UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS DE QUIXADÁ

CURSO DE GRADUAÇÃO EM ENGENHARIA DE SOFTWARE

GABRIEL UCHOA DE VASCONCELOS

DESENVOLVIMENTO DE UMA PLATAFORMA WEB PARA GERENCIAMENTO DE FILAS ONLINE PARA BANCOS

QUIXADÁ

2025

GABRIEL UCHOA DE VASCONCELOS

DESENVOLVIMENTO DE UMA PLATAFORMA WEB PARA GERENCIAMENTO DE FILAS ONLINE PARA BANCOS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Software do da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Software.

Orientador: Prof. Dr. Antônio Joel Ramiro de Castro.
Coorientadora: Prof. Dra. Maria do Socorro Assis Braun.

QUIXADÁ

2025

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

V45d Vasconcelos, Gabriel Uchoa de.
Desenvolvimento de uma plataforma web para gerenciamento de filas online para bancos / Gabriel Uchoa de Vasconcelos. – 2025.
60 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Engenharia de Software, Quixadá, 2025.
Orientação: Prof. Dr. Antônio Joel Ramiro de Castro.
Coorientação: Prof. Dr. Maria do Socorro Assis Braun.

1. Padrões de desenvolvimento. 2. Padrões de software. 3. Softwares. 4. Desenvolvimento de tecnologia. I. Título.

CDD 005.1

GABRIEL UCHOA DE VASCONCELOS

DESENVOLVIMENTO DE UMA PLATAFORMA WEB PARA GERENCIAMENTO DE FILAS ONLINE PARA BANCOS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Software do da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Software.

Aprovada em: 23 de julho de 2025

BANCA EXAMINADORA

Prof. Dr. Antônio Joel Ramiro de Castro (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dra. Maria do Socorro Assis Braun (Coorientadora)
Instituto Federal do Ceará (IFCE)

Prof. Dr. Thiago Werlley Bandeira da Silva
Universidade Federal do Ceará (UFC)

Prof. Dr. José Gadelha da Silva Filho
Universidade Estadual do Ceará (UECE)

Dedico este trabalho à minha família, pelo apoio, incentivo e orientação ao longo de toda a minha jornada.

Aos meus amigos, que estiveram comigo nos momentos mais difíceis e me ajudaram a seguir em frente.

Esta conquista é de vocês também.

AGRADECIMENTOS

Este trabalho só foi possível graças ao apoio de pessoas muito especiais, que caminharam comigo durante essa longa jornada.

Agradeço, primeiramente, à minha família, que sempre acreditou em mim. Aos meus pais, pelo incentivo constante e por nunca deixarem que eu desistisse. À minha madrasta Socorro, que me orientou com dedicação, junto ao meu orientador Joel, contribuindo imensamente para que eu pudesse concluir este trabalho com clareza e confiança.

Aos meus amigos, que foram fundamentais para que este projeto saísse do papel: Johnny Marcos, que me ajudou a configurar o ambiente e colocar o sistema no ar; Lucas, que teve paciência ao me explicar os conceitos e configurações do Kafka; Shélida e Letícia, que contribuíram com o design das interfaces e me ajudaram a pensar na experiência do usuário.

Agradeço também ao Crato, Lara e Kevin, que estiveram ao meu lado me apoiando no desenvolvimento do TCC, mesmo nos momentos mais difíceis. E, em especial, ao Gerardo Magela, que foi mais que um colega: foi um verdadeiro parceiro de desenvolvimento, com quem pude trocar ideias, superar desafios e aprender muito ao longo do processo.

A todos vocês, meu mais sincero agradecimento. Esse TCC é fruto de um esforço coletivo e carinhoso, que jamais esquecerei.

RESUMO

Este trabalho apresenta o desenvolvimento de uma plataforma web para gerenciamento de filas online em ambientes bancários, com o objetivo de minimizar o tempo de espera presencial e melhorar a experiência do cliente. O sistema permite a retirada de senhas de forma remota, fornece estimativas de tempo de atendimento em tempo real e organiza a fila com base em prioridades, como senhas normais e prioritárias. A metodologia adotada foi exploratória e descritiva, utilizando uma abordagem incremental e iterativa para o desenvolvimento do software. Foram implementadas tecnologias como Java com Spring Boot no backend, ReactJS no frontend, Apache Kafka para mensageria e WebSocket para comunicação em tempo real. A plataforma foi implantada em nuvem (Google Cloud) com dados simulados, validando os fluxos principais e a integração entre os serviços. Os resultados demonstram que a solução contribui para reduzir aglomerações, otimizar o atendimento e oferecer uma experiência mais previsível e satisfatória aos usuários. Como trabalhos futuros, propõe-se a integração com sistemas físicos de triagem, autenticação de usuários e a inclusão de painéis de chamada de senhas.
Palavras-chave:

Palavras-chave: Filas bancárias; Plataforma web; Gerenciamento de filas; Kafka; WebSocket.

ABSTRACT

This study presents the development of a web platform for online queue management in banking environments, aiming to minimize on-site waiting times and improve customer experience. The system allows users to obtain tickets remotely, provides real-time waiting time estimates, and organizes the queue based on priorities, such as regular and priority tickets. The methodology adopted was exploratory and descriptive, using an incremental and iterative approach to software development. Technologies such as Java with Spring Boot for the backend, ReactJS for the frontend, Apache Kafka for messaging, and WebSocket for real-time communication were implemented. The platform was deployed in the cloud (Google Cloud) with simulated data, validating the main workflows and the integration between services. The results show that the solution helps reduce crowds, optimize service, and provide a more predictable and satisfactory experience for users. Future work includes integration with physical triage systems, user authentication, and the inclusion of ticket display panels.

Keywords: Banking queues; Web platform; Queue management; Kafka; WebSocket.

SUMÁRIO

1 INTRODUÇÃO

No século XIII o famoso físico Benjamin Franklin (1706-1790), após um longo estudo sobre as obras do filósofo grego Teofrasto (372-288 a.C.), chegou a uma conclusão que se tornou famosa e amplamente difundida pelo mundo: "tempo é dinheiro". Esta simples frase, cunhada há mais de dois séculos, ainda é usada de forma orgulhosa pelas instituições bancárias, incluindo as do Brasil, ao investir amplamente em tecnologia (Ramos, 2015).

Entretanto, nos bancos nacionais, a ironia dessa afirmação se revela com a crescente insatisfação dos clientes, especialmente ao tratar das longas e demoradas filas. Se "tempo é dinheiro", por que as instituições que prestam os serviços bancários continuam com suas intermináveis filas, por mais que aleguem investimentos em equipamentos e tecnologia para facilitar a vida dos clientes? O problema permanece: o fato de que as filas permanecem, independentemente da instituição bancária, locais onde, ironicamente, o tempo do cliente não é devidamente valorizado (Ramos, 2015).

Nessa perspectiva, as filas bancárias representam um problema que ultrapassa o prejuízo ao atendimento, chegando a patamares financeiros, como as diversas instituições bancárias que já foram multadas pelo PROCON em diversas regiões do país. Um exemplo dessa situação foi a multa do Procon-ES à Caixa Econômica Federal em mais de R$ 110 mil por amplas filas e atendimento demorado (Procon, 2021). Em outra situação o Procon multou vários bancos em mais de R$ 250 mil por tempo de espera e por aglomerações nas filas durante a pandemia, tendo o Itaú e o Bradesco recebido as maiores multas, R$ 113.260,12 e R$ 98.494.82, respectivamente (Umuarama, 2021).

Segundo a FEBRABAN (2022), a indústria bancária é um dos setores que mais investe em tecnologia, tanto no Brasil quanto no mundo, pois de acordo com levantamento realizado pela Gartner, o setor bancário fica apenas atrás dos governos na composição dos dispêndios em tecnologia em 2021. O ano de 2022 foi o primeiro ano em que a indústria bancária brasileira ultrapassou a média global na proporção de investimentos em tecnologia.

Ainda de acordo com a instituição, os investimentos de 30,1 bilhões de reais em 2021 na área de tecnologia, esses recursos foram alocados principalmente para as áreas de segurança cibernética, inteligência artificial, 5G, cloud pública e Big Data FEBRABAN (2022), entretanto, nenhum investimento foi alocado priorizando seus clientes e a melhoria da qualidade de serviço, principalmente para a redução de tempo de espera em filas.

Pode-se perceber que, mesmo com investimentos em múltiplos setores, o problema

ainda não foi devidamente resolvido. Apesar do que se parece ser uma crítica fácil, pois os investimentos foram em áreas de maior interesse dos bancos, vistos serem empresas, encontrar um desenlace a contento ainda é um desafio. Portanto, a colaboração é fundamental na busca por soluções cujo objetivo seja a erradicação, ou pelo menos, a minimização das filas (RAMOS, 2015).

Os investimentos na mecanicidade dos bancos eletrônicos e instantâneos acaba por refletir no devido treinamento em pessoal, cujas atuações frias e distantes os faz soarem robóticos em um atendimento mecânico, cujo trato com os cliente acaba sendo desprovido de motivação ou calor humano (RAMOS, 2015). Esse distanciamento os torna alheios ao desconforto das pessoas, cuja espera em pé por vários minutos ou horas, demonstra a falta de preocupação do setor humano "atendimento"com o setor humano "cliente"(RAMOS, 2015). Esta inversão é contra intuitiva, pois a existência do cliente é a razão da existência do atendimento, algo que parece ser negligenciado.

Dito isso, é fato que, até hoje, há enormes concentrações de pessoas à espera de atendimento, principalmente em filas de banco. Ou seja, para acessar as instituições financeiras os clientes perdem muito tempo. Portanto, além de investir em tecnologia, é necessário adaptar os sistemas que hoje funcionam de forma precária, visto que estes não evitam aglomerações. Isso pode ser visto na implementação de caixas eletrônicos que, ainda que representem investimentos em tecnologia de autoatendimento por parte dos bancos, não conseguiram, efetivamente, solucionar o problema vigente da espera pelo atendimento em virtude das filas (Araújo; Carneiro, 2008). Desse modo, ainda faltam contribuições mais efetivas para melhorar a segurança e qualidade de vida para os consumidores dos serviços.

Este trabalho tem como objetivo desenvolver uma aplicação web geradora de senhas online, com estimativa de tempo de atendimento e a análise de satisfação do usuário, visando testar a aceitação do sistema online em diversas faixas etárias, avaliando níveis de entendimento da interface, facilidade de uso, entendimento das informações expostas e a adesão a ferramenta.

Desse modo, o sistema visa propor uma solução que melhore a qualidade dos serviços bancários, fazendo com que os clientes esperem menos nas filas presenciais, aumentando sua satisfação com os bancos, visto que o tempo de espera nas filas presenciais seriam menores, permitindo realizar outras atividades.

1.1 Objetivos

1.1.1 Objetivo Geral

– Desenvolver uma aplicação web geradora de senhas online com estimativa de tempo de atendimento para melhorar a qualidade dos serviços bancários.

1.1.2 Objetivos Específicos

– Desenvolver uma plataforma para retirada de senha virtual.
– Implementar mecanismos de fila dinâmica com tratamento de prioridades entre senhas normais e prioritárias.
– Estimar e exibir o tempo de espera com base em dados históricos e no ritmo atual de atendimentos.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Fila de espera em instituições bancárias

A fila, por definição, é composta por três fatores: i) a entrada ou chegada de clientes; ii) as regra de enfileiramento; e iii) os posto de serviços ou de atendimento. No aspecto social, a fila, como um fenômeno diário e global, inclui diversos serviços, de transporte à comunicações, cuja dinâmica ainda precisa ser melhor estudada para melhor operação desses serviços. (Li; Zhang, 2015; Xu; Liu, 2012)

Outro aspecto importante é a capacidade de atendimento de um sistema e suas dificuldades de prever a quantidade de demanda de um serviço, conforme Figueiredo e Escobar (2004) afirmam:

> "A capacidade do sistema formado pelos caixas de um banco pode representar o número de pessoas que tal sistema pode atender em uma hora. Mas a complexidade maior advém do fato de que os clientes que se dirigem a uma agência bancária podem ter demandas distintas, o que faz com que cada atendimento seja diferente do outro em termos de tempo de atenção ao cliente."

A formação de uma fila não é causada apenas por restrições na capacidade de atendimento de um serviço, mas também devido à variabilidade tanto no intervalo entre a chegada de clientes como na duração do atendimento ou serviço prestado a eles, visto que usuários com demandas distintas vão requerer tempos distintos de atendimento.

Estudos de teoria de filas mostram que grandes variações no tempo de atendimento podem causar gargalos significativos no fluxo de clientes, resultando em maiores tempos de espera e, consequentemente, baixa satisfação do usuário (Gross *et al.*, 2018). Esse cenário é particularmente crítico no setor bancário, onde autuações por filas extensas têm se tornado frequentes. Dados recentes apontam que, em 2023, o Procon-RJ autuou agências por tempos de espera superiores a 40 minutos (Procon-RJ. . . , 2023). Em São Paulo, o Procon-SP já havia aplicado multas semelhantes em 2014 devido a filas de até 1h23 (Fila. . . , 2014), evidenciando que o problema é recorrente no Brasil.

2.2 Engenharia de Requisitos

A engenharia de requisitos (ER) é uma das etapas mais importantes no desenvolvimento de um software, uma vez que conforme Sommerville (2019, p.39) "a engenharia de requisitos é um estágio particularmente critico do processo de software, pois os erros cometidos

nessa etapa inevitavelmente geram problemas posteriores no projeto e na implementação do sistema". Ainda o mesmo autor Sommerville (2019, p.39) define engenharia de requisitos como "o processo de compreender e definir quais serviços são necessários para o sistema e identificar as restrições sobre sua operação e desenvolvimento".

Dessa forma, as organizações tendem a ter etapas de engenharia de requisitos diferentes, adaptadas para atender suas peculiaridades e suas realidades. No entanto, é possível estabelecer quatro atividades básicas para sua definição, são elas: (i) elicitação dos requisitos; (ii) documentação dos requisitos; (iii) validação e negociação dos critérios de aceitação; (iv) gerenciamento dos requisitos. (Pohl, 2016)

A primeira etapa da ER, conhecida como elicitação dos requisitos, é a mais crítica, uma vez que nela ocorre o levantamento dos requisitos, que muitas vezes estão incompletos. Ela é usualmente definida como uma etapa de análise, visando descobrir, extrair e analisar as necessidades do usuário(Alexa; Avasilcai, 2018) . É crítica porque os clientes têm dificuldade em articular suas necessidades até que as vejam, sendo, muitas vezes, impossível para um usuário, mesmo que este trabalhe como um engenheiro de software, especificar exatamente os requisitos de um produto de software, só é possível após testar alguma versão do produto. (Vijayan *et al.*, 2016)

Dessa forma, se faz necessário construir uma ponte de diálogo entre o cliente e o desenvolvedor. A forma normalmente usada para isso é através das User Story ou Histórias do Usuário, que são estruturas de informações que conseguem capturar a explicação informal de um usuário e transmitir o que é necessário para o desenvolvedor. Isso se dá porque essas histórias de usuários são constituídas pelo sumário do requisito, pela métrica para considerar a requisição completa, qual o grau de importância/prioridade daquela funcionalidade pro cliente, quem vai usar e o passo a passo do que deve ser feito para a tarefa ser concluída. Essa necessidade de estar bem especificado se dá porque imprecisões na especificação de requisitos podem gerar conflitos entre os clientes e os desenvolvedores de software. Esses conflitos acontecem principalmente quando há algum tipo de requisito ambíguo, fazendo com que novos requisitos e novas mudanças sejam geradas. (Rehkopf, ; Sommerville, 2019)

Como resultado dessa etapa, tem-se a documentação dos requisitos do projeto. (Sommerville, 2019)

2.2.1 Requisitos Funcionais

Os requisitos funcionais, descrevem, em detalhes, funcionalidades dos sistemas, suas entradas, suas saídas e suas exceções mas não se restringem somente a isso, já que devem abranger desde de outros sistemas utilizados pela empresa até a sua cultura organizacional. (Sommerville, 2019)

2.2.2 Requisitos Não Funcionais

Os requisitos não funcionais(RNF) são requisitos que não estão diretamente relacionado com funcionalidades específicas oferecidas pelo sistema e sim como características da aplicação, como confiabilidade, tempo de resposta, etc. Esses RNF também podem ser encarados como restrições do software e se tornam mais importante devido ao fato de que, muitas vezes, os RNF são mais críticos para a aplicação, já que essas características, se não cumpridas, podem comprometer o uso do sistema. Um exemplo disso sistema de aeronave não satisfizer seus requisitos de confiabilidade, impedindo de que este seja certificado como seguro para operação. (Sommerville, 2019)

Em sistemas de missão crítica, como o desenvolvido neste trabalho, os requisitos não funcionais assumem maior relevância. Características como escalabilidade, alta disponibilidade e segurança são essenciais quando se utiliza arquiteturas de microsserviços e filas de mensagens como Kafka, tecnologias amplamente consolidadas no mercado (Newman, 2021).

2.3 Aplicações Web-Based

A definição de aplicações baseadas em web, segundo Cardoso (2012), pode ser estabelecida como: "um software que você usa através da internet e tendo um web browser como interface. Você não precisa instalar nenhum CD ou fazer download de nenhum software ou se preocupar com atualizações". Além disso, as aplicações baseadas em web são do tipo Cliente-Servidor, o que segundo Oluwatosin (2014, tradução nossa) seria definido como: "uma arquitetura de software composta por cliente e servidor, onde os clientes sempre enviam solicitações enquanto o servidor responde às solicitações enviadas."

2.3.1 Backend e Frontend

A relação servidor-cliente é dividida em duas partes, sendo a do servidor chamada de backend e a do cliente é chamada de frontend. Segundo Nice (2018, tradução nossa), "Back-end normalmente se refere às entranhas do aplicativo que reside no servidor (geralmente é chamado de 'lado do servidor'). O back-end está mais focado em garantir que todos os dados corretos sejam enviados ao navegador", enquanto segundo Nice (2018, tradução nossa), "Front End normalmente se refere ao que você realmente vê no site no navegador (geralmente é chamado de 'lado do cliente'). Isso cobre como o conteúdo é apresentado, incluindo todos os pequenos elementos da interface do usuário, como menus, menus suspensos, transições e modais."

Os frameworks são usados tanto no backend quanto no frontend para auxiliar no desenvolvimento devido ao fato deles proporcionarem estruturas que servem de base para construção de diversas aplicações, sejam elas sites, aplicativos ou softwares. Essas simplificações feitas pelo framework costumam ser esqueletos que se repetem em diversos ambiente mas que costumam ser muito custosas ou problemáticas para se implementar. (Nice, 2018)

Os testes automatizados podem ser definidos como um trecho de código que testa outro trecho de código. São escritos, normalmente, pelos desenvolvedores ou profissionais de qualidade de software, também conhecido como QA (Quality Assurance ou controle de qualidade). Esses testes servem para garantir a integridade da aplicação por todo seu ciclo de vida, garantindo que os novos trechos de código inseridos não irão afetar o funcionamento da aplicação. (Moura, 2019)

O banco de dados relacional trabalha com tabelas, com linhas e colunas, lembrando muito excel. Como cada tabela vai representar uma entidade do mundo real e as linhas irão representar os registros dos individuais, faz com que a principal característica do banco de dados relacional é a capacidade dele de estabelecer relacionamentos entre tabelas por meio de chaves primárias e chaves estrangeiras, permitindo que os dados fiquem associados, facilitando consultas. (Calanca, 2023)

No backend, tecnologias como o **Spring Boot** têm ganhado relevância por facilitar a criação de aplicações baseadas em microsserviços, além de oferecer integração com protocolos como WebSocket e sistemas de mensageria como Kafka, características essenciais para comunicação em tempo real (Wells, 2016; Newman, 2021). No frontend, frameworks modernos como o **React** se destacam por possibilitar o desenvolvimento de interfaces dinâmicas e reativas, com grande suporte da comunidade e compatibilidade com Progressive Web Apps (PWA) (Grinberg,

2020; Banks, 2017; Leiva, 2021).

2.3.2 Multiplataforma

A necessidade de atingir diferentes dispositivos de forma uniforme levou ao crescimento de tecnologias multiplataforma. Aplicações multiplataforma permitem que um único código-fonte seja executado em diferentes sistemas operacionais, como Android, iOS e Web, reduzindo custos e tempo de desenvolvimento. Ferramentas como **Flutter**, mantido pelo Google, têm se consolidado por oferecer alto desempenho e uma única base de código para múltiplas plataformas (Google, 2023). Além disso, soluções como **Progressive Web Apps (PWA)** permitem levar funcionalidades típicas de aplicativos nativos para navegadores, como funcionamento offline e notificações push (Leiva, 2021). Estudos comparativos demonstram que as ferramentas multiplataforma modernas já atingem desempenho semelhante ao desenvolvimento nativo em diversos cenários (Anwer *et al.*, 2016).

3 TRABALHOS RELACIONADOS

Nesta seção serão discutidos os trabalhos relacionados para o desenvolvimento do sistema, mostrando quais as principais características de cada sistema e como eles se relacionam e contribuem com o desenvolvimento da solução .

3.1 Development of Lasto Virtual Queue Management System

O trabalho proposto por Basgaran e Ibrahim (2023) decorre de que as filas são algo com que todos os indivíduos devem lidar no dia a dia, pois devem esperar longos períodos para serem atendidos em diversos locais públicos, como bancos, hospitais, lojas, farmácias, etc. Devido a isso, o trabalho afirma que um sistema de gestão de filas deve ser desenvolvido para gerir e organizar a prioridade de pessoas que estão a espera de atendimento da forma mais eficiente e eficaz. O mesmo autor ainda justifica que o gerenciamento tem sido uma grande preocupação nos últimos anos, uma vez que as organizações tendem a ficar extremamente lotadas, especialmente durante os períodos de pico.

Dessa forma, o desenvolvimento do sistema de gerenciamento de filas virtuais do Lasto foi pensado para atender um centro de serviços chamado Lasto Computer Service Centre, uma vez que eles focam em oferecer vários tipos de serviços, como manutenção de hardware, recuperação de dados, instalações de software, atualização, reformatação ou redefinição de janelas e compra de itens como laptops, impressoras e carregadores, visto que o sistema proposto pretende criar uma infraestrutura para vender como serviço, atendendo os mais variados propósitos.

Lasto também tem objetivos comparáveis com o deste trabalho, uma vez que conforme Basgaran e Ibrahim (2023) existem três objetivos principais: projetar o Lasto VQMS usando uma abordagem orientada a objetos, desenvolver a aplicação baseado em web e realizar testes de funcionalidade e aceitação do usuário no sistema desenvolvido.

Ainda de acordo com os autores do trabalho Basgaran e Ibrahim (2023), os usuários podem ser categorizados em três grupos, dentre eles: o administrador, funcionários e clientes. O administrador pode fazer login no sistema, gerenciar funcionários, serviços, balcões e clientes e gerenciar filas. Os funcionários também pode fazer login no sistema para gerenciar a fila. Por último, os clientes podem reservar virtualmente o seu token para obter serviços.

Finalmente, espera-se que as funcionalidades propostas pelos autores contribuam

com o desenvolvimento da solução da plataforma web para redução das filas das instituições bancárias, visto que os requisitos funcionais e não funcionais, etapas de desenvolvimento, tipos de usuários e artefatos são similares ao deste trabalho.

3.2 Community Request Queue Management System for Local Government Unit of Cabanatuan City

O trabalho proposto por Himo *et al.* (2022) fala sobre a criação de um sistema de gerenciamento de filas para o governo à pedido da comunidade, visto que as autoridades tinham problemas para administrar o estoque de papel e de caneta, complicações para recuperar registros e agendamentos comprometidos devido a pandemia da COVID-19.

A aplicação desenvolvida pelo trabalho foi baseada no modelo de desenvolvimento conhecido como modelo de cascata, seguindo as seguintes etapas: Analise de requerimentos, design do sistema, desenvolvimento, testes, implantação e manutenção. A forma que eles decidiram adotar para coletar dados e validar o sistema foi através de técnicas de observação e técnicas de entrevistas. Esses dados foram analisados usando o média ponderada. Os dados coletados diziam sobre os seguintes requisitos não funcionais: Funcionalidade, Usabilidade, Eficiência, Portabilidade, Confiabilidade e Manutenção.

Finalmente, espera-se que a avaliação dos requisitos não funcionais propostas pelos autores contribuam com o desenvolvimento da solução da plataforma web para redução das filas das instituições bancárias, visto que também há um interesse de avaliar a satisfação do público.

3.3 Otimização via Internet

Um trabalho relevante que pode ser associado ao presente TCC é o desenvolvido por Júnior (2010), intitulado *Otimização via Internet*. Nesse estudo, o autor explora técnicas de otimização matemática aplicadas em ambientes distribuídos, com ênfase na utilização da Internet como meio para execução de algoritmos de otimização em larga escala. O trabalho foca em estratégias de alocação de recursos e balanceamento de carga em sistemas que demandam tomada de decisão eficiente e contínua, utilizando métodos de otimização linear e não linear em tempo quase real.

Embora o objetivo do trabalho de Júnior (2010) não tenha sido diretamente voltado para o gerenciamento de filas em instituições bancárias, sua contribuição é altamente relevante

para este TCC. Isso se deve ao fato de que a otimização de recursos é um dos principais desafios em sistemas de atendimento ao público, onde a má alocação de recursos pode resultar em aumento significativo dos tempos de espera.

As técnicas descritas no trabalho de Júnior podem ser adaptadas ao contexto do sistema desenvolvido neste TCC, especialmente para: (i) balanceamento dinâmico de carga entre diferentes guichês de atendimento; (ii) previsão de demandas com base em dados históricos para otimizar a distribuição de senhas e recursos; e (iii) melhoria contínua da eficiência do sistema por meio de algoritmos de otimização integrados ao fluxo de atendimento.

Dessa forma, o estudo de Júnior (2010) se apresenta como um referencial teórico complementar, que poderia enriquecer futuras evoluções do sistema proposto neste trabalho ao incorporar técnicas mais avançadas de otimização distribuída.

3.4 Gerenciamento de filas em ambientes de atendimento ao público – Elisa Cristina Loss (2019)

Outro trabalho relevante é o de Loss (2019), que desenvolveu um estudo aprofundado sobre o gerenciamento de filas para atendimento ao público, com foco em instituições como o EBANX e a Central de Estágios da Prefeitura Municipal de Curitiba. O objetivo central do estudo foi analisar e comparar diferentes mecanismos e práticas de gerenciamento de filas, buscando compreender como estes impactam a experiência do usuário e a eficiência operacional.

A autora avaliou, por meio de estudos de caso, métricas fundamentais como tempo médio de espera, tempo de atendimento e taxa de desistência, além de investigar a implementação de soluções tecnológicas e organizacionais para reduzir gargalos no atendimento. O trabalho destacou a importância de um planejamento adequado e do uso de ferramentas digitais para monitorar o fluxo de clientes em tempo real.

A relação deste estudo com o presente trabalho se dá pelo foco em ambientes de atendimento que lidam com grande volume de usuários. Assim como a pesquisa de Loss, o sistema proposto neste TCC busca minimizar o tempo de espera e melhorar a experiência do usuário por meio de um gerenciamento mais inteligente das filas. Enquanto a pesquisa de Loss concentra-se na análise dos processos existentes e em soluções organizacionais, o presente trabalho se propõe a desenvolver uma plataforma tecnológica que permita o acompanhamento remoto das filas e a notificação dos usuários em tempo real, agregando maior flexibilidade e eficiência ao processo.

Dessa forma, o estudo de Loss serve como um importante embasamento teórico e prático, reforçando a relevância da adoção de sistemas de gerenciamento de filas em instituições que dependem de um atendimento ágil e organizado, como bancos e demais serviços públicos.

3.5 Comparativo entre os trabalhos

Quadro 1 – Comparativo entre os trabalhos relacionados e o trabalho atual

Critérios	Basgaran et al. (2023)	Himo et al. (2022)	Ricardo Júnior (2010)	Elisa Loss (2019)	Trabalho atual
É um sistema específico para banco					X
Implantação online da aplicação					X
Foco no gerenciamento de filas em ambientes de atendimento ao público				X	X
Proposta de solução tecnológica própria			X		X
Possibilidade de acompanhamento remoto das filas					X
Avaliação detalhada de métricas (tempo de espera, desistência, etc.)				X	X

Fonte: elaborado pelo autor (2025).

4 METODOLOGIA

O presente trabalho se caracteriza com uma abordagem descritiva e exploratória (quali-quantitativa), construído a partir conhecimento existente, para gerar o “estado da arte”. Nesse sentido, a revisão de literatura realizada no início e de durante o desenvolvimento deste trabalho, demonstrou que o tema escolhido é pouco estudado, com muito ainda a ser sistematizado para solucionar um problema recorrente das filas nas instituições financeiras. Destarte, para efetivar este estudo será realizada uma pesquisa exploratória com abordagem quali-quantitativa. Neste caso, adotar-se-á a estrutura de pesquisa exploratória, com abordagem quali-quantitativa e descritiva, por ser realizado com base em referências teóricas publicadas em artigos e obras similares, além de se configurar em um tipo de investigação, que aponta e discute métodos e sistemas, buscando meios de mostrar os passos necessários para construção de plataforma web para emissão de senhas. (Gil, 2014)

A pesquisa é exploratória, visto que visa desenvolver, esclarecer e proporcionar uma visão geral sobre um determinado fato pouco explorado. É descritiva pois tem como objetivo coletar as opiniões da população sobre o desenvolvimento de uma plataforma web de gerenciamento de filas online para bancos e sua adoção através de formulários. A análise do presente trabalho é quali-quantitativa. Qualitativa porque a análise dos dados nas pesquisas é essencialmente qualitativa, já que o que será aferido são os gostos e opiniões sobre o trabalho.(Gil, 2014) Quantitativa pois as opiniões aferidas serão analisadas. (Silva; Menezes, 2005, p.20) define análise quantitativa como "considera que tudo pode ser quantificável, o que significa traduzir em números opiniões e informações para classificá-las e analisá-las".

4.1 Etapas do desenvolvimento

4.1.1 Planejamento

Na etapa do planejamento será definido o projeto a ser desenvolvido, com a identificação do problema, qual o escopo das atividades, seus objetivos e justificativa. Nesse primeiro momento, foi elaborado o projeto com a identificação do problema de geração de filas nas instituições bancárias.

Com isso, foi definido seu escopo: o desenvolvimento de uma plataforma web para gerenciamento de filas onlines para instituições bancárias, de tal forma que, com essa aplicação

as pessoas possam retirar senhas onlines, podendo assim, aproveitar melhor seu tempo. Em seguida, foram definidos os objetivos gerais e específicos para desenvolver uma solução para o problema apresentado.

Finalmente, a justificativa foi apresentada quando foi mostrado que apesar dos bancos brasileiros serem os que mais investem em tecnologia no mundo, ainda não há investimento que possa melhorar a qualidade de atendimento dos usuários.

4.1.2 Design

Nesta etapa ocorrerá a análise da problemática, gerando, a partir dela, os diagramas, os designs e a definição dos requisitos. Ou seja, o que precisa ser feito para criar a aplicação, dividindo as funcionalidades em história do usuário, que servirão para guiar o desenvolvimento e gerar o documento de requisitos.

Após a definição de todas as história do usuário, iniciará a etapa de design dos diagramas, na qual será analisado como serão divididas as responsabilidades de classe, gerando o diagrama de classe, como serão os fluxos principais e alternativos de funcionamento do sistema, resultando no Diagrama de Raia, além de quantos e quem serão os usuários e quais funcionalidades terão acesso, originando o Diagrama de Caso de Uso.

Após a conclusão de cada um desses diagramas, estarão definidos quais e quantos serão os usuários. Assim, após a conclusão dessas etapas, será possível iniciar o frontend, analisando e definindo telas que correspondam ao fluxo e às necessidades de cada usuário, gerando um wireframe.

4.1.3 Desenvolvimento

Na etapa de desenvolvimento serão usados todos os artefatos já gerados para guiar o início do desenvolvimento. Nesta etapa, irá ocorrer a programação do sistema usando os diagramas e os designs como norte e a implementação de testes automatizados para validar as funcionalidades. Assim, no final desta etapa, poderá ser visualizada a primeira versão do código do sistema, os casos de testes e o protótipo.

O desenvolvimento começará pelo frontend, pois costuma demandar mais tempo quando comparado ao do backend, devido à montagem das telas inteiras a partir do zero, além de haver necessidade de fazer pequenos ajustes nos detalhes.

A linguagem usada para desenvolvimento foi o Javascript com o framework React,

devido à maior familiaridade com essas ferramentas. Todo o desenvolvimento do frontend será baseado no design das telas feitos na etapa anterior, com resalva de pequenas mudanças que podem vir acontecer caso o que esteja pensado seja muito complicado ou caso haja uma forma melhor de se fazer.

O desenvolvimento do backend usará a linguagem Java com o framework Spring, visto a maior familiaridade com essas ferramentas. Devido ao problema aqui atacado, será usado Kafka para criação de tópicos, consumers e producers, usados para gerenciar as filas online.

Para a parte dos testes automatizados,a ferramenta usada JUnit, visto que é a biblioteca mais utilizada no mercado para fazer testes unitários em aplicações web/java. Todos os cenários, até então planejados, terão seus testes automatizados em 3 cenários: o caso pra qual foi planejada, o limite do caso qual foi planejada e casos de erros a qual ela não deveria aceitar/responder.

Para banco de dados será usado o banco relacional Postgres, também devido pela familiaridade.

4.1.4 Protótipo

Esta etapa servirá para detectar os possíveis erros inesperados, como: funcionalidades, lógica e design, durante a aplicação do sistema, por meio do teste em 3 cenários: funcionar conforme o planejado, no limite do caso para o qual foi planejado e casos de erros a qual ela não deveria aceitar/responder.

4.1.5 Implantação do sistema

A implantação do sistema foi realizada em ambiente de nuvem, por meio da plataforma Google Cloud, com o objetivo de disponibilizar a aplicação em um ambiente público acessível. Embora não tenha sido conduzido um processo formal de avaliação com usuários externos, o sistema foi implantado com dados fictícios, permitindo simulações completas de uso.

Durante o período em que permaneceu disponível, a plataforma permitiu a realização de cadastros, retirada de senhas e acompanhamento da fila em tempo real, incluindo o tratamento de prioridades e a atualização contínua do status das senhas. O objetivo principal dessa implantação foi validar, em ambiente real, o funcionamento dos fluxos implementados, a estabilidade da aplicação, e a correta integração entre os serviços utilizados, como WebSocket, Kafka e banco de dados.

Com isso, foi possível assegurar que a aplicação está tecnicamente pronta para uso, mesmo que sua validação por usuários finais tenha sido postergada para trabalhos futuros.

5 DESENVOLVIMENTO DA PLATAFORMA WEB PARA GERENCIAMENTO DE FILA ONLINE PARA BANCOS

A execução do projeto seguiu uma abordagem de desenvolvimento incremental, com fases bem definidas e entregáveis concretos em cada etapa, conforme ilustrado na Tabela 2.

Quadro 2 – Fases de prototipação

Etapa	**Atividade**	**Saída**
Planejamento	Proposta do projeto Identificação do problema, escopo e objetivos	Definição do projeto
Design	Análise das informações Design das telas	Diagrama de Caso de Uso Diagrama de Raia Diagrama de Classe Documento de Requisitos Arquitetura do Sistema Interface do Usuário
Desenvolvimento	Programação Testes do sistema Correção de bugs	Código do sistema Casos de teste Protótipo
Protótipo	Detecção de erros e ajustes	Versão final
Implantação	Implantação online do sistema	Teste da aplicação online

Fonte: elaborado pelo autor (2025).

A fase inicial de *planejamento*, já abordada anteriormente neste trabalho, foi responsável pela delimitação do escopo, definição dos objetivos e identificação do problema a ser resolvido. Em seguida, deu-se início à etapa de *design*, que teve como objetivo estruturar tecnicamente a solução proposta.

Durante essa etapa, foram produzidos seis entregáveis fundamentais: (i) Documento de Arquitetura do Sistema, (ii) Documento de Requisitos , (iii) Diagrama de Classe, (iv) Diagrama de Caso de Uso, (v) Diagrama de Raia e (vi) Interface do Usuário. É importante destacar que alguns desses artefatos foram refinados ao longo do desenvolvimento, como foi o caso do Diagrama de Classe, que inicialmente contemplava apenas entidades básicas, mas foi estendido conforme novas funcionalidades foram sendo incorporadas ao sistema.

Cada artefato cumpriu uma função específica no direcionamento do projeto. O Diagrama de Raia mapeou os dois principais fluxos operacionais (Retirada de Senha e Atendimento), enquanto o Diagrama de Caso de Uso explicitou, de forma gráfica, as ações disponíveis para cada ator do sistema. O Documento de Arquitetura sintetizou as principais decisões técnicas e tecnologias utilizadas, conforme descrito na Seção 5.1.

5.1 Documento de Arquitetura do Sistema

A arquitetura da plataforma foi concebida com o objetivo de garantir comunicação em tempo real, alta disponibilidade, escalabilidade horizontal e consistência transacional. Para atender a esses requisitos, foi adotado um modelo baseado em aplicação distribuída, com uso de tecnologias consolidadas no mercado, como Java com Spring Boot, Apache Kafka e WebSocket.

5.1.1 Visão Geral da Arquitetura

O sistema de atendimento por senhas consiste em uma aplicação distribuída, desenvolvida com a linguagem Java (versão 17), utilizando o framework Spring Boot. Seu principal objetivo é permitir que usuários realizem a retirada de senhas remotamente, acompanhem o andamento da fila e sejam chamados em tempo real por atendentes, com base em regras de prioridade. Para viabilizar essa comunicação em tempo real, são utilizados WebSocket e Apache Kafka como tecnologias de mensageria assíncrona.

5.1.2 Estilo Arquitetural

Foi adotado o estilo arquitetural em camadas (*Layered Architecture*), dividido da seguinte forma:

- **Apresentação:** camada responsável por expor os recursos REST e as conexões WebSocket.
- **Serviço:** camada intermediária que encapsula a lógica de negócio.
- **Persistência:** camada dedicada à comunicação com o banco de dados, utilizando os repositórios do Spring Data JPA.
- **Integração:** camada responsável por publicar e consumir eventos via Apache Kafka.

5.1.3 Componentes Principais

5.1.3.1 Backend

O backend da aplicação foi desenvolvido utilizando o framework Spring Boot 3, com as seguintes responsabilidades:

- **REST Controllers:** fornecem os pontos de entrada para o frontend, permitindo operações como retirada de senha, visualização do estado da fila e comandos do atendente.
- **WebSocket via STOMP:** utilizado para notificações em tempo real aos clientes conectados,

sem necessidade de polling.

- **Services:** encapsulam a lógica de negócio da aplicação, como regras de fila e prioridade.
- **Repositórios JPA:** abstraem o acesso aos dados persistidos no banco de dados.
- **Apache Kafka:** utilizado como mecanismo de mensageria assíncrona entre serviços e para atualização de estado da fila.

5.1.3.2 Banco de Dados

A persistência dos dados é feita utilizando o sistema gerenciador de banco de dados PostgreSQL 13, com estrutura relacional. As principais tabelas da aplicação são:

- `usuario`: armazena os dados de usuários cadastrados na aplicação.
- `senha`: registra as senhas geradas, seu tipo (normal ou prioritária), status e timestamps.

5.1.3.3 Mensageria

A comunicação assíncrona e reativa do sistema é realizada com o uso do Apache Kafka. Os tópicos definidos no sistema são:

- `fila.normal`: tópico utilizado para o gerenciamento da fila geral de senhas.
- `fila.atualizacao`: tópico responsável por transmitir atualizações do estado da fila para os clientes via WebSocket.

5.1.3.4 Frontend

Embora não esteja incluído neste documento, o frontend foi desenvolvido com ReactJS. Ele consome os endpoints REST fornecidos pelo backend e se mantém conectado por meio do canal WebSocket, recebendo atualizações em tempo real do estado da fila e notificações sobre senhas chamadas.

5.1.4 Fluxos Arquiteturais Importantes

A seguir, são descritos os dois principais fluxos funcionais do sistema:

5.1.4.1 Retirada de Senha

1. O usuário realiza uma solicitação de senha por meio de um endpoint REST.
2. O `Controller` aciona o `Service`, que valida a requisição e gera a nova senha.

3. A senha é persistida no banco de dados, publicada no tópico Kafka correspondente e, em seguida, uma notificação é enviada via WebSocket ao frontend.

5.1.4.2 Atendimento

1. O atendente aciona o endpoint para chamar a próxima senha da fila.
2. O `Service` finaliza o atendimento anterior (caso exista).
3. Uma nova senha é buscada e marcada como em atendimento.
4. A atualização é enviada para os tópicos Kafka e, consequentemente, repassada ao frontend via WebSocket.

5.1.5 Padrões Arquiteturais Utilizados

A arquitetura do sistema incorporou os seguintes padrões de projeto:

- **DTO (Data Transfer Object):** para encapsulamento e transporte de dados entre camadas.
- **Service Layer:** separação da lógica de negócio da lógica de controle.
- **Repository Pattern:** com uso do Spring Data JPA para abstração da camada de persistência.
- **Observer:** via WebSocket, permitindo comunicação reativa com os clientes.
- **Producer/Consumer:** para publicação e consumo de mensagens no Apache Kafka.

5.1.6 Tecnologias e Ferramentas

- Java 17
- Spring Boot
- Spring Web / WebSocket / Kafka / JPA
- PostgreSQL 13
- Apache Kafka
- Docker / Docker Compose
- Lombok

5.1.7 Requisitos Arquiteturais

A arquitetura proposta foi construída de forma a atender os seguintes requisitos não funcionais:

- **Escalabilidade horizontal:** obtida por meio do uso de Kafka e WebSocket desacoplados.
- **Baixa latência:** essencial para atualizações em tempo real via protocolo STOMP.
- **Alta disponibilidade:** assegurada com uso de containers Docker e orquestração facilitada.
- **Consistência de dados:** garantida por meio do uso de transações com `@Transactional` do Spring.

5.1.8 Considerações Finais

A arquitetura apresentada reflete o estado atual do sistema, oferecendo uma base sólida para futuras expansões, tais como a inclusão de dashboards analíticos, chamadas por voz ou módulos de estatísticas avançadas. A estrutura modular e orientada a eventos favorece a manutenção, extensibilidade e a adaptação a diferentes contextos de atendimento.

5.2 Documento de Requisitos do Sistema

Esta seção apresenta os requisitos funcionais e não funcionais, as regras de negócio, as histórias de usuário e os componentes envolvidos no desenvolvimento da plataforma de atendimento por senhas. O objetivo é assegurar que o sistema atenda às necessidades operacionais e de usabilidade esperadas, com base em critérios técnicos e experiências reais de uso.

5.2.1 Visão Geral

O sistema de atendimento por senhas é uma aplicação web voltada para a organização de filas em ambientes como agências bancárias, repartições públicas ou instituições de serviços presenciais. A proposta central é permitir que usuários realizem a retirada de senhas (normais ou prioritárias) de forma remota, acompanhem sua posição na fila e recebam notificações em tempo real sobre seu atendimento. Os atendentes, por sua vez, têm acesso a funcionalidades para chamar, finalizar ou descartar senhas, conforme o andamento do fluxo de atendimento.

A comunicação em tempo real é viabilizada por meio de WebSocket e Apache Kafka, permitindo a integração de eventos e notificações em tempo real com o frontend.

5.2.2 Requisitos Funcionais (RF)

A seguir, são listados os requisitos funcionais definidos para o sistema:

- **RF01:** O sistema deve permitir que o usuário retire uma senha informando o tipo (NOR-

MAL ou PRIORITÁRIA).

- **RF02:** O sistema deve gerar um código único para cada senha emitida.
- **RF03:** O sistema deve armazenar a data e o horário da retirada da senha.
- **RF04:** O sistema deve impedir que um usuário retire mais de uma senha ativa no mesmo dia.
- **RF05:** O sistema deve permitir que um atendente chame a próxima senha, respeitando as regras de prioridade configuradas (exemplo: duas senhas normais seguidas por uma prioritária).
- **RF06:** O sistema deve atualizar dinamicamente a posição de todas as senhas aguardando atendimento após cada alteração de estado.
- **RF07:** O sistema deve permitir que o atendente registre a desistência de uma senha.
- **RF08:** O sistema deve permitir finalizar um atendimento em andamento.
- **RF09:** O sistema deve enviar via WebSocket atualizações em tempo real da fila (senha atual, próxima senha, tempo estimado de espera).
- **RF10:** O sistema deve publicar eventos no Apache Kafka para integração com outros serviços.
- **RF11:** O sistema deve disponibilizar um histórico de senhas finalizadas ou desistidas.
- **RF12:** O sistema deve expor endpoints REST para operações como retirada de senha, consulta de fila e chamadas de atendimento.

5.2.3 Requisitos Não Funcionais (RNF)

Os requisitos não funcionais determinam os atributos de qualidade e as restrições técnicas da aplicação:

- **RNF01:** O sistema deve garantir consistência transacional nas operações de chamada e finalização de senhas.
- **RNF02:** O sistema deve utilizar WebSocket para notificações em tempo real aos usuários.
- **RNF03:** O sistema deve utilizar Apache Kafka para comunicação assíncrona entre módulos internos e externos.
- **RNF04:** O sistema deve ser implementado na linguagem Java, utilizando o framework Spring Boot.
- **RNF05:** O sistema deve persistir os dados em um banco de dados relacional.
- **RNF06:** As mensagens publicadas no Apache Kafka devem ser serializadas em formato

JSON.

5.2.4 Regras de Negócio (RN)

As regras de negócio orientam o funcionamento interno da fila, priorizando justiça, previsibilidade e coerência lógica:

- **RN01:** Senhas prioritárias devem ser atendidas intercaladamente, após cada duas senhas normais.
- **RN02:** Cada senha deve possuir uma estimativa de espera baseada em sua posição na fila (por exemplo, 5 minutos por posição).
- **RN03:** A ordem de atendimento deve seguir a lógica FIFO (*First In, First Out*) dentro de cada tipo de prioridade.
- **RN04:** O sistema deve bloquear a retirada de uma nova senha caso o usuário já possua uma senha ativa.
- **RN05:** O sistema deve atualizar dinamicamente a posição de todas as senhas na fila após qualquer evento que altere seu estado (retirada, chamada, finalização ou desistência).

5.2.5 Componentes Envolvidos

Os principais componentes arquiteturais utilizados na implementação do sistema foram:

- **Backend REST API:** Java 17 com Spring Boot, utilizando Repositórios JPA e Services para encapsular a lógica de negócio.
- **Mensageria:** Apache Kafka com os seguintes tópicos: `fila.normal`, `fila.prioritaria` e `fila.atualizacao`.
- **WebSocket:** Canal de notificação: `/topic/fila/atualizacao`.
- **Banco de Dados:** Tabelas principais: `senha` e `usuario`.

5.2.6 Histórias de Usuário

Para auxiliar na elicitação dos requisitos, foram elaboradas as seguintes histórias de usuário, separadas por tipo de ator:

Usuário

- **US01 - Retirada de Senha:** Como usuário, desejo retirar uma senha (NORMAL ou PRIORITÁRIA), para aguardar atendimento com organização.
- **US02 - Impedir Retirada Repetida:** Como usuário, desejo ser impedido de retirar outra senha no mesmo dia se já houver uma ativa, para evitar duplicidade.
- **US03 - Acompanhar Fila:** Como usuário, desejo visualizar minha posição na fila e o tempo estimado de espera.
- **US04 - Ser Notificado em Tempo Real:** Como usuário, desejo receber notificações imediatas quando minha senha for chamada.

Atendente

- **US05 - Chamar Próxima Senha:** Como atendente, desejo chamar a próxima senha, respeitando as regras de prioridade.
- **US06 - Finalizar Atendimento:** Como atendente, desejo finalizar um atendimento, atualizando o status da fila.
- **US07 - Registrar Desistência:** Como atendente, desejo registrar a desistência de uma senha para garantir o correto andamento da fila.

Sistema

- **US08 - Notificar Atualização de Fila:** Como sistema, desejo enviar atualizações via WebSocket para todos os usuários conectados.
- **US09 - Enviar Mensagens ao Kafka:** Como sistema, desejo enviar eventos relevantes ao Kafka, para integração com outros serviços.
- **US10 - Estimar Tempo de Espera:** Como sistema, desejo calcular e exibir a estimativa de espera para cada senha.

5.2.7 Futuras Expansões

Embora não façam parte do escopo atual, algumas possibilidades de expansão do sistema foram identificadas:

- Implementação de um módulo de autoatendimento por totem.
- Desenvolvimento de dashboards com monitoramento em tempo real.

– Geração de relatórios com tempo médio de atendimento por dia.
– Integração com sistemas de chamadas por voz.

5.3 Modelagem UML do Sistema

A modelagem do sistema foi conduzida com base na linguagem UML, visando representar de forma clara a estrutura estática e os principais fluxos de interação entre usuários e componentes da aplicação. Nesta seção são apresentados o Diagrama de Classes, o Diagrama de Casos de Uso e os Diagramas de Raia, que documentam formalmente a lógica do sistema e sua organização interna.

5.3.1 Diagrama de Classes

O Diagrama de Classes tem como objetivo representar a estrutura estática do sistema, detalhando suas classes, atributos, métodos e relacionamentos. Nas Figuras 13 a 18, observa-se que o sistema é organizado de forma modular, com separação clara entre camadas de controle, serviço, domínio e repositório.

Entre os destaques do diagrama, temos:

– A classe `Senha` representa a entidade principal do sistema, associada às classes `Usuario`, `PrioridadeSenha` e `StatusSenha`.
– As classes `AtendimentoService`, `SenhaService`, `UsuarioService` e `MediaAtendimentoDiaServ` encapsulam a lógica de negócio, respeitando o padrão Service Layer.
– Controllers específicos (`SenhaController`, `UsuarioController`, `AtendimentoController`) expõem os endpoints REST e orquestram os fluxos de atendimento.
– A mensageria é manipulada por `KafkaConfig` e `KafkaMessage`, responsáveis pela serialização e publicação de mensagens em tópicos Kafka.
– A classe `WebSocketService` realiza o envio de notificações em tempo real para os clientes conectados.

A modelagem favorece coesão, baixo acoplamento e facilita a manutenção evolutiva do sistema.

O Diagrama de Classes, que apresenta a estrutura das entidades do sistema, suas relações e responsabilidades, encontra-se no Apêndice A. Esse diagrama serviu como base para o desenvolvimento das camadas de domínio, serviço e persistência.

5.3.2 Diagrama de Casos de Uso

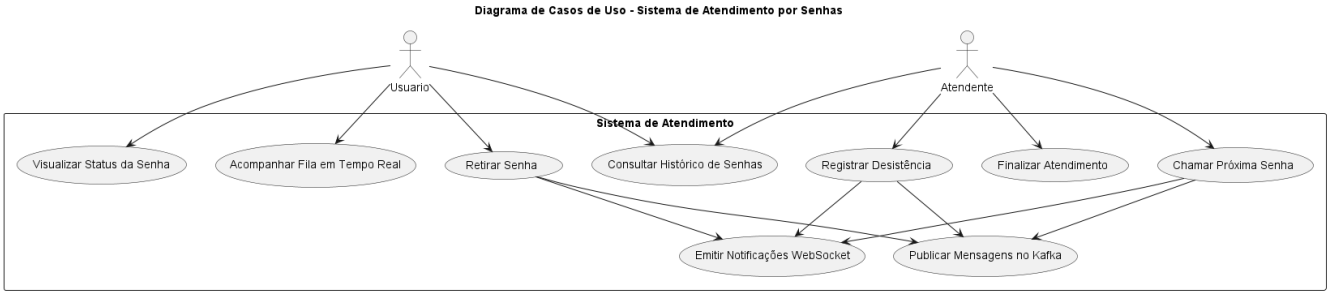
O Diagrama de Casos de Uso, apresentado na Figura 1, ilustra as principais funcionalidades disponíveis para os dois atores do sistema: **Usuário** e **Atendente**.

Os casos de uso foram definidos conforme os requisitos funcionais identificados:

- O **Usuário** pode retirar senhas, consultar seu status, visualizar sua posição na fila em tempo real e acessar seu histórico.
- O **Atendente** pode chamar a próxima senha, finalizar atendimentos ou registrar desistências, além de consultar o histórico completo.
- Casos internos como emissão de notificações por WebSocket e publicação de eventos no Kafka são acionados como consequência de eventos principais.

A organização dos casos de uso reforça a centralidade do fluxo de atendimento e a automação da fila via eventos assíncronos.

Figura 1 – Diagrama de Casos de Uso do Sistema



Fonte: elaborado pelo autor (2025).

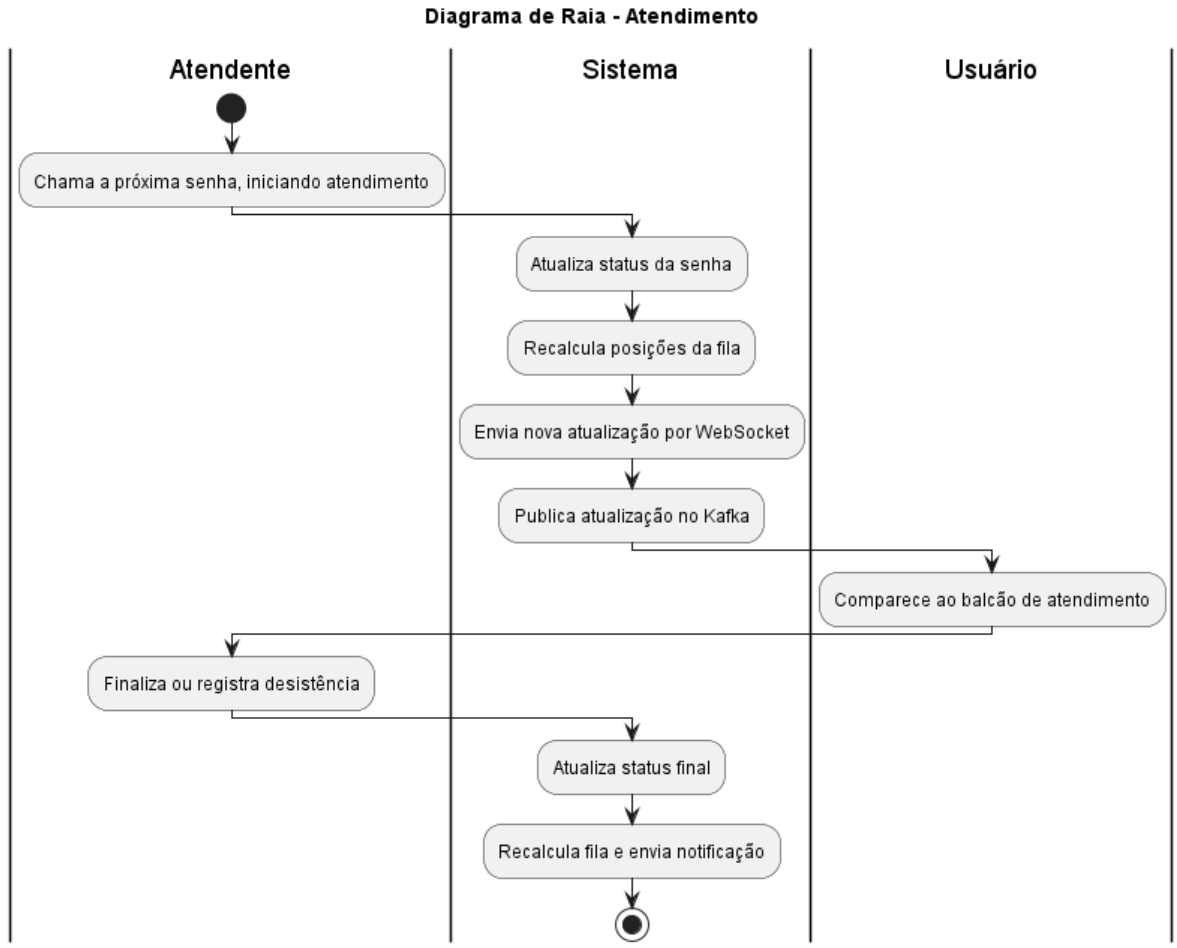
5.3.3 Design

Os Diagramas de Raia (*Swimlanes*) descrevem os fluxos operacionais do sistema, divididos em trilhas que representam os principais atores: Usuário, Sistema e Atendente. Dois fluxos distintos são abordados:

Fluxo 1 – Atendimento

Na Figura 2, o processo de atendimento é detalhado desde a chamada da próxima senha até a finalização ou desistência. O Sistema atualiza o status, reorganiza a fila, envia notificações via WebSocket e publica atualizações nos tópicos Kafka, enquanto o usuário comparece ao atendimento.

Figura 2 – Diagrama de Raia – Processo de Atendimento

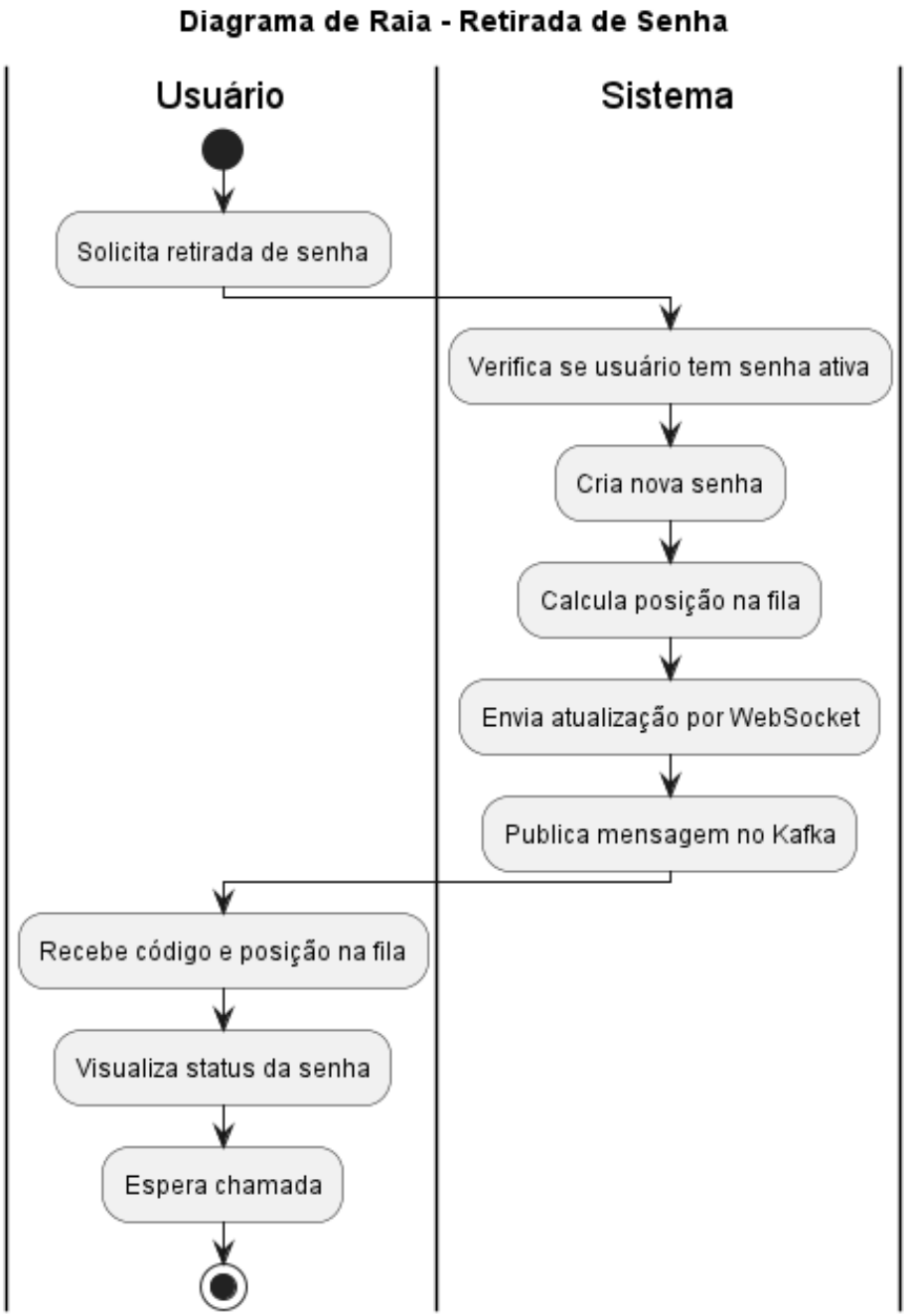


Fonte: elaborado pelo autor (2025).

Fluxo 2 – Retirada de Senha

A Figura 3 ilustra o processo de retirada de senha. Após o pedido do usuário, o sistema verifica se já existe uma senha ativa, cria a nova senha com posição na fila, envia notificações e publica no Kafka. O usuário, por fim, recebe o código da senha e acompanha seu status.

Figura 3 – Diagrama de Raia – Retirada de Senha



Fonte: elaborado pelo autor (2025).

5.4 Design da Interface do Usuário

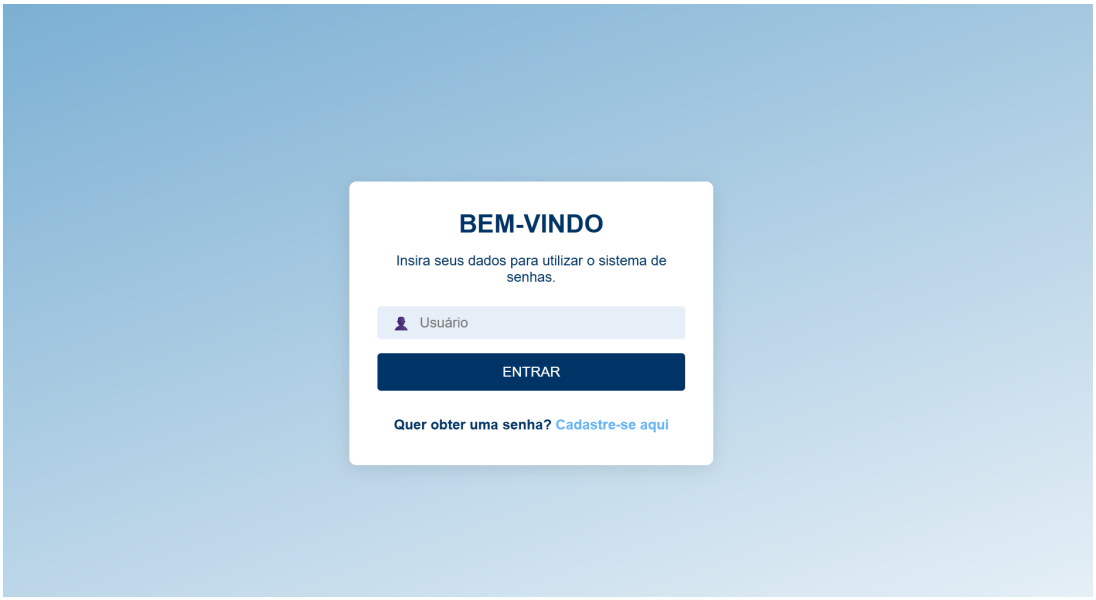
Para o design das interfaces do sistema, foi utilizada a ferramenta **Figma**, uma das mais populares atualmente para a prototipação de interfaces de usuário. A criação do layout contou com a colaboração de duas colegas com experiência prévia na área de design de interfaces, as quais contribuíram com sugestões e esboços iniciais. A partir dos dois protótipos

desenvolvidos por elas, foram feitas adaptações e ajustes necessários para alinhar o design às necessidades funcionais do sistema desenvolvido neste trabalho.

O foco principal do design foi a **usabilidade**, priorizando uma experiência intuitiva, limpa, acessível e responsiva para o usuário. Todas as telas voltadas para o cliente foram construídas com atenção à responsividade, permitindo seu uso tanto em navegadores de computador quanto em dispositivos móveis. Já a tela do atendente, voltada para operadores em um ambiente fixo, foi projetada para ser utilizada exclusivamente em desktops.

A seguir, são apresentadas e descritas as principais telas do sistema, com destaque para os seus fluxos e funcionalidades:

Figura 4 – Tela de Login do Sistema - Desktop



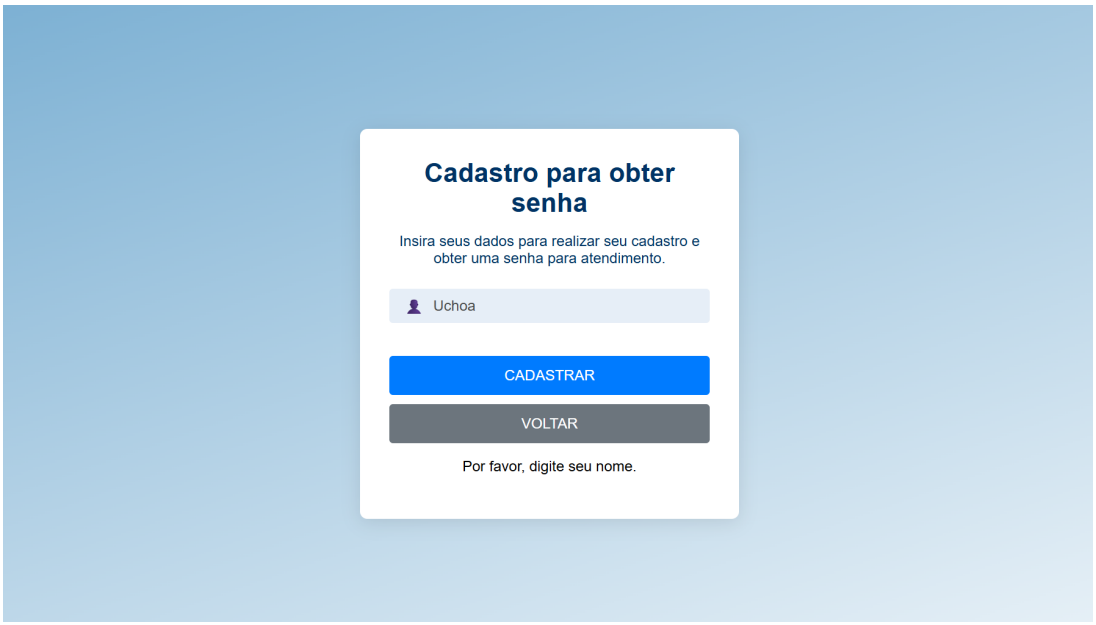
Fonte: elaborado pelo autor (2025).

Figura 5 – Tela de Login do Sistema - Mobile

Fonte: elaborado pelo autor (2025).

A **tela de login** é o ponto de entrada do sistema. O usuário informa seu nome de usuário para tentar acesso ao sistema. Caso o nome de usuário esteja cadastrado, o sistema permite o acesso à funcionalidade de retirada de senha. Caso contrário, é exibida uma mensagem informando que o usuário ainda não está registrado.

Figura 6 – Tela de Cadastro de Usuário



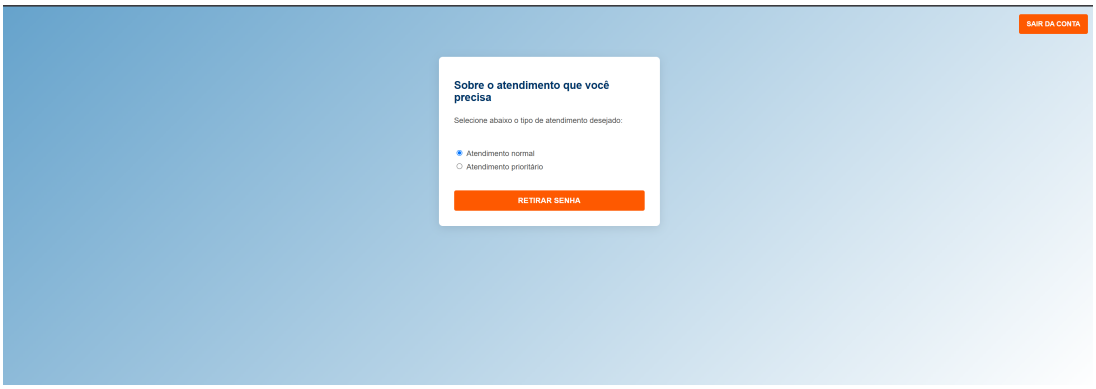
Fonte: elaborado pelo autor (2025).

Figura 7 – Tela de Login do Sistema - Mobile

Fonte: elaborado pelo autor (2025).

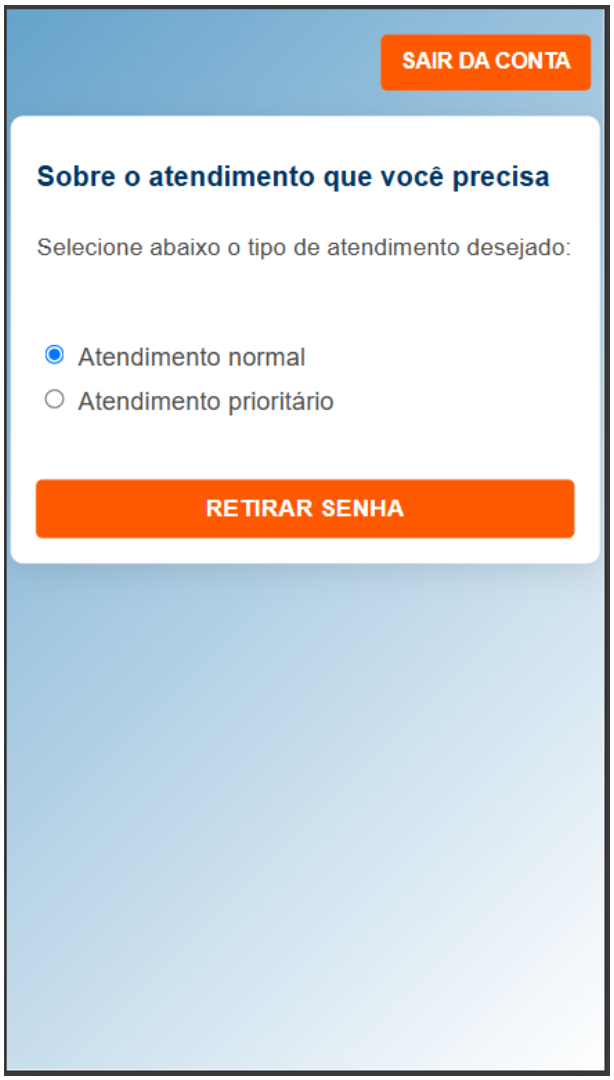
A **tela de cadastro** é acessada por meio de um botão presente na tela de login. Nela, o usuário pode se cadastrar informando um nome de usuário único. Caso o nome informado já esteja sendo utilizado, uma mensagem de erro é exibida. Após o cadastro bem-sucedido, o usuário é redirecionado automaticamente para a tela de login.

Figura 8 – Tela de Retirada de Senha



Fonte: elaborado pelo autor (2025).

Figura 9 – Tela de Login do Sistema - Mobile

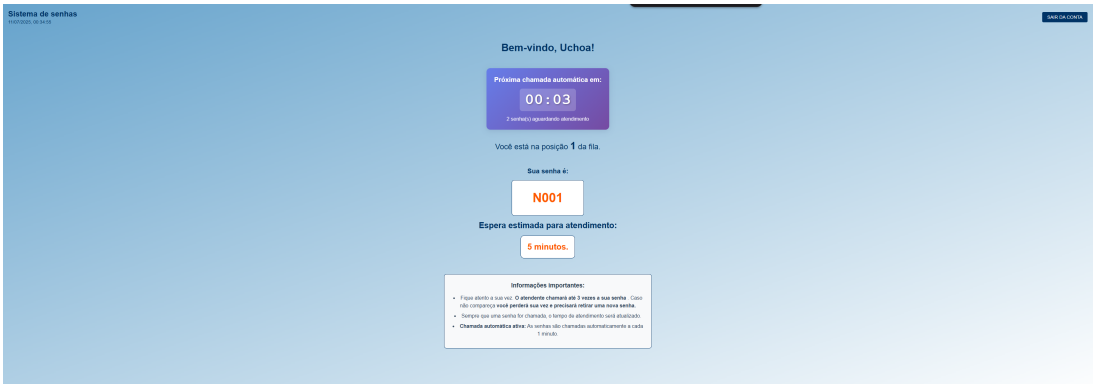


Fonte: elaborado pelo autor (2025).

A **tela de retirada de senha** é exibida após o login ser realizado, desde que o usuário não possua uma senha ativa no sistema. Caso ele já possua uma senha em atendimento ou

em espera, é redirecionado automaticamente para a tela de status. Nessa tela, o cliente pode escolher entre dois tipos de senha: *normal* ou *prioritária*. Embora não haja verificação real para a elegibilidade de prioridade (por tratar-se de um protótipo com dados fictícios), a lógica de atendimento é mantida: para cada duas senhas normais chamadas, a próxima chamada será uma senha prioritária, se disponível.

Figura 10 – Tela de Acompanhamento de Status da Senha



Fonte: elaborado pelo autor (2025).

Figura 11 – Tela de Acompanhamento de Status da Senha

Fonte: elaborado pelo autor (2025).

A **tela de status da senha** é uma das principais telas para o usuário final. Nela, o cliente pode acompanhar em tempo real o progresso da fila, o tempo estimado de espera e a posição de sua senha. Essas informações são atualizadas dinamicamente através de um canal WebSocket. A cada novo atendimento, o sistema emite um som chamando a senha e exibe uma notificação em destaque para o usuário chamado. As informações principais, como o código da senha e o tempo restante, são destacadas com o uso de cores e tamanhos de fonte maiores, garantindo uma rápida assimilação visual.

Figura 12 – Tela do Atendente para Gerenciamento de Atendimentos

Fonte: elaborado pelo autor (2025).

A **tela do atendente** é voltada para o operador responsável pelos atendimentos. Nela, o atendente pode acionar a próxima senha da fila, finalizar o atendimento anterior, visualizar o histórico de senhas já chamadas, e registrar uma desistência, caso o usuário não compareça. Essa interface foi pensada para uso exclusivo em desktops, com foco na eficiência e controle operacional. Ela não possui design responsivo, pois será utilizada em um ambiente controlado (como balcões fixos de atendimento).

Padrões Visuais e Feedbacks

Embora não tenha sido seguido um padrão formal de design (como Material Design ou Bootstrap), foi definida uma paleta de cores própria, com tons neutros para facilitar a leitura e realces em vermelho, azul e verde para destacar ações importantes. As informações críticas ao usuário são apresentadas em negrito ou com fonte aumentada, e há mensagens de erro e carregamento nas principais interações.

Responsividade e Acessibilidade

Todas as interfaces destinadas ao cliente final foram desenvolvidas com atenção à responsividade, permitindo o uso em diferentes dispositivos e tamanhos de tela. Elementos como botões, campos e mensagens são redimensionados conforme a resolução, mantendo a usabilidade tanto em celulares quanto em computadores.

Integração com Casos de Uso

As telas da aplicação foram projetadas de forma a refletirem diretamente os casos de uso definidos na fase de design do sistema, garantindo uma correspondência precisa entre os requisitos funcionais e a interface apresentada ao usuário. Cada funcionalidade descrita nos casos de uso — como a retirada de senha, a visualização do status, o acompanhamento da fila em tempo real e a chamada da próxima senha — foi representada visualmente por meio de interfaces específicas, assegurando que os fluxos de interação fossem coerentes e intuitivos. O percurso do usuário dentro da aplicação, desde o login até o atendimento final, foi cuidadosamente alinhado com os cenários previstos, permitindo uma navegação fluida e lógica. Essa aderência entre o planejamento e a implementação demonstra o cuidado na transição entre a análise dos requisitos e a concretização da solução, comprovando que os casos de uso levantados foram eficazmente traduzidos em funcionalidades reais e acessíveis na plataforma desenvolvida.

CONCLUSÃO

Este trabalho teve como objetivo desenvolver uma aplicação web voltada para o gerenciamento de filas de atendimento por senha, com foco em ambientes como agências bancárias. A proposta partiu da constatação de que os modelos tradicionais de gerenciamento de atendimento presencial geram insatisfação, filas físicas, desorganização e sensação de tempo perdido por parte dos usuários. Como resposta a esse problema, foi idealizada uma plataforma capaz de permitir a retirada online de senhas, com acompanhamento em tempo real, priorização de atendimentos e estimativas de espera baseadas em dados dinâmicos.

Durante o desenvolvimento do sistema, buscou-se adotar uma abordagem incremental e iterativa, fundamentada em boas práticas de engenharia de software. O projeto foi cuidadosamente estruturado com base em diagramas de caso de uso, classe e raia, além de documentação de requisitos e arquitetura, o que favoreceu a organização das funcionalidades, divisão de responsabilidades e clareza de implementação.

A aplicação resultante conta com um frontend desenvolvido em ReactJS e um backend em Java com Spring Boot, fazendo uso de WebSocket e Apache Kafka para garantir atualizações em tempo real sobre o status das filas. Dentre os principais desafios enfrentados, destacam-se a configuração e integração de tecnologias de mensageria, bem como a implementação de uma fila dinâmica que respeite regras de prioridade. O uso do protocolo STOMP sobre WebSocket permitiu uma comunicação assíncrona eficaz com os clientes conectados, garantindo reatividade e fluidez na experiência de uso.

As interfaces foram projetadas com foco na usabilidade, acessibilidade e responsividade. Foram criadas experiências distintas para o cliente e para o atendente, respeitando os diferentes papéis e fluxos de interação no sistema. O protótipo inicial passou por refinamentos com base em testes manuais e observações práticas, levando a melhorias importantes na lógica de fila, tratamento de eventos e exibição das informações.

Com base na proposta inicial deste trabalho, é possível afirmar que todos os objetivos estabelecidos foram plenamente alcançados. Primeiramente, foi desenvolvida uma plataforma web funcional que permite aos usuários realizarem a retirada de senhas de forma virtual, com uma interface intuitiva e acessível tanto por dispositivos móveis quanto por computadores. Em seguida, foram implementados mecanismos de gerenciamento de fila dinâmica, com uma lógica de prioridade que garante a alternância entre senhas normais e prioritárias, seguindo a regra de que, a cada duas senhas normais atendidas, uma prioritária será chamada, caso exista. Por

fim, foi implementado um sistema de estimativa de tempo de espera, que calcula e exibe ao usuário uma previsão baseada no ritmo atual dos atendimentos. Essas funcionalidades garantem ao usuário uma experiência completa, previsível e transparente ao longo de todo o processo de atendimento, atingindo com sucesso os objetivos traçados.

Como trabalhos futuros, sugere-se a inclusão de um módulo de avaliação de satisfação, a integração com sistemas físicos de triagem e painéis de senha, autenticação por CPF ou QR Code, e a ampliação da camada de segurança. Também seria possível enriquecer a análise histórica com dados estatísticos mais robustos, o que poderia gerar previsões ainda mais precisas.

Conclui-se, portanto, que a plataforma desenvolvida contribui significativamente para a modernização dos processos de atendimento, oferecendo uma solução acessível, eficiente e escalável para ambientes que ainda operam com filas físicas e processos manuais.

REFERÊNCIAS

ALEXA, L.; AVASILCAI, S. The requirement elicitation process of designing a collaborative environment: The cre@tive.biz case. In: **MATEC Web of Conferences**. [*S. l.*]: EDP Sciences, 2018. v. 184, p. 04010. Acesso em: 16 abr. 2024.

ANWER, M. *et al.* Comparative analysis of cross-platform mobile development tools. In: **Proceedings of the 2016 International Conference on Software Engineering**. ACM, 2016. Disponível em: https://dl.acm.org/doi/10.1145/2884781.2884804. Acesso em: 25 jul. 2025.

ARAúJO, C. A. S.; CARNEIRO, T. C. J. Filas nos bancos: por que a tecnologia da informação não resolve? a percepção dos gerentes sobre causas e prováveis soluções. **REAd- Revista Eletrônica de Administração**, Universidade Federal do Rio Grande do Sul, v. 14, n. 3, p. 569–593, 2008. Disponível em: https://seer.ufrgs.br/read/article/view/40059. Acesso em: 10 mar. 2023.

BANKS, L. **React for Real**: Front-end code, untangled. Pragmatic Bookshelf, 2017. Disponível em: https://pragprog.com/titles/lfreact/react-for-real/. Acesso em: 23 jul. 2025.

BASGARAN, P. S.; IBRAHIM, N. Design and development of lasto virtual queue management system. **Applied Information Technology And Computer Science**, v. 4, n. 1, p. 916–934, 2023. Acesso em: 12 abr. 2024.

CALANCA, P. **SQL e NoSQL**: trabalhando com bancos relacionais e não relacionais. Alura, 2023. Disponível em: https://www.alura.com.br/artigos/sql-nosql-bancos-relacionais-nao-relacionais. Acesso em: 20 abr. 2024.

CARDOSO, C. **Web based x Cliente-Servidor**: Porque isso é importante? | kite mes. Kitemes, 2012. Disponível em: http://www.kitemes.com.br/2012/06/06/web-based-x-cliente-servidor-porque-isso-e-importante-para-minha-empresa/. Acesso em: 17 abr. 2024.

FIGUEIREDO, K. F.; ESCOBAR, D. Gestão de capacidade em serviços. **Relatórios COPPEAD**, Universidade Federal do Rio de Janeiro, 2004. Disponível em: https://www.coppead.ufrj.br/upload/publicacoes/relatorio_coppead_231.pdf. Acesso em: 10 mar. 2023.

FILA de até 1h23 faz Procon-SP autuar 4 bancos. 2014. Disponível em: https://economia.uol.com.br/noticias/redacao/2014/05/26/fila-de-ate-1h23-faz-procon-sp-autuar-4-bancos-multa-pode-ser-de-r-74-mi.htm. Acesso em: 22 jul. 2025.

GIL, A. C. **Métodos e técnicas de pesquisa social**. 6. ed. [*S. l.*]: Editora Atlas SA, 2014. Acesso em: 10 abr. 2024.

GOOGLE. **Flutter Documentation**. 2023. Disponível em: https://docs.flutter.dev/. Acesso em: 24 jul. 2025.

GRINBERG, M. **React Explained**. Pragmatic Bookshelf, 2020. Disponível em: https://pragprog.com/titles/mgrexp/react-explained/. Acesso em: 24 jul. 2025.

GROSS, D. J.; SHORTLE, J. M.; THOMPSON, J. M.; HARRIS, C. M. **Fundamentals of Queueing Theory**. 5. ed. Wiley, 2018. Disponível em: https://www.wiley.com/en-us/Fundamentals+of+Queueing+Theory%2C+5th+Edition-p-9781119456321. Acesso em: 22 jul. 2025.

HIMO, A. J. J.; MEDINA, A. G. M.; SANTOS, J. M.; SERVITO, J. G.; ALEGADO, R. T. Community request queue management system for local government unit of cabanatuan city. **International Journal**, v. 11, n. 6, 2022. Acesso em: 12 abr. 2024.

JúNIOR, R. d. O. L. **Otimização via Internet**. 108 p. Dissertação (Dissertação (Mestrado em Matemática Aplicada)) – Universidade Estadual de Campinas, Campinas, 2010. Disponível em: https://hdl.handle.net/20.500.12733/1613115. Acesso em: 25 jul. 2025.

LEIVA, L. Progressive web apps: Bridging the gap. **Communications of the ACM**, ACM, 2021. Disponível em: https://dl.acm.org/doi/10.1145/3447526. Acesso em: 24 jul. 2025.

LI, J.; ZHANG, H. A generalized queuing model and its solution properties. **Transportation Research Part B: Methodological**, Elsevier, v. 79, p. 78–92, 2015. ISSN 0191-2615. Disponível em: https://www.sciencedirect.com/science/article/pii/S0191261515001034. Acesso em: 12 mar. 2023.

LOSS, E. C. **Gerenciamento de filas para atendimento ao público**: Estudos de caso ebanx e central de estágios da prefeitura municipal de curitiba. 109 p. Dissertação (Dissertação (Mestrado Profissional em Economia)) – Universidade Federal do Paraná, Curitiba, 02 2019. Disponível em: https://acervodigital.ufpr.br/handle/1884/64167. Acesso em: 25 jul. 2025.

MOURA, P. **Uma breve explicação sobre testes automatizados**. Medium, 2019. Disponível em: https://medium.com/@paulociecomp/uma-breve-explica%C3%A7%C3%A3o-sobre-testes-automatizados-5f6060cfbad4. Acesso em: 19 abr. 2024.

NEWMAN, S. **Building Microservices**. 2. ed. O'Reilly, 2021. Disponível em: https://www.oreilly.com/library/view/building-microservices-2nd/9781492034018/. Acesso em: 23 jul. 2025.

NICE, B. **Front-End vs Back-End vs Full Stack Development**. Level Up!, 2018. Disponível em: https://medium.com/level-up-web/front-end-vs-back-end-vs-full-stack-development-78267f545121. Acesso em: 19 abr. 2024.

OLUWATOSIN, H. S. Client-server model. **IOSR Journal of Computer Engineering**, IOSR Journals, v. 16, n. 1, p. 67–71, 2014. Acesso em: 18 abr. 2024.

POHL, K. **Requirements engineering fundamentals**: a study guide for the certified professional for requirements engineering exam-foundation level-ireb compliant. [*S. l.*]: Rocky Nook, Inc., 2016. Acesso em: 16 abr. 2024.

PROCON. **Procon-ES multa Caixa Econômica Federal em mais de R**$110 mil reais por ampla s fila se atendi

. Acesso em: 10 abr. 2024.

PROCON-RJ autua banco Itaú em Copacabana por exceder tempo de espera em fila. 2023. Disponível em: https://diariodorio.com/procon-rj-autua-banco-itau-em-copacabana-por-exceder-tempo-de-espera- Acesso em: 22 jul. 2025.

RAMOS, F. J. C. **O atendimento precário nas filas de bancos no Brasil**: Sedep. SEDEP, 2015. Disponível em: https://www.sedep.com.br/artigos/o-atendimento-precario-nas-filas-de-bancos-no-brasil/. Acesso em: 10 abr. 2024.

REHKOPF, M. **User Stories**: Examples and template. Atlassian. Disponível em: https://www.atlassian.com/agile/project-management/user-stories. Acesso em: 20 abr. 2024.

SILVA, E. L. D.; MENEZES, E. M. **Metodologia da pesquisa e elaboração de dissertação**. 4. ed. Florianópolis: UFSC, 2005. 123 p. Disponível em: https://repositorio.ufsc.br/handle/123456789/103956. Acesso em: 11 abr. 2024.

SOMMERVILLE, I. **Engenharia de Software**: ed. 10. [*S. l.*]: Pearson Universidades, 2019. Acesso em: 16 abr. 2024.

UMUARAMA, P. de. **Procon multa bancos em mais de R**$250 mil por tempo de espera em filas. Umuarama. p
. Acesso em: 11 abr. 2024.

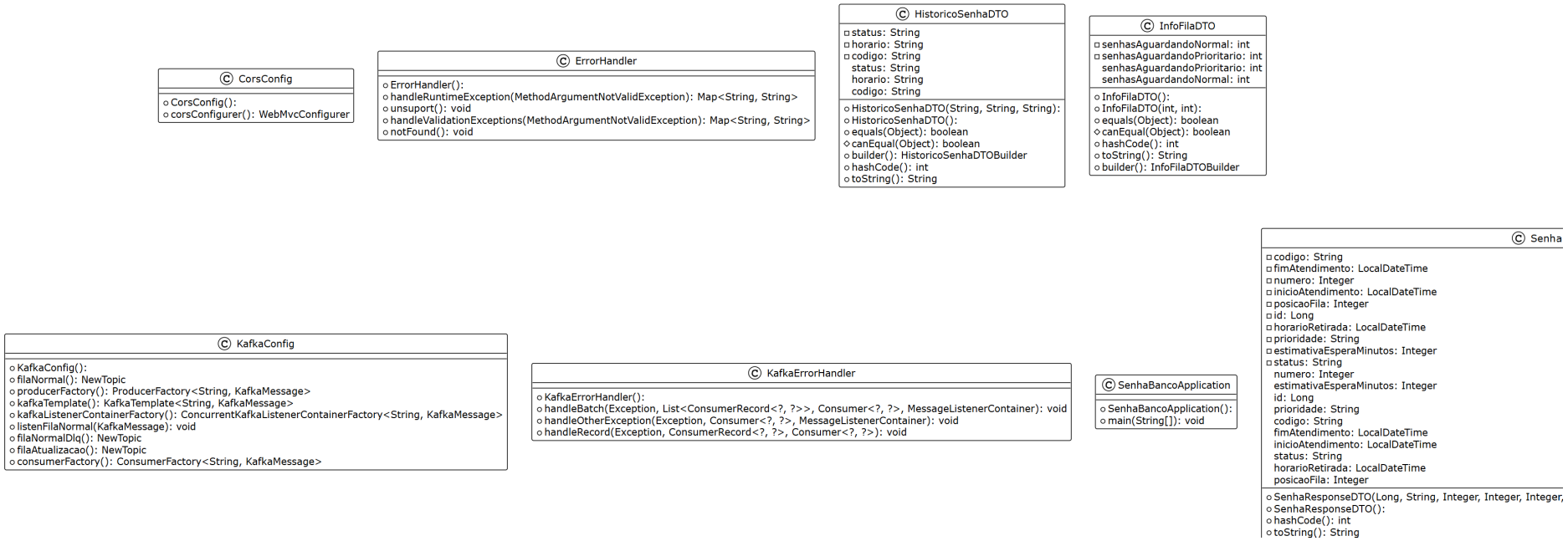
VIJAYAN, J.; RAJU, G.; JOSEPH, M. Collaborative requirements elicitation using elicitation tool for small projects. In: **2016 international conference on signal processing, communication, power and embedded system (scopes)**. [*S. l.*]: IEEE, 2016. p. 340–344. Acesso em: 17 abr. 2024.

WELLS, C. **Spring Boot in Action**. Manning, 2016. Disponível em: https://www.manning.com/books/spring-boot-in-action. Acesso em: 23 jul. 2025.

XU, J.; LIU, D. Queuing models to improve port terminal handling service. **Systems Engineering Procedia**, Elsevier, v. 4, p. 345–351, 2012. ISSN 2211-3819. Disponível em: https://www.sciencedirect.com/science/article/pii/S2211381911002384. Acesso em: 12 mar. 2023.

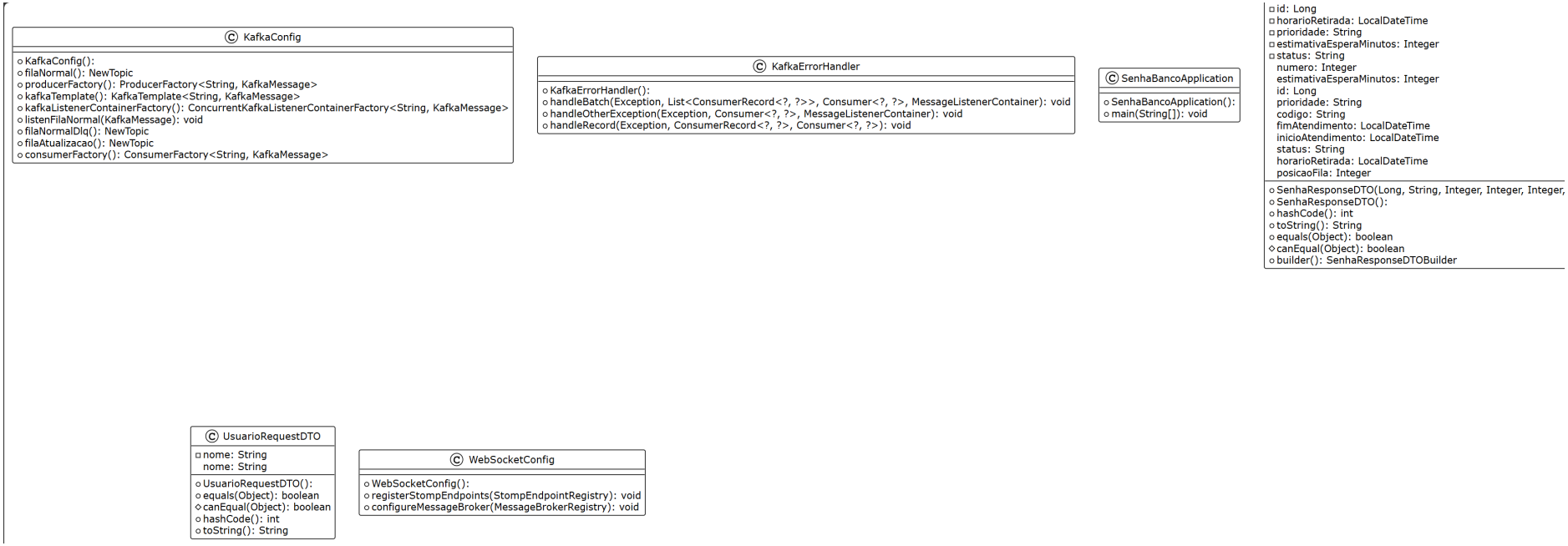
APÊNDICE A – DIAGRAMA DE CLASSES

Figura 13 – Diagrama de Classes – Página 1



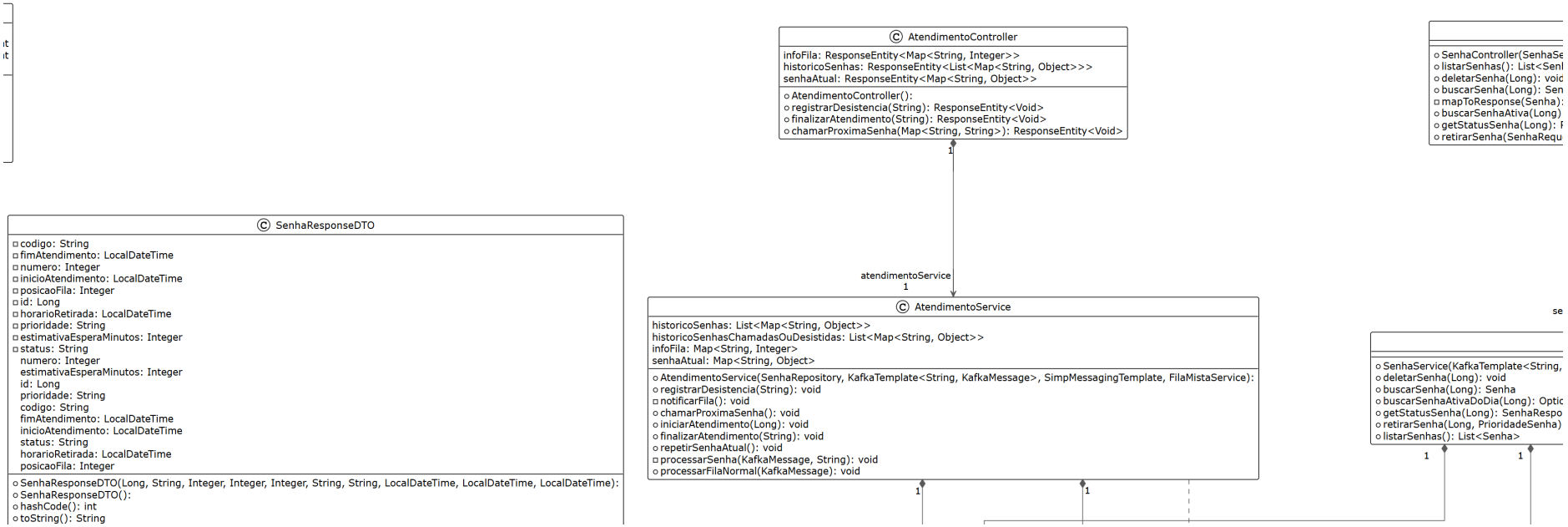
Fonte: elaborado pelo autor (2025).

Figura 14 – Diagrama de Classes – Página 2



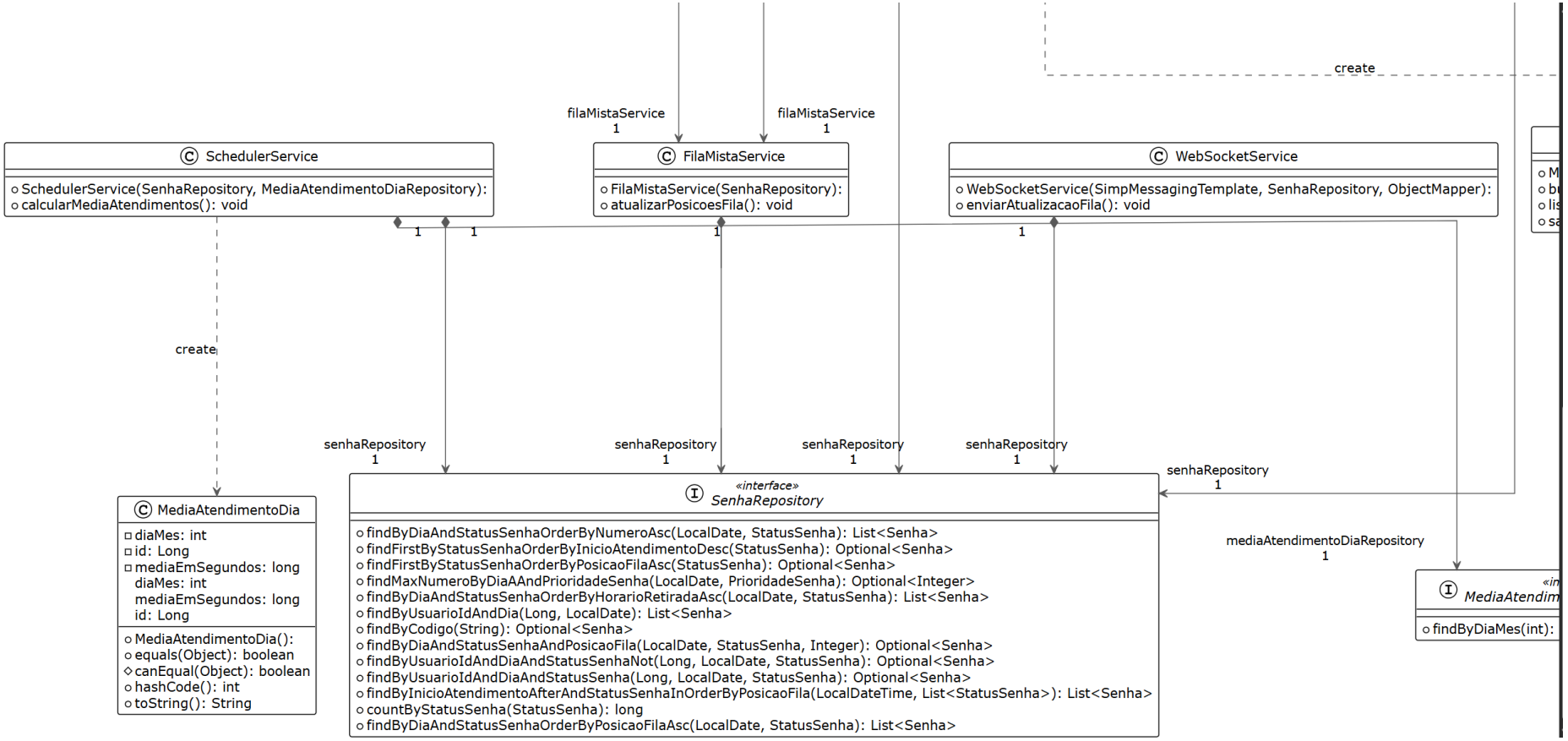
Fonte: elaborado pelo autor (2025).

Figura 15 – Diagrama de Classes – Página 3



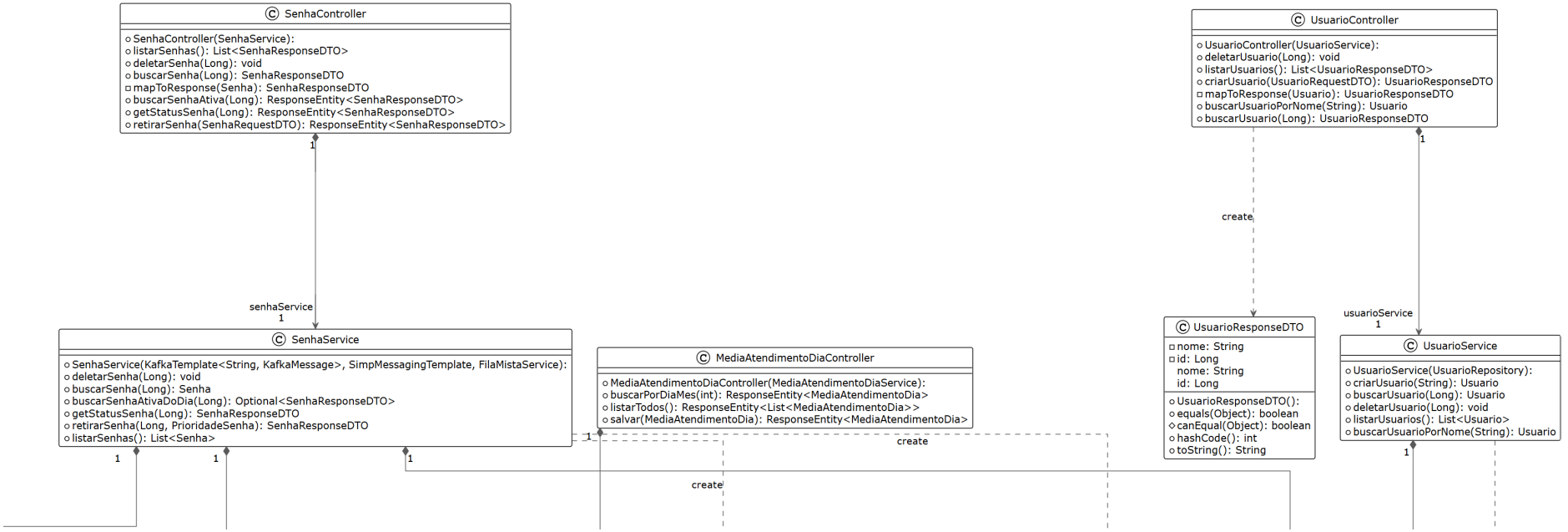
Fonte: elaborado pelo autor (2025).

Figura 16 – Diagrama de Classes – Página 4



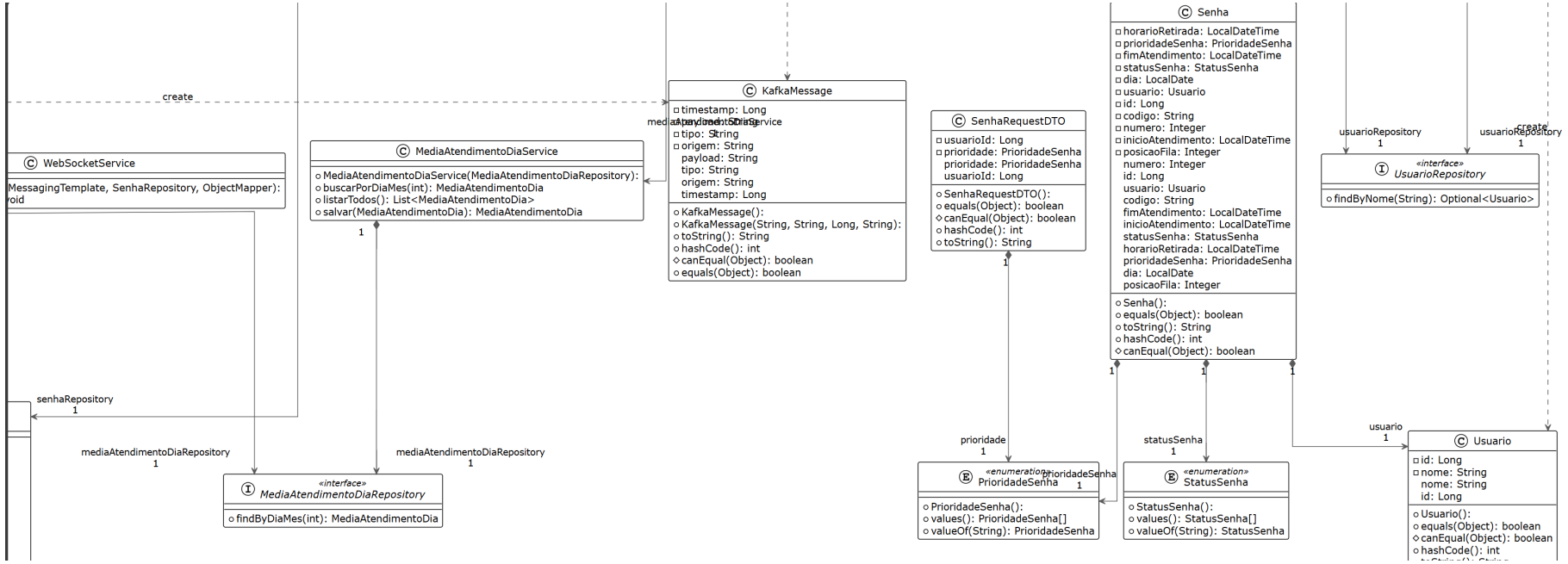
Fonte: elaborado pelo autor (2025).

Figura 17 – Diagrama de Classes – Página 5



Fonte: elaborado pelo autor (2025).

Figura 18 – Diagrama de Classes – Página 6



Fonte: elaborado pelo autor (2025).