s12652-019-01625-3. 86

[163] React. React - uma biblioteca javascript para criar interfaces de usuário, 2022. URL https://pt-br.reactjs.org/. 87

[164] Node JS. Node.js is a javascript runtime built on chrome's v8 javascript engine., 2022. URL https://nodejs.org/en/. 87, 91

[165] Python. Python is a programming language that lets you work quickly and integrate systems more effectively., 2022. URL https://www.python.org/. 87, 91

[166] Docker. Developers love docker. businesses trust it., 2022. URL https://www.docker.com/. 87

[167] MongoDB. Working with data doesn't need to be hard, 2022. URL https://www.mongodb.com/pt-br. 87

[168] Amazon. Amazon web services, 2023. URL https://aws.amazon.com/. 91

[169] Amazon. Aws lambda, 2022. URL https://aws.amazon.com/pt/lambda/. 91

[170] AWS. Aws appsync, 2022. URL https://aws.amazon.com/pm/appsync. 91

[171] Elar Saks. Javascript frameworks: Angular vs react vs vue. 2019. 92

[172] Sanchit Aggarwal *et al.* Modern web-development using reactjs. *International Journal of Recent Research Aspects*, 5(1):133–137, 2018. 92

[173] Artemij Fedosejev. *React. js essentials*. Packt Publishing Ltd, 2015. 92

[174] S Shahu Gaikwad e PRATIBHA Adkar. A review paper on bootstrap framework. *IRE Journals*, 2(10):349–351, 2019. 92

[175] W3Schools. What is bootstrap, 2023. URL `https://www.w3schools.com/whatis/whatis_bootstrap.asp`. 93

[176] Jake Spurlock. *Bootstrap: responsive web development*. "O'Reilly Media, Inc.", 2013. 93

[177] AWS. What is cloud computing, 2023. URL `https://aws.amazon.com/what-is-cloud-computing/`. 93

[178] Eric Knorr e Galen Gruman. What cloud computing really means. *InfoWorld*, 7 (20-20):1–17, 2008. 93

[179] AWS. What is aws cdk?, 2023. URL `https://docs.aws.amazon.com/pt_br/cdk/v2/guide/home.html`. 94

[180] Bradley Campbell e Bradley Campbell. The aws cdk and pulumi. *The Definitive Guide to AWS Infrastructure Automation: Craft Infrastructure-as-Code Solutions*, pages 237–272, 2020. 94

[181] Christian Nwamba. Your first look at graphql queries, mutations, and subscriptions, 2023. URL `https://www.telerik.com/blogs/your-first-look-at-graphql-queries-mutations-and-subscriptions`. 95

[182] Daniel Lemire e Melissa E. O'Neill. Xorshift1024*, xorshift1024+, xorshift128+ and xoroshiro128+ fail statistical tests for linearity. *Journal of Computational and Applied Mathematics*, 350:139–142, 2019. ISSN 0377-0427. doi: https://doi.org/10.1016/j.cam.2018.10.019. URL `https://www.sciencedirect.com/science/article/pii/S0377042718306265`. 99, 100

Anexo I

Lista ordenada contendo as atividades realizadas durante a validação empírica

1. O autor deste trabalho questionou se o provável participante poderia participar de uma pesquisa de opinião pública em relação a uma aplicação bancária.
2. O participante confirmou que poderia participar.
3. O autor apresentou um vídeo ao participante ao qual explica de maneira geral o projeto e também apresenta um tutorial de utilização da aplicação.
4. O autor questionou se o participante consentia em continuar com a pesquisa, e também se o participante poderia assinar um termo de consentimento.
5. Para preencher o termo de consentimento, as seguintes perguntas foram realizadas:
   (a) Qual o seu nome?
   (b) Qual o seu e-mail?
   (c) Qual a sua idade?
   (d) Qual o seu gênero?
   (e) Qual o seu nível de escolaridade?
6. Após isso, o termo foi enviado de forma automática para o e-mail do participante.
7. Com isso, a pesquisa foi iniciada. No primeiro momento, o participante respondeu as seguintes perguntas:
   (a) Você acessa serviços bancários por meio da internet?

(b) Você acessa serviços bancários por meio de um computador?

8. Finalmente, o participante começou a utilizar a aplicação, sendo que na primeira tela continha um botão com a frase "Iniciar Teste".

9. Após clicar no botão "Iniciar Teste", a página inicial da aplicação foi mostrada, a qual continha os serviços bancários disponíveis em forma de linhas e colunas, contendo também uma breve explicação de cada serviço.

10. O participante realizou 3 sessões de navegação, sendo que uma sessão é composta por 20 cliques, para cada condição (Estática, Frequência, MCTS).

11. Ao final de cada sessão, o participante responde o seguinte questionário:

    (a) Você notou alguma alteração na tela durante o uso?

    (b) Se sim, você julga que melhorou a utilização?

    (c) Por que melhorou ou piorou?

12. Após as sessões de todas as condições serem finalizadas, um botão com a frase "Encerrar Teste"foi mostrado para o usuário.

13. Foi explicado aos participantes a seguinte correspondência entre números e categoria: 1: Péssimo, 2: Ruim, 3: Indiferente, 4: Bom, 5: Ótimo;

14. Por fim, o usuário respondeu ao último conjunto de perguntas:

    (a) De 1 a 5, qual nota você atribui para a aparência do *frontend*? Cor, imagens, tamanho.

    (b) De 1 a 5, qual nota você atribui para a facilidade de uso do sistema?

    (c) De 1 a 5, qual nota você atribui para o tempo de resposta entre os cliques?

    (d) De 1 a 5, qual nota você atribui para a disposição dos elementos? Isto é, organização dos elementos no *frontend*.

    (e) De 1 a 5, qual nota você atribui para a sua satisfação em utilizar esse site, que representa a página inicial de um sistema bancário?

    (f) Use este espaço para apresentar os comentários acerca do sistema.

15. A validação teve uma duração média de 23 minutos e 33 segundos por participante. Cada condição teve uma duração média de 7,77 minutos e cada sessão teve uma duração média de 2,59 minutos.