INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Área Departamental de Engenharia de Electrónica e Telecomunicações e de Computadores

Sistema de gestão e previsão de filas de espera

Ronilton Carvalho de Almeida Neves

Licenciado em Engenharia Informática e de Computadores

Projeto Final para obtenção do Grau de Mestre
em Engenharia Informática e de Computadores

Orientador : Professor Especialista Paulo Alexandre Leal Barros Pereira, ISEL-ADEETC

Júri:

Presidente: Professor Doutor Nuno Miguel Soares Datia, ISEL-ADEETC

Vogais: Professor Especialista José Luis Falcão Cascalheira, ISEL-ADEETC
Professor Especialista Paulo Alexandre Leal Barros Pereira, ISEL-ADEETC

Setembro, 2021

INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Área Departamental de Engenharia de Electrónica e Telecomunicações e de Computadores

Sistema de gestão e previsão de filas de espera

Ronilton Carvalho de Almeida Neves

Licenciado em Engenharia Informática e de Computadores

Projeto Final para obtenção do Grau de Mestre
em Engenharia Informática e de Computadores

Orientador : Professor Especialista Paulo Alexandre Leal Barros Pereira, ISEL-ADEETC

Júri:

Presidente: Professor Doutor Nuno Miguel Soares Datia, ISEL-ADEETC

Vogais: Professor Especialista José Luis Falcão Cascalheira, ISEL-ADEETC
Professor Especialista Paulo Alexandre Leal Barros Pereira, ISEL-ADEETC

Setembro, 2021

À minha familia!

Agradecimentos

Agradeço ao meu orientador, Professor Paulo Pereira, pela disponibilidade, apoio e orientação que proporcionou.

Ao Instituto Superior de Engenharia de Lisboa (ISEL) e docentes por todos os ensinamentos que contribuíram para a minha formação, e os colegas que me acompanharam nesta caminhada.

A todos os meus colegas de trabalho que deram o apoio e incentivo para concluir o projeto.

Um agradecimento especial aos meus pais e à minha família que estiveram sempre disponíveis, pelo excepcional apoio, incentivo e por acreditar em mim nesta etapa académica.

Por último agradeço os meus amigos e colegas de casa que contribuíram com seu apoio e incentivo.

Resumo

Atualmente em serviços institucionais (Bancos, Serviços de Estrangeiros e Fronteiras, Embaixadas, Segurança Social entre outros), por vezes se verifica uma grande perda de tempo por parte dos clientes devido a longas filas de espera e/ou devido a demora nos atendimentos. Neste projeto, foi desenvolvido um sistema que permite gerir as filas de espera, oferencedo ao utilizador uma previsão de tempo em relação ao tempo de espera.

O sistema, para interação com um indivíduo tem uma componente *mobile* que permite a recolha de senhas, tendo atualizações em tempo real sobre o estado das filas de interesse. Ao retirar uma senha, o objetivo passa por apresentar ao cliente do sistema a informação sobre o tempo de espera previsto. Um mecanismo de notificação é utilizado de forma a alertar o indivíduo quando a vez de ser atendido se aproxima. Desta forma, deixa de ser necessário a deslocação para recolha de senhas, permitindo ao indivíduo realizar outras tarefas fora do local que pretende visitar.

Palavras-chave: Serviços, *mobile*, mecanismo de notificação, senhas, previsão e tempo real.

Abstract

Nowadays, in institutional services (Banks, Portuguese Immigration and Borders Service, Embassies, Social Security and others), there is often a great time loss from customers due to long queues and/or delays when being attended to. With this project, a sistem that allows queue management and waiting time forecast was developed.

The system, to interact with an individual, has a mobile component that allows the taking of the service's tickets, having real time updates about the state of the queues of interest. When taking a ticket, the objective is to show the client of the system the information about the expected waiting time. A notification mechanism is used to notify an individual when his turn to be attended is approaching. With this, there is no longer the need to dislocate to retrieve tickets, allowing an individual to carry out other tasks outside the place he/she wishes to visit.

Keywords: Services, mobile, notification mechanism, tickets, forecast and real-time.

Índice

Lista de Figuras

Lista de Listagens

Lista de Abreviaturas e Siglas

API Application Programming Interface. 39, 41

DAL Data Access Layer. 41

EA Entidade Associação. 49

FCFS First Come First Served. 8, 11, 12

FCM Firebase Cloud Messaging. 24, 40, 41, 42, 43, 47, 48

FIFO First In First Out. 8, 11, 12

JDBC Java Database Connectivity. 41

MVC Model–View–Controller. 41

MVVM Model–View–ViewModel. 43

RGPD Regulamento Geral sobre a Proteção de Dados. 44

1

Introdução

Atualmente em serviços institucionais (Bancos, Serviços de Estrangeiros e Fronteiras, Embaixadas, Segurança Social entre outros), por vezes se verifica uma grande perda de tempo por parte dos clientes devido a longas filas de espera e/ou devido a demora nos atendimentos quando se tem uma marcação.

Ao se deslocar a um serviço institucional, um cliente nunca sabe exatamente quanto tempo é que pode demorar até ser atendido nem se o atendimento será rápido. Alguns serviços apresentam informação sobre o tempo de espera previsto e/ou o tempo médio de atendimento na senha retirada pelo cliente. No entanto, na perspectiva do cliente, essa informação por vezes não é verificada, acabando por não ter o tempo de espera apresentado na senha.

Para tratar assuntos em determinados serviços de forma presencial, atualmente não se consegue evitar possíveis demoras durante os atendimentos e/ou longas filas, visto que grande parte dos serviços não fazem a recolha e tratamento de dados em tempo real e por consequência, não fornecem plataformas que apresentam o estado das filas. Com isso, um cliente pode deslocar-se a um posto de um serviço que esteja lotado, da mesma forma como pode deslocar-se para um posto vazio.

No presente documento está detalhado a identificação do problema relacionado com as filas de espera, assim como as soluções e componentes desenvolvidas para a sua resolução.

1.1 Motivação

Durante anos de observação e vivência de filas longas e/ou demoras nos atendimentos em certos serviços, complementando-se com as pesquisas online, constatou-se que apesar de todos os esforços na tentativa de melhorar as filas e demoras, um cliente por várias vezes se encontra sem uma estimativa de quanto tempo irá precisar para resolver os assuntos de seu interesse e é sempre forçado a realizar uma deslocação.

Com as tecnologias presentes no dia-a-dia de cada indivíduo e com a facilidade de acesso e processamento de dados, o contexto deste projeto permite o desenvolvimento de um sistema com capacidade de gestão e distribuição de senhas, de acordo com as regras definidas pelas entidades de cada serviço, que consegue dar ao cliente informações referentes aos estados dos vários postos dos serviços de interesse, permitindo a este planear e realizar outras atividades sem estar "preso" a um posto a aguardar a vez de ser atendido.

1.2 Objetivos

O principal objetivo deste projeto passa pela idealização, desenho e implementação de um sistema que permite a gestão centralizada de vários serviços de forma a otimizar o tempo gasto em filas de espera e nos atendimentos dos serviços institucionais. Para alcançar o objetivo, o sistema tem que ser capaz de:

- Ter a capacidade de estimar quanto tempo demora até que um cliente seja atendido num determinado serviço;
- Adaptar-se com a entrada de novos dados, apresentando novas estimativas aos clientes;
- Ser flexível e escalável de forma a permitir integração de diferentes tipos de serviços;

1.3 Metodologia

1.3.1 Prototipagem

De forma a permitir a idealização, desenho e implementação do sistema, realizou-se a construção de um protótipo funcional, que permite a um utilizador a visualização de

todos os tipos de senhas disponíveis, recebendo atualização em tempo real do estado das senhas. Para além da visualização o utilizador pode escolher retirar a senha que pretende, continuando a receber atualizações sobre o estado da fila da senha escolhida, bem como uma notificação quando a vez de ser atendido se aproxima.

1.3.2 Investigação

Um dos principais desafios deste projeto é a capacidade de estimar os tempos de espera de diversas filas, o que leva ao estudo dos algoritmos e metodologias existentes no que diz respeito à previsão de intervalos de tempo. Este estudo está mais virado para a área de aprendizagem automática (*time series forecasting [22]*) e de mineração de dados [20]. Com a conclusão da investigação, foi desenvolvida uma componente que permite realizar a previsão de tempo, de acordo com os dados históricos dos tempos decorridos nos atendimentos.

1.4 Estrutura do Documento

O presente documento encontra-se dividido em 5 capítulos:

No capítulo 2, é apresentado o estado da arte de forma a contextualizar as ideias em torno do tema do trabalho.

No capítulo 3, é detalhado como é que o *moving average* pode ser adaptado ao projeto para previsão de tempos de espera.

No capítulo 4, é feita a caracterização do protótipo desenvolvido.

No capítulo 5, é descrito a arquitetura baseada para a realização do protótipo.

No capítulo 6, é apresentado as conclusões obtidas ao longo da realização do projeto e é apresentado o trabalho futuro que pode ser feito de forma a evoluir o sistema.

2

Estado da Arte - Teoria de Filas de Espera

Neste capítulo é apresentado o enquadramento sobre a teoria de filas de espera, de forma a expor todos os conceitos que podem ser utilizados para alcançar o objetivo principal do projeto.

A ocorrência de filas de espera é um fenómeno do quotidiano, seja em lojas, no banco ou qualquer local onde é necessário esperar por um serviço. Atualmente, estamos habituados a um tempo de espera razoável nas filas. No entanto, sempre que este é maior que o normal, existem consequências que podem afetar por exemplo a economia de uma país visto que quanto maior tempo de espera, menor fornecimento de serviço existe. A fila de espera é composta por duas componentes importantes, cliente e serviço.

Agner Krarkup Erlang (Lonborg, 1 de Janeiro de 1878 - Lonborg, 3 de Fevereiro de 1929) foi o matemático, estatístico e engenheiro dinamarquês que inventou os campos de Engenharia de Tráfego e da Teoria das Filas. A teoria das filas [2] [13], é o estudo todas as diferentes filas de espera, utilizando modelos matemáticos para demonstrar o seu comportamento conforma a procura aleatória de clientes. Aplicando métricas aos modelos, tenta-se fornecer um atendimento adequado às necessidades dos clientes, indicando como é que a fila deve funcionar.

As filas de espera são uma necessidade, visto que por norma, o número de clientes é superior ao número de servidores. O objetivo da teoria das filas de espera passa por

otimizar o desempenho de um sistema, de modo a reduzir os seus custos operacionais e a aumentar a satisfação dos clientes.

A existência de longas filas de espera em diversos tipos de serviço, constitui um problema que deriva do estabelecimento de *trade-offs* entre o custo do serviço e o custo do tempo perdido pelos clientes na fila de espera.

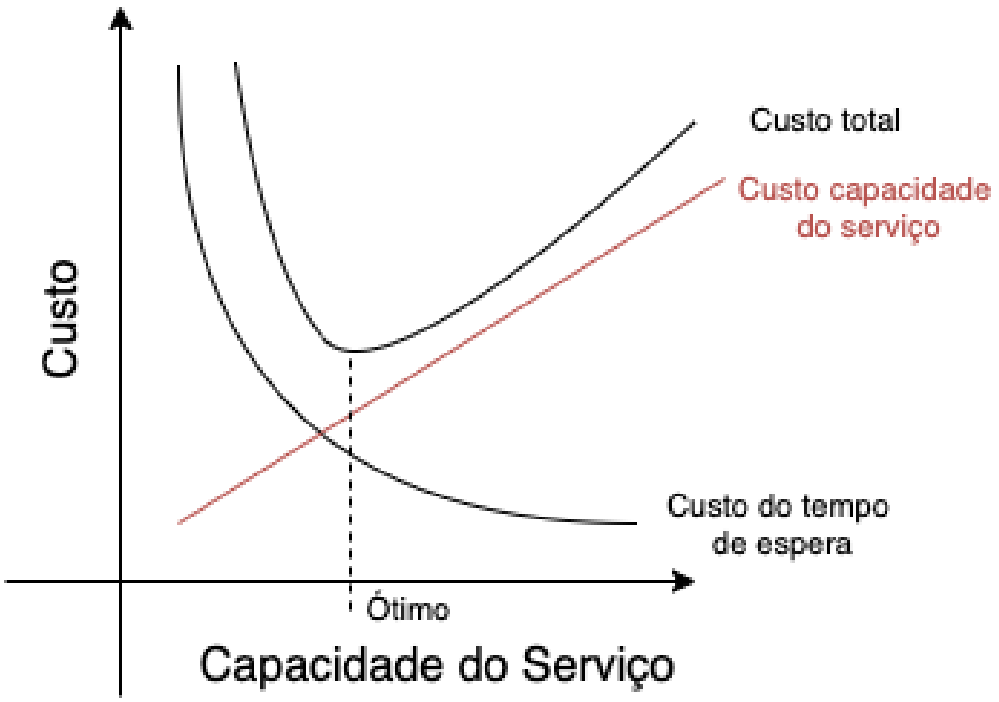

Figura 2.1: Trade-off entre o custo da capacidade do serviço e o custo do tempo de espera (retirada de Waiting Line Management: Costs Function and Service Level [23])

As filas de espera existem muito por causa da insuficiente capacidade dos serviços, que por sua vez, existe devido ao custo elevado a considerar pelo aumento da mesma. Por outro lado, o *trade-off* representado na Figura 2.1, nem sempre é considerado, visto que só é avaliado o custo do serviço, o que diminui a capacidade de serviço, criando filas de espera.

2.1 Sistema de fila de espera

Um sistema de filas, representado na Figura 2.2, é estruturada pelos clientes (população ou fonte), filas e pelo serviço de atendimento. O processo do sistema, dá-se início com os clientes a entrarem no sistema, juntando-se à fila pretendida (caso não esteja vazio), aguardando ser chamado pelo servidor, de acordo com uma regra, denominada

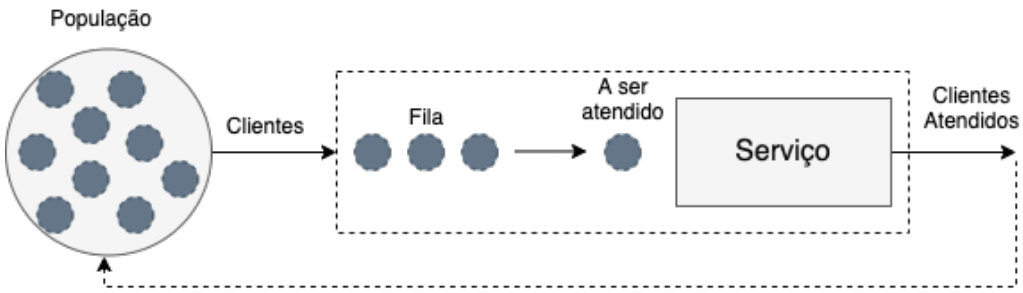

Figura 2.2: Sistema de filas de espera (retirada de Filas de espera [2])

por disciplina da fila. O cliente ao ser atendido, sai do sistema, podendo regressar posteriormente. Grande parte dos modelos de filas de espera utilizam este processo.

Fila + Serviço = Sistema

Número de clientes no sistema (em cada instante) = **estado do sistema**

2.1.1 População ou Fonte

A população ou fonte é responsável por gerar os clientes que vão chegar ao sistema. Uma das características da fonte, é a dimensão, sendo o total de clientes que podem querer o serviço. Esta dimensão pode-se assumir como infinita/ilimitada ou finita/limitada. De forma a realizar os cálculos da previsão de tempo para as filas de espera, a dimensão da fila, não tem grande influência, visto que é necessário os tempos dos clientes já atendidos. A dimensão só irá servir para indicar a quantos clientes é que deve ser dado uma previsão.

A chegada de clientes no sistema, é um aspeto importante na teoria de filas de espera, quando é pretendido lidar com análises das filas nos serviços. Essa chegada é indeterminada, podendo ter vários padrões distintos. Dessa forma é comum assumir que a chegada de clientes, tem uma distribuição de Poisson [14]. Neste projeto não existe a necessidade de análise das chegadas, visto que o maior foco é os clientes já atendidos.

2.1.2 Fila de espera

A fila de espera é onde os clientes ficam à espera de serem atendidos. Ela é caracterizada pelo número máximo de clientes que pode ter e disciplina da fila.

Relativamente ao número de filas, representado na Figura 2.3, pode-se considerar uma fila simples (fila única), onde pode haver um ou mais postos de atendimento (múltiplos servidores), ou múltiplas filas, com uma fila por posto de atendimento, onde cada par fila/posto de atendimento constitui um sistema separado de fila de espera.

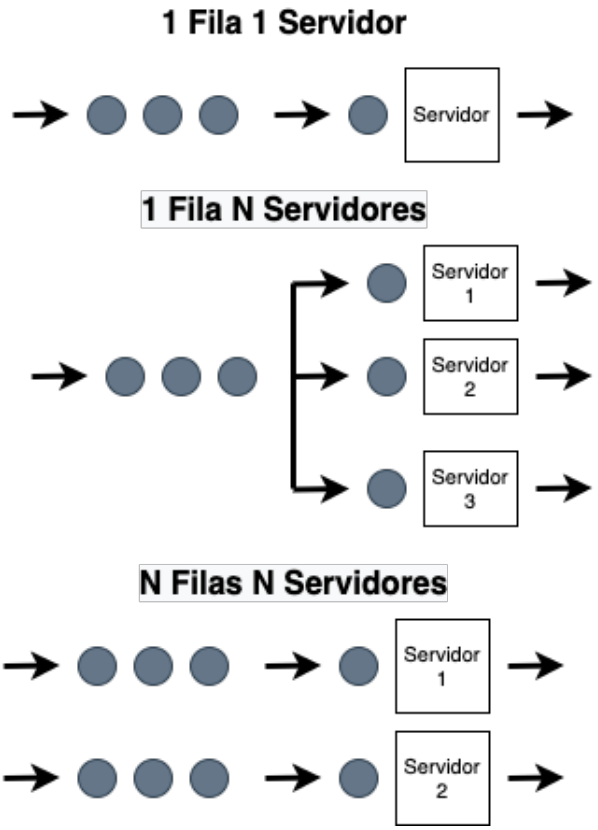

Figura 2.3: Filas Únicas e Múltiplas (adaptado de Teoria das Filas [21])

O comprimento da fila está relacionado com a capacidade do sistema, podendo ser finita de acordo com a limitação do espaço, ou pode infinita, quando não existe uma limitação ao tamanho da fila de espera.

Na Figura 2.3, é possível verificar os múltiplos tipos de servidores.

2.1.3 Disciplina da fila de espera

A disciplina da fila de espera, se refere à ordem que os membros de uma fila são escolhidos para serem atendidos. Existem várias possibilidades, onde pode haver um atendimento por ordem de chegada ou por prioridade. A disciplina foco deste projeto é o atendimento por ordem de chegada, sendo uma disciplina *First In First Out (FIFO)*, conhecido também por *First Come First Served (FCFS)*. Esta disciplina, sendo a

mais comum é a única considerada, visto que com o atendimento aleatória não é possível fornecer uma previsão sobre o tempo de espera aos clientes. Atendimentos por prioridade, pode ser inserido na disciplina FIFO, visto que geralmente existe uma fila própria que utiliza as mesmas regras.

2.1.4 Serviço

O serviço corresponde aos postos onde os clientes podem obter/requerer serviços, em que cada contém canais paralelos de atendimentos, chamados servidores. A capacidade de cada posto, é determinada pelo número de canais paralelos disponíveis e pela capacidade de cada canal em servir os clientes. Esta componente do sistema, tem grande importância, tendo em conta a tamanha influência que pode ter no tempo despendido pelos clientes nos postos de serviço e na economia do serviço.

2.1.5 Medidas de desempenho

De forma a gerir um sistema de filas de espera algumas medidas têm que ser levadas em conta:

- Comprimento médio da fila
- Número médio de clientes no sistema
- Tempo médio de espera na fila
- Tempo médio de espera no sistema
- Taxa média de ocupação do serviço

Com essas medidas é possível encontrar soluções equilibradas que permitem estabelecer *trade-offs* entre custo e qualidade de serviço.

Na Figura 2.4, é possível observar que quanto maior o nível do serviço, menor o tempo de espera, tendo no entanto um custo bastante acrescido à medida que esse nível aumenta.

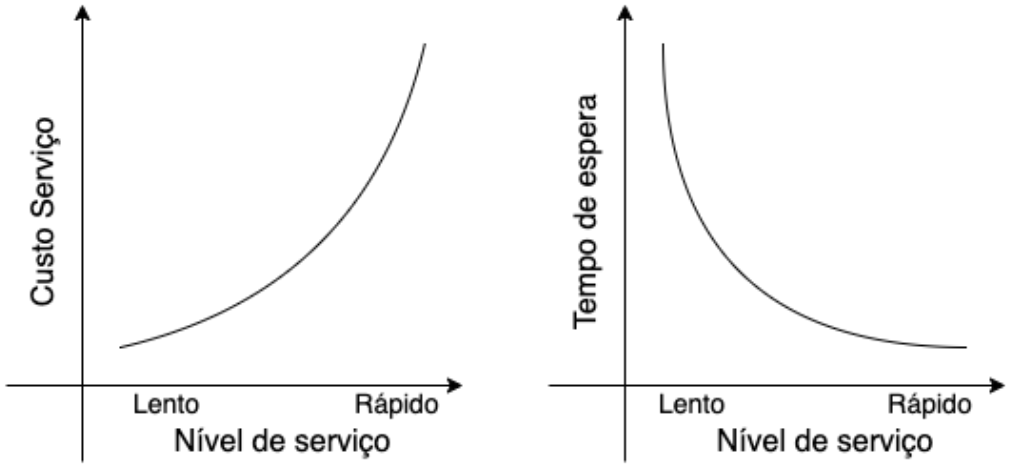

Figura 2.4: Funções de custo de serviço e tempo de espera em relação ao nível do serviço (adaptada de Waiting Line Management: Costs Function and Service Level [23])

2.2 Processo de vida e morte

No contexto do sistema de filas de espera, em grande parte do modelos de filas elementares, assume-se que a chegada de clientes à fila e a consequente saída, ocorre de acordo com o processo de vida e morte. A este processo, vida (nascimento) se refere à entrada enquanto que morte é a saída de um cliente já atendido (Figura 2.5). A modelização deste processo, supõe estados de equilíbrio, excluindo por exemplo a variação da taxa de chegada, ou a possibilidade de ter números anormais de clientes no início do funcionamento. Neste processo, assume-se as seguintes hipóteses:

- Dado o instante t, com o sistema no estado n:
  - A probabilidade do próximo nascimento (chegada) segue uma distribuição exponencial negativa com parâmetro λn;
  - A probabilidade próxima morte (serviço terminado) segue uma distribuição exponencial negativa com parâmetro μn.
- Nunca podem ocorrer simultaneamente mais do que um nascimento ou uma morte.

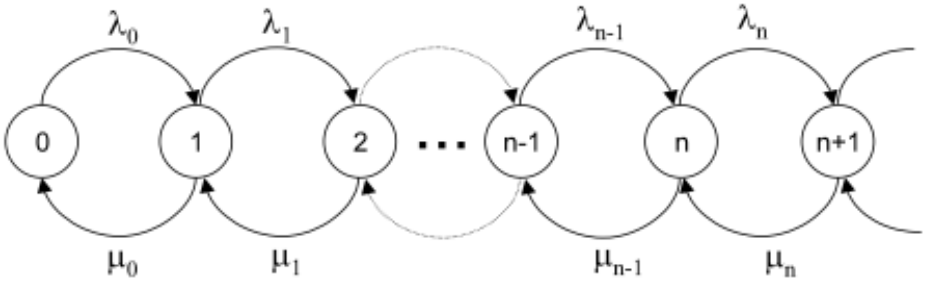

Figura 2.5: Diagrama de transição correspondente ao processo de vida e morte (retirada de Filas de espera [2])

2.3 Modelos das filas de espera baseados nos processos de vida e morte

A maioria dos modelos analíticos de filas de espera, têm como base o processo de vida e morte, onde as entradas têm uma distribuição de Poisson [14] e tempos de atendimento com distribuição exponencial negativa. No entanto, cada modelo se difere nas suas assumpções sobre como é que a chegada e a saída afetam o estado do sistema.

2.3.1 Classificação da filas de espera - X/Y/Z/W

X, Y — Distribuições do intervalo de tempo entre chegadas e do tempo de serviço, respetivamente:

- M — distribuição exponencial negativa
- G — distribuição não especificada
- D - chegadas ou atendimentos determinísticos

Z — número de servidores em paralelo.

W — outras características do sistema, tais como comprimento da fila ilimitado ou população finita:

- em branco ou ∞ — modelo-base sem qualquer restrição adicional
- K — comprimento da fila limitado, não podendo o número de elementos no sistema exceder K
- N - população finita

2.3.2 Modelo M/M/1

Neste modelo, existe um único posto de atendimento (1) e assume-se que a distribuição do intervalo de tempo entre chegadas (M) e do tempo de serviço (M) é exponencial. Assume-se que o sistema tem capacidade ilimitada, e população, utilizando uma disciplina FIFO/FCFS no atendimento.

2.3.3 Modelo M/M/S

Neste modelo, existem múltiplos servidores (S), que fornecem serviço independentemente uns dos outros. Assume-se que a distribuição do intervalo de tempo entre chegadas (M) e do tempo de serviço (M) é exponencial. Assume-se que o sistema tem capacidade ilimitada, e população, utilizando uma disciplina FIFO/FCFS no atendimento.

2.3.4 Modelo M/M/1/K

Este modelo difere de M/M/1 visto que tem a capacidade do sistema limitada até K clientes ao mesmo tempo no local de prestação de serviço. Quando o sistema tiver K clientes, será recusado novas chegadas de cliente. A disciplina utilizada é a FIFO.

2.3.5 Modelo M/M/S/K

Neste modelo, o sistema tem capacidade finita, com s servidores, com tempos de atendimento exponenciais independentes e identicamente distribuídos (não dependem do estado do sistema). A capacidade do sistema deve ser, no mínimo, tão grande quanto o número de servidores, $K \geqslant s$. Utiliza a disciplina FIFO.

2.3.6 Modelo M/M/∞

Neste modelo, o serviço é ilimitado, existindo um número infinito de servidores disponíveis, não existindo uma fila de espera. A distribuição do intervalo de tempo entre chegadas e tempo de serviço é exponencial. A capacidade do sistema e da população é infinita. A disciplina seguida é a FCFS.

2.3.7 Outros modelos

Para a análise de filas de espera, existem vários tipos de modelos, no entanto neste trabalho só é tido em consideração os modelos com distribuição exponencial.

2.4 Redes de filas de espera

Em muitos casos reais, um cliente pode passar por uma sucessão de filas, provocando dessa forma um sistema com múltiplas fases, ficando perante uma rede de filas de espera. Um exemplo seria as urgências hospitalares, onde é necessário passar por uma fase de triagem até à consulta e a possíveis exames.

2.5 Considerações finais sobre filas de espera

Em qualquer posto de serviço existente, vai sempre existir filas de espera para os clientes, em vários modelos diferentes. A criação de um sistema de gestão de senhas e de previsão de tempos de espera irá ter impacto na forma como os clientes são atendidos, permitindo a estes uma melhor gestão do tempo para outras tarefas enquanto aguardam a vez de ser atendido. Os dados recolhidos podem ajudar a otimizar a forma como cada serviço trabalha no dia-a-dia.

3

Moving Average

Neste capítulo, é detalhado como é que o *moving average* pode ser adaptado ao projeto para prever tempos de espera.

3.1 Time Series Forecasting

Time Series Forecasting [5] é uma área importante da aprendizagem automática, que permite prever padrões futuros, de acordo com modelos baseados em dados observados de forma sequencial durante certo período de tempo. *Forecasting* é uma tarefa estatística comum, encontrada em negócios, que fornece guias para tomadas de decisões em planeamentos sobre produção, transporte e recursos humanos.

3.1.1 Métodos de Forecasting

Existem dois métodos de previsão de padrões, que dependem dos dados disponíveis:

- Qualitativo - Utilizado quando não existem dados ou quando os dados são pouco relevantes para a previsão.
- Quantitativo - Utilizado quando existem dados do passado disponíveis, podendo assumir que os padrões passados vão continuar no futuro.

No âmbito deste projeto o **método quantitativo** será utilizado para prever tempos de espera em filas.

3.2 Moving Average

Dados *time series* podem exibir vários padrões pelo que devem ser decompostos por forma que diversas componentes permitam a obter as previsões desejadas. Estas componentes são, tendências (aumento ou redução de longo prazo associados aos dados), ciclos (dados com comportamentos aleatórios) e estação (dados influenciados por fatores externos que ocorrem sempre no mesmo período). Na decomposição de um *time series*, geralmente é feito a combinação entre tendência e ciclos, resultando na componente *trend-cycle*.

Para este projeto, considera-se mais importante a componente *trend-cycle*, visto que nas filas de espera podem ocorrer aumentos e reduções aleatórias nos tempos de atendimento ao cliente. No entanto dados passados da estação podem ser utilizados para uma previsão inicial.

Moving average [8] é um método clássico de decomposição de *time series*, utilizado para filtrar todo o ruído presente em dados aleatórios (do passado), de forma a identificar tendências (valor médio). Existem assim três tipos:

- *SMA (Simple Moving Average)* - média aritmética dos dados recolhidos, através de cálculos simples.
- *WMA (Weighted Moving Average)* - dados mais recentes/relevantes têm maior peso no cálculo do padrão, onde a soma do peso deve ser igual a 1 (100%).
- *EMA (Exponential Moving Average)* - dados mais recentes também têm maior peso no cálculo do padrão, onde o peso é exponencial, podendo alterar consoante a entradas dos dados.

EMA consegue responder mais rapidamente a mudanças nos dados, sendo considerado o melhor a identificar tendências em dados, visto que utiliza pesos exponenciais. Com isso, este tipo é escolhido para ser integrado na previsão de tempo de espera. Abaixo é possível observar a fórmula *EMA*, onde A_t é o valor no período de tempo t e *alpha* (α) o peso a ser aplicado a cada dado.

$$EMA_{t+1} = \alpha A_t + (1 - \alpha) EMA_t$$

$$0 \leqslant \alpha \leqslant 1$$

3.3 Moving Average aplicado ao projeto

Moving average pode ser utilizado para oferecer ao utilizador previsão de tempos de espera em diferentes tipos de serviços. Essa previsão é calculada tendo em conta a análise de dados passados e dados do dia corrente. A melhor forma de aplicar o *moving average*, consiste em aplicar regras e restrições às métricas extraídas, de forma a permitir oferecer aos clientes os tempos de espera mais prováveis de acontecer.

3.3.1 Métricas para cálculo da previsão

Nesta secção é descrita as métricas utilizadas para aplicar o *moving average* sobre uma fila de espera, com as regras e restrições (permitem detetar irregularidades) que permitem uma previsão mais próxima da realidade.

3.3.1.1 Dados do passado

Para dados passados, podem ser considerados os dados de cada dia de semana por estações do ano, em que para analisar, por exemplo, o comportamento da fila de uma segunda feira, teria em conta a análise de todas as segundas feiras passadas, inseridas na estação. Desta forma é possível identificar possíveis irregularidades numa fila de espera, como por exemplo o atendimento rápido ou demorado que sai do padrão normal de atendimentos. No que diz respeito às irregularidades, deve ser possível realizar uma rápida atualização de dados e continuar a oferecer uma previsão correta sem ser afetada pela irregularidade. Para períodos de atendimentos mais regulares, não haverá a necessidade de se adaptar e processar irregularidades.

3.3.1.2 Períodos sem atendimentos

A análise a ser aplicada para previsão de tempo, ignora os "buracos" entre atendimentos, ou seja, se não houver clientes no serviço esse tempo não será considerado no cálculo das previsões, utilizando assim só os tempos de atendimentos pelos servidores.

3.3.1.3 Indivíduos na fila

Para aplicar o *moving average* e consequentes análises nas filas de espera, não é necessário ter o conhecimento do número de pessoas presentes na fila, conseguindo fornecer aos utilizadores as previsões através de cálculos simples, multiplicando a previsão (média) pelo número de clientes na fila.

3.3.1.4 Tipos de fila e Servidores disponíveis

Tendo em conta as filas únicas e múltiplas, descritas no **estado da arte**, para cálculo das previsões nas filas únicas (uma fila e um servidor) e filas múltiplas (uma fila por posto de atendimento), não existe a necessidade de considerar estes tipos de fila visto que, o tempo a ser calculado vai sempre depender do servidor, sem ter a dependência e/ou influência no atendimento das outras filas, obtendo assim previsões diferentes por cada fila. No entanto, existe a necessidade de considerar filas únicas que têm N servidores, visto que o cálculo das previsões dependem dos atendimentos de cada servidor. Neste caso, é necessário ter o conhecimento de quantos servidores é que estão ativos, o que irá provocar intervalos de tempos de atendimento, tendo em conta que cada servidor pode ter a sua velocidade de trabalho. O número de servidores terá influência na previsão apresentada ao cliente. Outro aspeto a ter em conta neste caso é a verificação do servidor que iniciou o primeiro atendimento, por forma a aplicar o tempo já decorrido no atendimento à parte mais baixa do intervalo de previsão. No entanto pode haver um atendimento mais demorado por parte do primeiro servidor (considerado uma irregularidade), o que deve provocar a adaptação ou alteração do sistema, onde o próximo servidor com menos tempo de atendimento, passa a ser considerado para a parte mais baixa do intervalo.

3.3.1.5 Atrasos

Atrasos nos atendimentos, são mais complicados de gerir pelo sistema, visto que não terá o conhecimento do tempo extra que o atendimento pode ter. Neste caso existem duas opções que podem ser tomadas. Primeiro, com a aplicação a dar um tempo por defeito que pode ser renovado quando este expirar. Segundo, realização de análise aos dados passados de forma a ter um valor possível de atraso. No entanto, com a falta de dados passados para o cálculo, a combinação das duas opções pode ser utilizada.

3.3.1.6 Faltas

Faltas não esperadas são consideradas irregularidades pelo que deve ser aplicado um intervalo de validação (ou janela de tolerância), de forma a identificar que houve uma falta (não acidental), permitindo ao sistema informar os clientes da fila. Com um sistema de troca de senhas, será possível evitar possíveis faltas que podem ocorrer por consequência da primeira.

3.3.2 Métricas com possíveis problemas

Nesta secção é descrita as métricas que podem causar uma má previsão do tempo de espera, onde a existência de dados aleatórios com maior desfasamento é uma possibilidade, se forem utilizados da pior forma possível.

3.3.2.1 Tolerância

A tolerância de senhas e utilização da mesma num serviço é algo que, de acordo com as regras existentes pode levar a perda de informação e/ou deteção de irregularidades no sistema. Em serviços que permitem o atendimento de clientes cuja senha tenha passado a vez ([evento de tolerância] dentro do intervalo de tolerância), é necessário ter o conhecimento se essa informação é passada ou não para o sistema. Com a passagem dessa informação e do evento de tolerância, o processo de previsão irá funcionar normalmente, incluído os dados da tolerância. No caso de filas que têm regras definidas para a previsão, caso não haja a informação de tolerância, o sistema pode detetar a existência de irregularidades, visto que o atendimento de um indivíduo depois da sua vez e do anterior e/ou seguinte pode ser considerado como um atendimento.

3.3.2.2 Múltiplas filas - Um servidor

Alguns serviços fornecem a possibilidade de um servidor atender várias filas diferentes, tendo dessa forma duas possibilidades para extração da métrica a ser utilizada na previsão. A primeira, é o atendimento sempre dentro de um determinado padrão, onde assume-se a existência de indivíduos nas filas a serem atendidos pelo servidor. Com o padrão, o processamento da previsão pode ser feito como se fosse uma fila e um servidor, sem a necessidade de ter a informação de múltiplas filas. No entanto existe sempre a possibilidade de uma das filas ficar vazia, pelo que o sistema deve ser capaz de se adaptar ao novo padrão, sendo necessário a informação das filas presentes. A segunda possibilidade, é o atendimento das várias filas de forma aleatória o que provoca más previsões, por não ter uma base/padrão que permite o cálculo.

3.3.2.3 Filas de propósito geral

Filas para servidores que tratam de vários tipos de assunto, não podem existir regras e restrições para o cálculo de previsão, não existindo dessa forma irregularidades. Sendo assim, será necessário o cálculo de intervalos de previsões maiores, de forma a tentar incluir todas as possibilidades de tempo de espera reais.

3.3.2.4 Dia seguinte (sem retirar senha)

Existem serviços que permitem o atendimento de clientes no dia seguinte sem a necessidade de obtenção de uma senha. Dependendo do objetivo do cliente e servidor, em filas baseadas em regras, pode haver a criação de irregularidades ou influência no tempo de espera a ser apresentado aos restantes clientes da fila.

3.3.3 Algoritmo de previsão exponencial

Na listagem 3.1, é apresentado o algorimo *moving average* exponencial, utilizado como base para o calculo da previsão. A partir deste algoritmo é possível aplicar as métricas mencionadas acima de forma a obter os melhores resuldados possíveis.

```
1  package isel.meic.tfm.fei.forecast
2
3  import java.math.BigDecimal
4
5  class ExponentialForecast {
6
7      companion object {
8          private const val NUMBER_OF_DIGITS_AFTER_DECIMAL_POINT: Int =
       2
9
10         fun singleExponentialForecast(data: List<Double>, alpha:
       Double, numForecasts: Int): List<Double> {
11             val forecast = ArrayList<Double>()
12             forecast.add(round(data[0]))
13             var i = 1
14             while (i < data.size) {
15                 forecast.add(round(alpha * data[i - 1] + (1 - alpha)
       * forecast[i - 1]))
16                 i++
17             }
18             var j = 0
19             while (j < numForecasts) {
20                 forecast.add(round(alpha * data[data.size - 1] + (1 -
       alpha) * forecast[i - 1]))
21                 j++
22                 i++
```

```
23                }
24                return forecast
25            }
26
27            private fun round(value: Double): Double {
28                var bigDecimal = BigDecimal(value)
29                bigDecimal = bigDecimal.setScale(NUMBER_OF_DIGITS_AFTER_
    DECIMAL_POINT, BigDecimal.ROUND_HALF_UP)
30                return bigDecimal.toDouble()
31            }
32        }
33    }
```

Listagem 3.1: Exponential Forecast

4

Caracterização do Protótipo

Neste capítulo, é detalhada a abordagem utilizada para a realização do projeto.

4.1 Protótipo

O projeto foi desenvolvido seguindo uma abordagem de prototipagem evolutiva, ilustrada na Figura 4.1. Com a implementação do protótipo, foi possível reduzir as incertezas em torno do projeto.

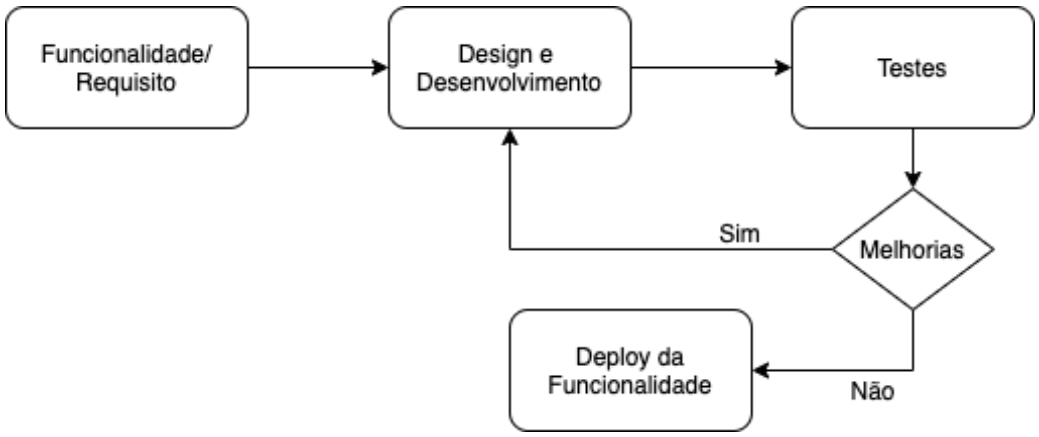

Figura 4.1: Protótipo evolutivo

Funcionalidade/Requisito, é realizado o levantamento das necessidades da funcionalidade a ser implementada e se é viável para o sistema. Após o levantamento, é

realizado o **Design e Desenvolvimento** da funcionalidade, de ponta a ponta, sendo realizados **Testes/Análises** para detetar possíveis problemas e se de facto é o ideal para o objetivo do trabalho. Com a conclusão dos testes, o processo de design e desenvolvimento pode ser repetido/revisto de forma a melhorar a funcionalidade. Caso não seja necessário melhorias, é realizado o **Deploy da funcionalidade**, dando assim como concluída.

4.2 Tecnologias e frameworks utilizadas

Antes do início da implementação do projeto, foi importante a escolha das tecnologias e *frameworks* corretas de forma a fornecer aos utilizadores do sistema as funcionalidades desejadas, garantindo a interoperabilidade entre todas as componentes do sistema.

As tecnologias utilizadas foram escolhidas maioritariamente pelo facto de já ter conhecimento e ter trabalhado com elas, tanto a nível profissional como académico, o que permitiu focar os esforços nos aspetos/requisitos do projeto.

Spring Boot [18] (utilizando a linguagem *kotlin*) foi a *framework* escolhida para criar a componente servidor do sistema, pela facilidade que oferece na configuração *Spring*, permitindo o rápido desenvolvimento de aplicações.

Android (utilizando a linguagem *kotlin*), foi escolhido para fornecer ao utilizador a aplicação cliente do sistema, por ser a tecnologia mais utilizada em dispositivos móveis e a que tenho maiores valências por ser a área de maior foco a nível profissional.

Flutter Web [4] com a linguagem *dart*, é a opção para o simulador de serviço, que tem como objetivo simular aquilo que acontece em postos de serviços, em relação ao avanço das filas, permitindo também a configuração (criação, atualização e eliminação) dos serviços, dos postos e das filas. Por ter já experiência em desenvolvimento de aplicação móveis a nível profissional utilizando o *flutter*, o simulador surgiu como oportunidade para desenvolver as minhas capacidades mais direcionadas para a Web, levando à escolha da *framework*.

Firebase Cloud Messaging (FCM) [3] é a solução escolhida para realizar a entrega de mensagem/notificações aos utilizadores por parte do servidor o que permite realizar a atualização em tempo do estado das filas de espera. Escolhida pelo facto de ser uma solução confiável e sem custos, sendo a única dependência externa do sistema.

Heroku [6] foi a plataforma em nuvem escolhida para realizar o *deploy* do servidor e do simulador de serviço, visto ser uma plataforma que suporta várias linguagens, permitindo a hospedagem desde protótipos (sem custo) a aplicações empresariais.

Node.JS [11] é utilizado simplesmente para a criação de um servidor que permite exibir/expor o Javascript/HTML/CSS estático gerado através do *flutter web*. Foi desenvolvido esta solução pelo facto do Heroku não ter ainda a capacidade de construir aplicações em *flutter web*, nem de exibir ficheiros estáticos visto que o objetivo da plataforma é mais focada em servir requisitos a nível do servidor.

4.3 Requisitos

Para a realização deste projeto, foi feito o levantamento dos requisitos funcionais e não funcionais gerais do sistema, que foram incrementando à medida que o protótipo foi avançando, identificando em primeiro lugar os atores do sistema. A especificação dos requisitos está estruturada sob a forma de modelo de casos de utilização.

4.3.1 Ambiente/Contexto de utilização

O contexto de utilização, presente na Figura 4.2 é a representação geral do sistema, onde os atores dão inicio a operações/ações (casos de utilização), obtendo e visualizando resultados devolvidos pelo sistema.

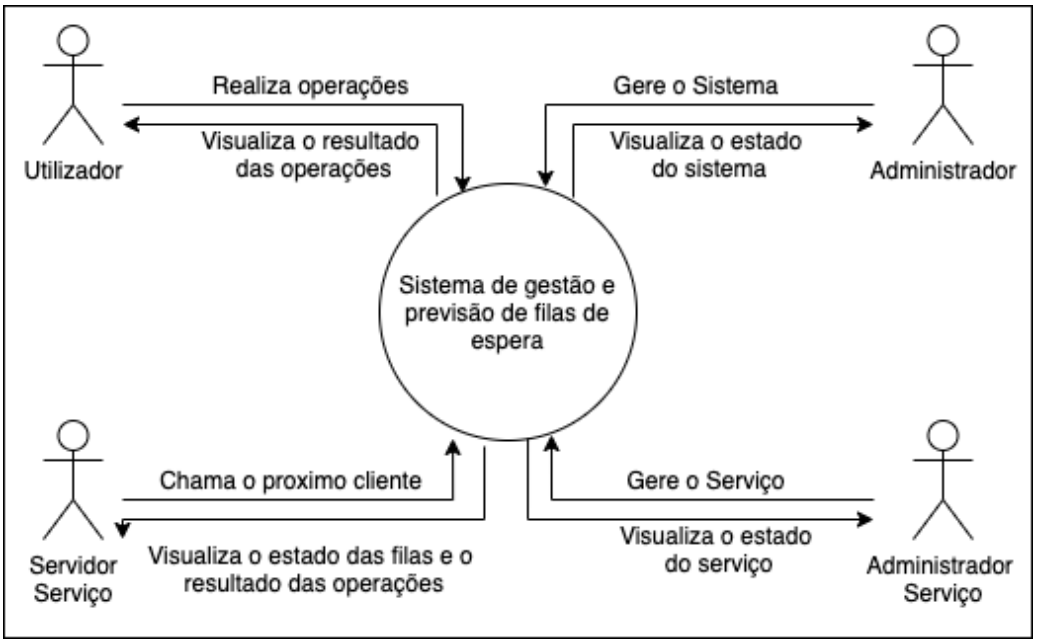

Figura 4.2: Contexto de Utilização

Conceitos apresentados na Figura 4.2:

- Atores/Entidades (utilizador, administrador, servidor de serviço e administrador de serviço) - entidade externa que interage com o sistema de forma a realizar trabalho, mas sobre a qual não tem controlo, estando fora da influência do sistema.
- Sistema de gestão e previsão - Realiza um conjunto de operações de acordo com as ações dos atores.
- Fluxos/Interação - Comunicação entre os atores e o sistema.

4.3.2 Diagramas de Casos de Utilização

Os diagramas de caso de utilização especificam os comportamentos resultantes da interação entre atores e sistema para atingir um objetivo. Para cada tipo de ator identificado, nomeadamente servidor de serviço (Figura 4.3), utilizador (Figura 4.4), administrador (Figura 4.5) e administrador de serviço (Figura 4.6), é apresentado o respetivo diagrama.

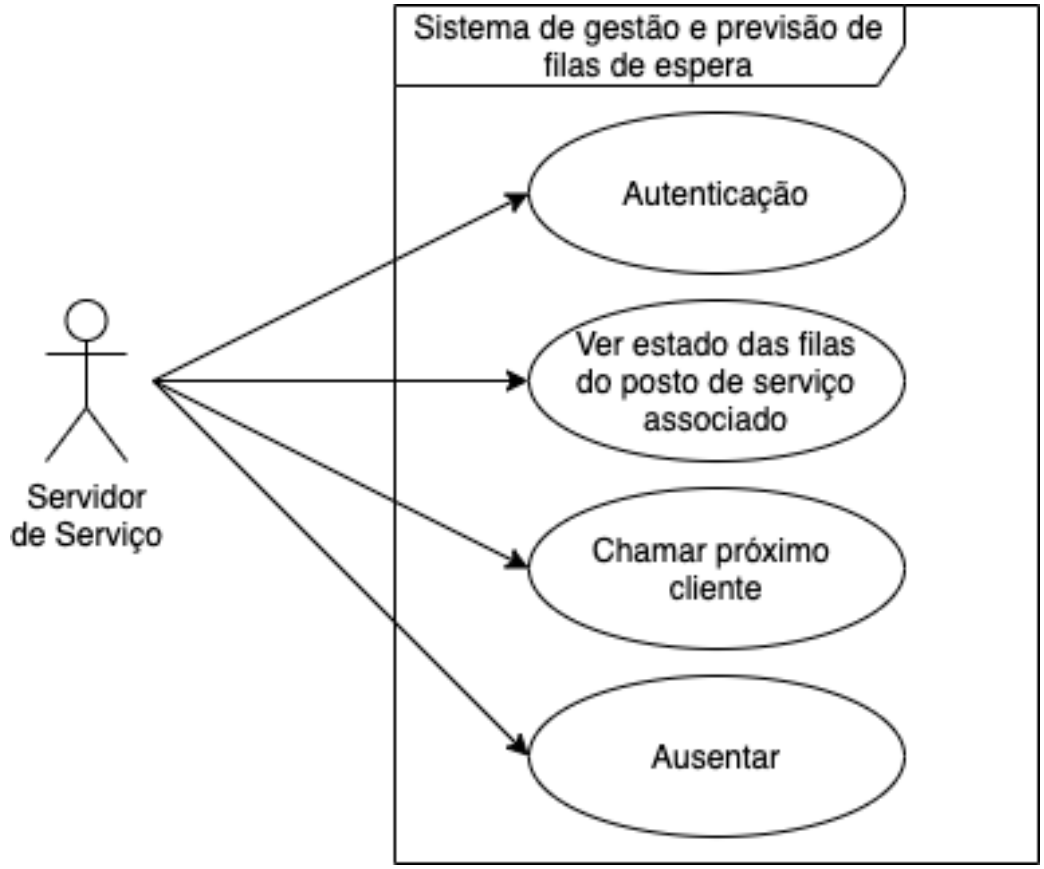

Figura 4.3: Casos de utilização do servidor de serviço

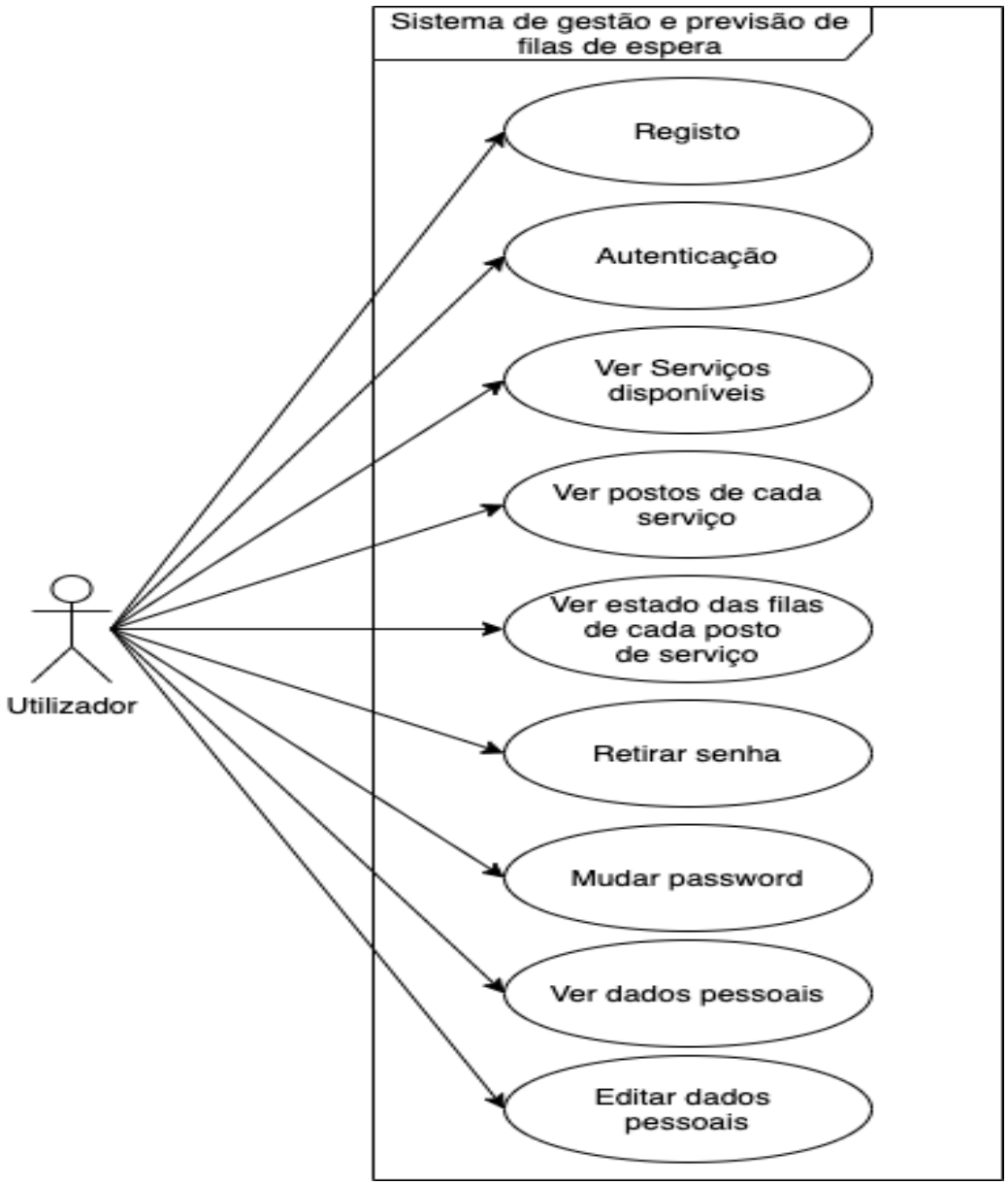

Figura 4.4: Casos de utilização do utilizador

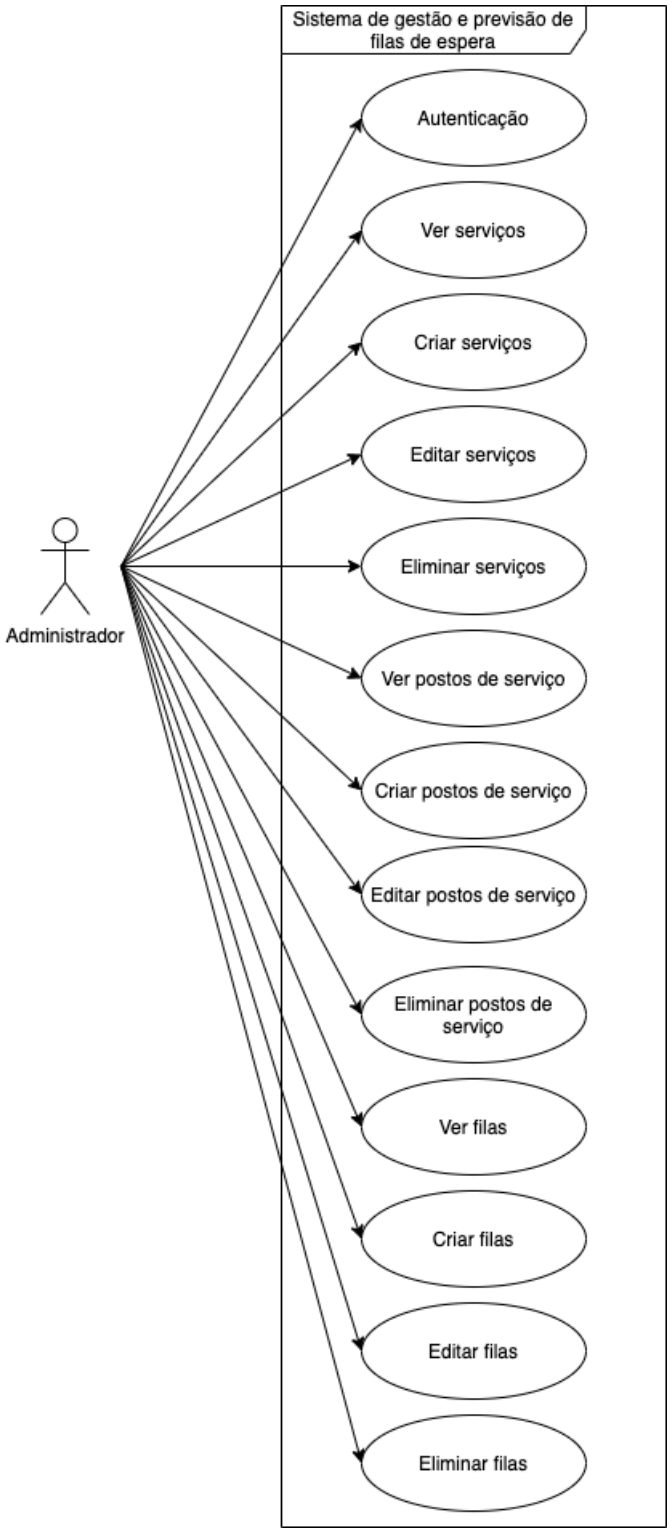

Figura 4.5: Casos de utilização do administrador

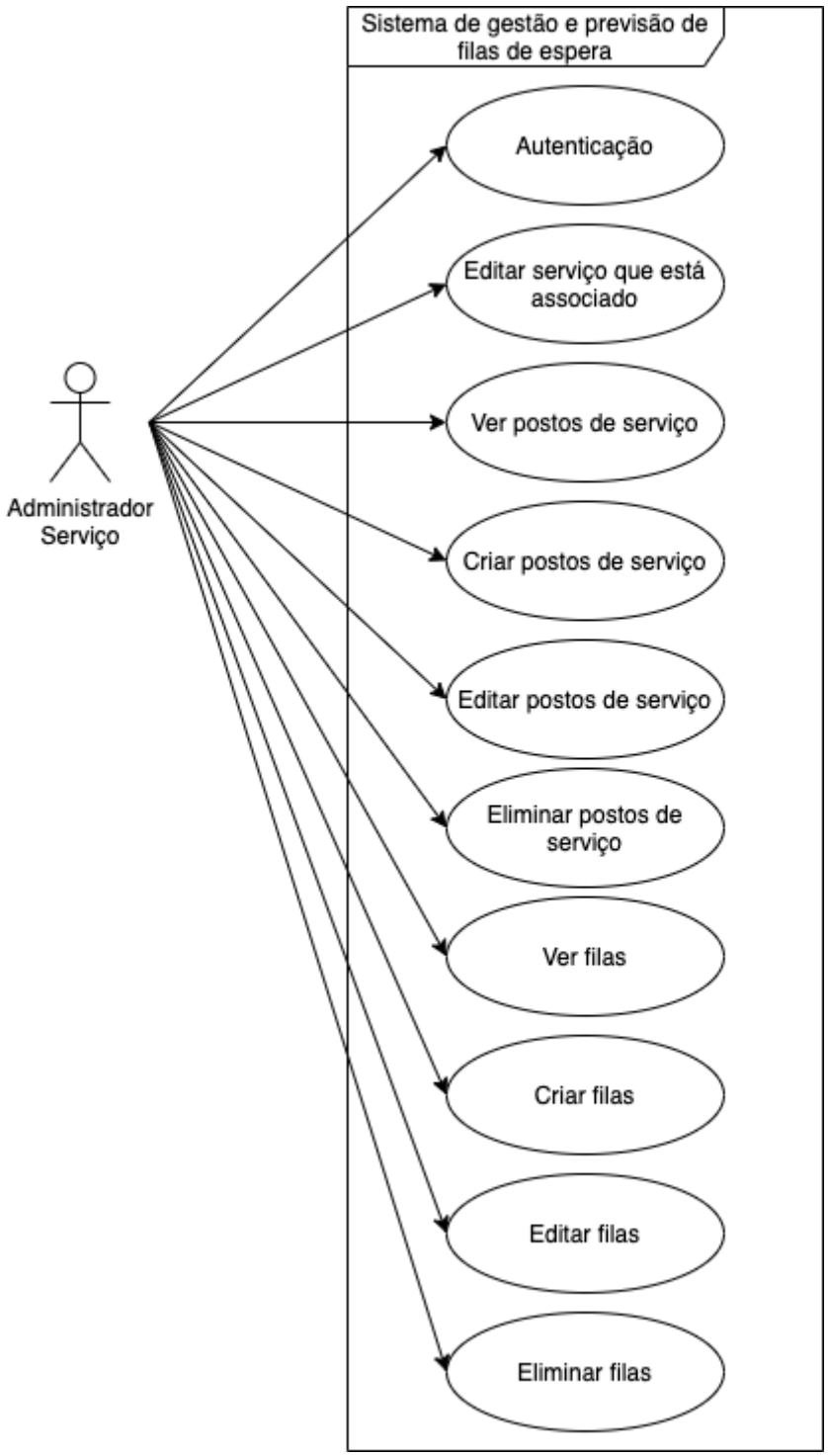

Figura 4.6: Casos de utilização do administrador de serviço

4.3.3 Descrição dos Casos de Utilização

Cada caso de utilização descreve o que pode acontecer (operação) no sistema e apresentado ao utilizador quando este realiza uma determinada ação.

Registo do Utilizador

Ator: Utilizador

Cenário Principal:

1. O utilizador inicia a aplicação móvel
2. A aplicação apresenta a janela de registo
3. O utilizador preenche com os dados requisitados
4. O sistema valida os dados. Se os dados forem válidos apresenta ao utilizador uma mensagem de sucesso

Cenário Alternativo A:

1. No passo 4, os dados não são válidos
2. O sistema apresenta ao utilizador os campos que não estão válidos

Autenticação do Utilizador

Ator: Utilizador, Administrador, Servidor e Administrador de serviço

Cenário Principal:

1. O utilizador inicia a aplicação móvel ou web
2. A aplicação apresenta a janela de autenticaçao
3. O utilizador preenche com os dados de autenticação (*username* e *password*)
4. O sistema valida os dados. Se forem válidos apresenta uma página com os serviços disponíveis

Cenário Alternativo A:

1. No passo 4, os dados não são válidos
2. A aplicação apresenta uma mensagem ao utilizador informando que a autenticação não é válida

Ver Serviços

Ator: Administrador e utilizador

Cenário Principal:

1. Incluir "Autenticação do Utilizador"
2. O sistema apresenta os serviços disponíveis

Criar Serviços

Ator: Administrador

Cenário Principal:

1. Incluir "Ver Serviços"
2. O utilizador seleciona a opção para criar serviços
3. O sistema apresenta um formulário para criação de um serviço
4. O utilizador preenche o formulário e submete.
5. O sistema processa o novo serviço.
6. O sistema apresenta os serviços disponíveis atualizado

Editar Serviços

Ator: Administrador e administrador de serviço

Cenário Principal - Administrador:

1. Incluir "Ver Serviços"
2. O utilizador seleciona o serviço que pretende editar
3. O sistema apresenta um formulário com informação editável do serviço
4. O utilizador edita o formulário e submete.
5. O sistema processa a edição do serviço e apresenta mensagem de sucesso.

Cenário Principal - Administrador de serviço:

1. Incluir "Autenticação do Utilizador"
2. O sistema apresenta um formulário com informação editável do serviço
3. O utilizador edita o formulário e submete.
4. O sistema processa a edição do serviço e apresenta mensagem de sucesso.

Eliminar Serviços

Ator: Administrador

Cenário Principal:

1. Incluir "Ver Serviços"
2. O utilizador seleciona o serviço que pretende eliminar
3. O utilizador requisita a eliminação do serviço e aprova.
4. O sistema processa a eliminação e apresenta os serviços disponíveis atualizado.

Cenário Alternativo A:

1. No passo 3, o utilizador rejeita.
2. O sistema permanece no serviço selecionado.

Ver Postos de Serviço

Ator: Administrador, administrador de serviço e utilizador

Cenário Principal:

1. Incluir "Ver Serviços"
2. O utilizador seleciona o serviço que pretende ver

3. O sistema apresenta a informação do serviço e os postos associados

Criar Postos de Serviço

Ator: Administrador e administrador de serviço
Cenário Principal:

1. Incluir "Ver Postos de Serviço"
2. O utilizador seleciona a opção para criar um posto
3. O sistema apresenta um formulário
4. O utilizador preenche o formulário e submete.
5. O sistema processa o novo posto.
6. O sistema apresenta o serviço com os postos disponíveis atualizado

Editar Postos de Serviço

Ator: Administrador e administrador de serviço
Cenário Principal:

1. Incluir "Ver Postos de Serviço"
2. O utilizador seleciona o posto pretendido
3. O sistema apresenta um formulário editável do posto de serviço
4. O utilizador edita o formulário e submete.
5. O sistema processa a edição do serviço e apresenta mensagem de sucesso.

Eliminar Postos de Serviço

Ator: Administrador e administrador de serviço
Cenário Principal:

1. Incluir "Ver Postos de Serviço"
2. O utilizador seleciona o posto pretendido
3. O sistema apresenta um formulário editável do posto de serviço
4. O utilizador requisita a eliminação do posto e aprova.
5. O sistema processa a eliminação e apresenta o serviço com os postos disponíveis atualizado.

Cenário Alternativo A:

1. No passo 4, o utilizador rejeita.
2. O sistema permanece no posto selecionado.

Ver Filas

Ator: Administrador, utilizador, administrador e servidor de serviço
Cenário Principal Administrador, Utilizador e Administrador de Serviço:

1. Incluir "Ver Postos de Serviço"

2. O utilizador seleciona o posto de serviço que pretende ver
3. O sistema apresenta a informação do posto serviço e as filas associadas

Cenário Principal Servidor de Serviço:

1. Incluir "Autenticação do Utilizador"
2. O sistema apresenta a informação do posto serviço associado ao utilizador e as filas associadas

Criar Filas

Ator: Administrador e administrador de serviço
Cenário Principal:

1. Incluir "Ver Filas"
2. O utilizador seleciona a opção para criar uma fila
3. O sistema apresenta um formulário
4. O utilizador preenche o formulário e submete.
5. O sistema processa a nova fila.
6. O sistema apresenta o posto serviço com as filas disponíveis atualizado

Editar Filas

Ator: Administrador e administrador de serviço
Cenário Principal:

1. Incluir "Ver Filas"
2. O utilizador seleciona a fila pretendida
3. O sistema apresenta um formulário editável da fila
4. O utilizador edita o formulário e submete.
5. O sistema processa a edição da fila e apresenta mensagem de sucesso.

Eliminar Filas

Ator: Administrador e administrador de serviço
Cenário Principal:

1. Incluir "Ver Filas"
2. O utilizador seleciona a fila pretendida
3. O sistema apresenta um formulário editável da fila
4. O utilizador requisita a eliminação da fila e aprova.
5. O sistema processa a eliminação e apresenta o posto de serviço com as filas disponíveis atualizado.

Cenário Alternativo A:

1. No passo 4, o utilizador rejeita.

2. O sistema permanece na fila selecionada.

Chamar próximo cliente

Ator: Servidor de serviço
Cenário Principal:

1. Incluir "Ver Filas"
2. O utilizador realiza a ação chamar próximo cliente caso a tolerância já tenha passado.
3. O sistema faz a chamada, alterando o estado da fila

Cenário Alternativo A:

1. No passo 3, o sistema dá uma mensagem a indicar que ainda não pode chamar o próximo cliente

Ausentar

Ator: Servidor de serviço
Cenário Principal:

1. Incluir "Autenticação do Utilizador"
2. O utilizador realiza a ação para ausentar do posto de trabalho

Regressar

Ator: Servidor de serviço
Cenário Principal:

1. Incluir "Autenticação do Utilizador"
2. O utilizador realiza a ação para regressar ao posto de trabalho

Retirar Senha

Ator: Utilizador
Cenário Principal:

1. Incluir "Ver Filas"
2. O utilizador seleciona a fila pretendida
3. O sistema apresenta a fila ao utilizador
4. O utilizador realiza a ação de obtenção de senha
5. O sistema processa o pedido, criando serviços para apresentar ao utilizador a atualização do estado das filas

Ver dados pessoais

Ator: Utilizador
Cenário Principal:

1. Incluir "Autenticação do Utilizador"

2. O utilizador seleciona o menu/definições
3. O sistema apresenta os dados do utilizador

Editar dados pessoais

Ator: Utilizador
Cenário Principal:

1. Incluir "Ver dados pessoais"
2. O utilizador edita os dados editáveis e submete
3. O sistema dá mensagem de sucesso

Mudar Password

Ator: Utilizador
Cenário Principal:

1. Incluir "Incluir "Ver dados pessoais"
2. O utilizador escolhe a opção para alterar a password
3. O utilizador edita a password e submete
4. O sistema dá mensagem de sucesso

4.3.4 Diagrama de Sequência

Na Figura 4.7, é apresentado o diagrama de sequência na perspectiva do utilizador, que permite a entrada na aplicação para autenticação assim como a obtenção de uma senha num posto de serviço, conseguindo visualizar o tempo de espera previsto.

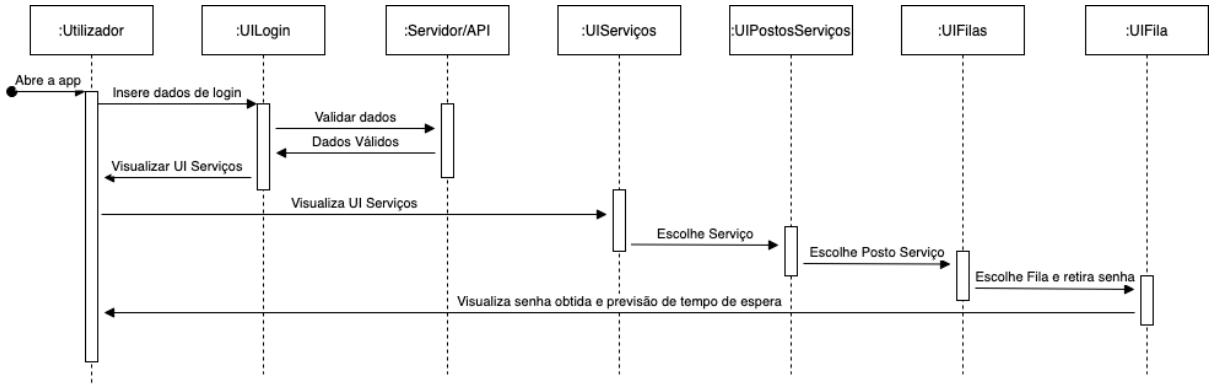

Figura 4.7: Diagrama de sequência

O utilizador ao abrir a aplicação móvel é lhe apresentado o ecrã de login, onde deve indicar as credenciais corretas de forma a realizar a autenticação no sistema. Ao realizar o pedido de autenticação, os dados são enviados ao servidor de forma a realizar a validação. Em caso de sucesso o utilizador é reencaminhado para o ecrã onde consegue

ver os serviços disponíveis, caso contrário é apresentado uma mensagem indicando que as credenciais não estão corretas. O utilizador no ecrã dos serviços, tem a possibilidade de escolher o pretendido, sendo apresentado os postos que o serviço tem para os atendimentos. Em caso de escolha de um dos postos, é apresentado as filas com detalhes como tempo de espera previsto, nome/descrição da fila e o número que está a ser atendido. O utilizador pode escolher uma das filas e retirar uma senha, onde passa a receber atualizações em tempo real sobre o estado da fila.

4.4 Modelo Domínio

Na Figura 4.8, é apresentado o modelo sobre a qual o sistema se baseia.

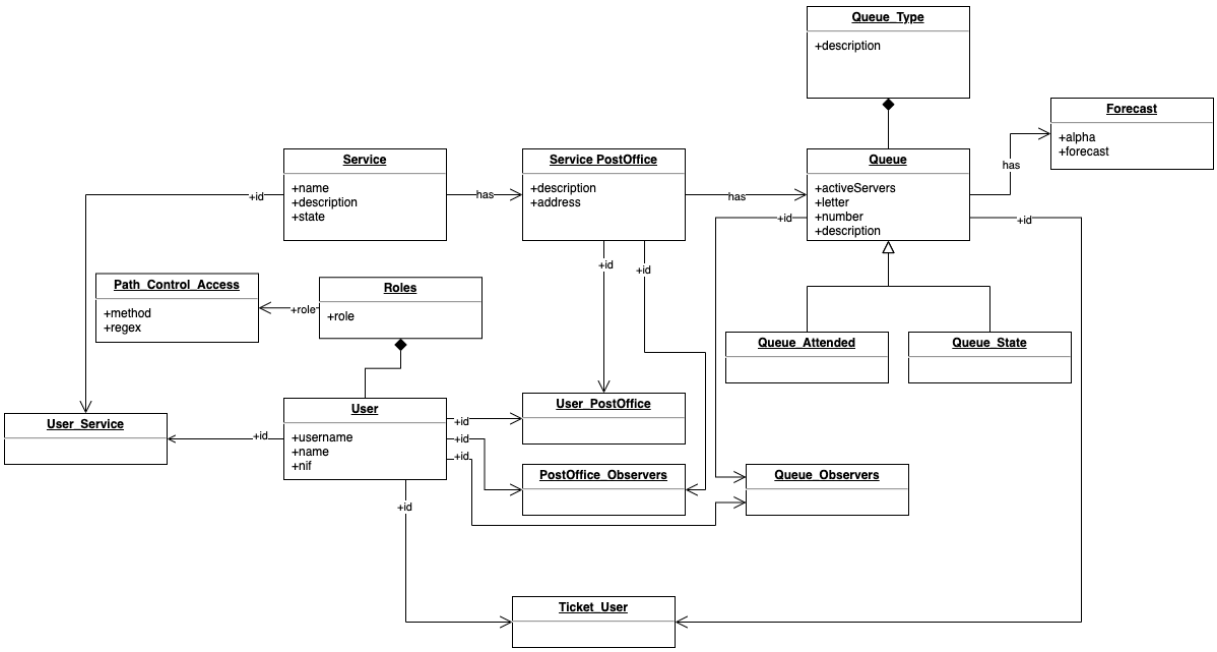

Figura 4.8: Modelo de Domínio

O modelo representado na Figura 4.8, apresenta as entidades resultantes dos requisitos do projeto. Para simplificação, não é incluído todas as caracteristicas/propriedades das entidades. Cada entidade representa um tipo de informação do domínio do problema. A entidade ***User***, com os atributos, *name*, *username*, *nif*, representa todos os utilizadores que fazem parte do sistema. Recorrendo à agregação com ***Roles*** é possível diferenciar o papel que cada utilizador pode ter no sistema, conseguindo apenas realizar as operações autorizadas para cada papel de forma a garantir os comportamentos corretos. Com a entidade ***Path_Control_Access***, é possível definir as regras de acesso para cada

tipo de utilizador. Com as entidades ***User_Service*** e ***User_PostOffice***, um administrador poderá definir em que serviços e postos de serviços é que cada utilizador pode realizar operações sobre, desde que tenha o papel correto para o efeito. ***Service*** representa os serviços institucionais, tendo cada serviço um conjunto de postos de serviço (***Service_PostOffice***), onde um utilizador pode realizar o tratamento dos assuntos de interesse. Para tratamento de assuntos em cada posto de serviço existe um conjunto de filas, representado pela entidade ***Queue*** que podem ter propósitos diferentes e prioridades diferentes, representado por ***Queue_Type***, o que permite ter regras apropriadas na gestão das senhas dos utilizadores. De forma a gerir os estados de atendimento e espera das senhas, é utilizado as entidades ***Queue_Attended*** e ***Queue_State***, respetivamente. Para realizar o cálculo da previsão de espera para cada fila, a entidade ***Forecast*** é utilizada. Para auxiliar na previsão de tempos e ter registo da associação entre utilizador e senha, é utilizado o ***Ticket_User***. Com esta entidade é possível ter o registo do tempo de entrada e de saída aproximada no atendimento do cliente, o que permite de acordo com as regras a melhor previsão para os clientes da fila. As entidades ***Queue_Observers*** e ***PostOffice_Observers*** são utilizadas para fornecer ao utilizador a atualização do estado das filas nos postos de serviço.

5

Arquitetura do Protótipo

Neste capítulo é apresentada a arquitetura definida para concretizar o protótipo.

5.1 Desenho

Nesta secção está detalhado o desenho do sistema.

5.1.1 Componentes do Sistema

O sistema, ilustrado na Figura 5.1, é constituído pelas seguintes componentes:

- Aplicação móvel
- Servidor/Web API
- Firebase Cloud Messaging [3]
- Simulador de Serviço

A **aplicação móvel**, implementada em Android, corresponde à componente que define a aplicação cliente do sistema. Esta componente é responsável por fornecer uma interface gráfica de suporte às funcionalidades disponibilizadas pelo sistema. De forma a receber atualizações em tempo real e notificações a aplicação utiliza a API do *Firebase Cloud Messaging*.

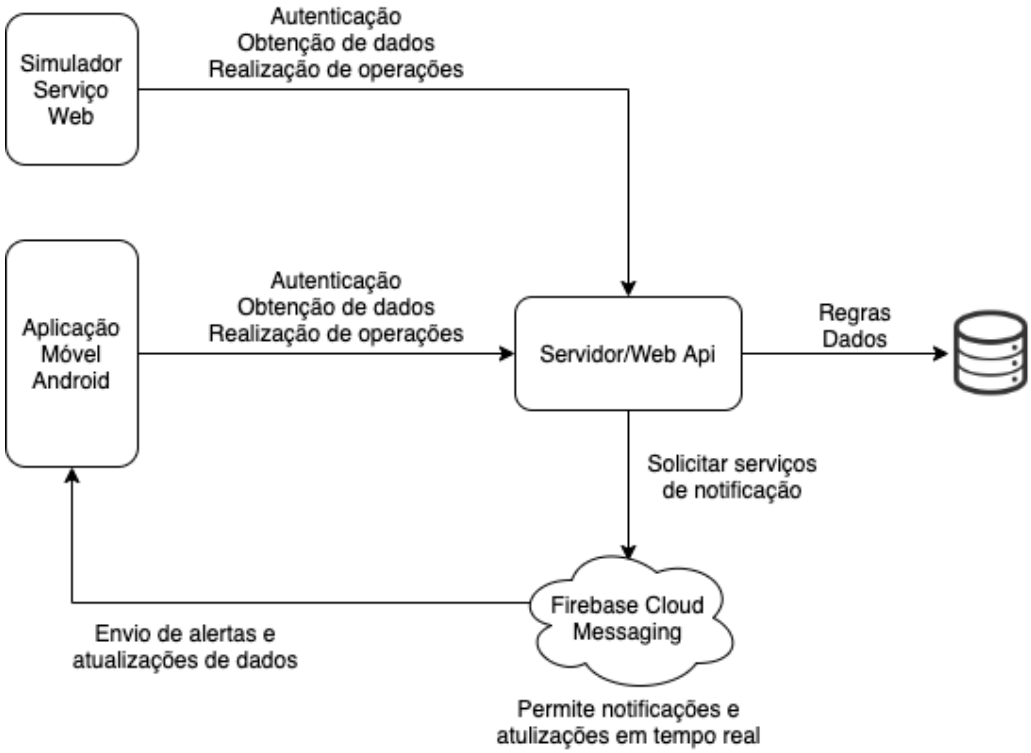

Figura 5.1: Arquitetura do sistema

A componente **Servidor/Web API** é a responsável pela gestão das senhas do sistema. Ela define as operações que permite aos utilizadores a observação e obtenção de senhas de interesse. O acesso às operações suportadas é realizada por meio da API disponibilizada pelo servidor. O envio de notificações e de atualizações em tempo real é realizado através da interação com os servidores *Firebase Cloud Messaging*. Por estar numa fase de prototipagem, esta componente gere de forma direta a autenticação de utilizadores no sistema. Para persistência de dados, é utilizada uma base de dados PostgreSQL.

Firebase Cloud Messaging (FCM), é uma solução *cross-platform* em nuvem, que fornece ao sistema soluções/serviços que permitem enviar mensagens e notificações de forma confiável às componentes clientes do sistema. A plataforma **Firebase** disponibiliza toda a infraestrutura para oferecer o serviço. Para delegar o *push* de mensagens e notificações às componentes cliente é utilizado o protocolo HTTP.

O **Simulador de serviço**, concretizado em Flutter, corresponde à componente que faz a simulação daquilo que acontece nos postos de serviço, no que diz respeito à mudança de estados das filas e permite a gestão dos serviços, postos de serviços e as filas, funcionando como um *backoffice*.

Cada seta apresentada na Figura 5.1, representa o fluxo/interação entre cada componente do sistema.

5.1.2 Interação entre Componentes

Existem várias interações entre as componentes do sistema, por forma a capacitar a disponibilização das funcionalidades desenvolvidas.

O sistema utiliza o modelo **cliente-servidor**, com o servidor responsável pela disponibilização de uma API que permite ao cliente a realização de operações. Este serviço, em conjunto com a aplicação cliente móvel ou simulador de serviço, disponibiliza ao utilizador uma interface gráfica, que permite a realização das operações e acesso a recursos.

Por forma a permitir a apresentação do estado das filas de determinados serviços e anunciar ao utilizador que a sua vez de ser atendido se aproxima, é utilizado o mecanismo de notificações em tempo real, com base no modelo *push notifications* [12]. Para capacitar ao sistema com este recurso, utiliza-se a componente FCM como serviço externo. A interação das componentes funciona com o padrão de *publish-subscribe* [15], em que o utilizador através da aplicação, subscreve a atualizações de forma explícita, ou seja, quando obtém uma senha de um serviço, e de forma implícita, ao entrar num ecrã que apresenta os estados das filas em que tem interesse. Ao sair do ecrã ou da aplicação a subscrição é retirada. O servidor é a componente que tem a informação sobre as subscrições efetuadas, realizando o envio de notificações/mensagens, recorrendo ao serviço externo FCM. O envio é realizado quando existe o desencadeamento de eventos de interesse dos utilizadores.

5.2 Implementação

Nesta secção é detalhada a implementação dos componentes que permitem o funcionamento do sistema.

O código fonte do sistema está presente no repositório:
`https://github.com/Pompeu208/isel-tfm-sgpfe-rn.git`.

5.2.1 Componentes

O transporte de dados entre componentes, é feito utilizando o formato ***hypermedia Siren [17]***, que permite a utilização de *link relations* e *actions*. Com este formato, a componente cliente não precisa de saber *à priori* todas as operações permitidas pelo sistema, recebendo nas respostas da API que tipos de ações é que se pode realizar sobre determinado recurso.

A **Application Programming Interface (API)** disponibilizada pelo servidor, que define as operações existentes no sistema e faz a gestão das senhas foi desenvolvida utilizando a *framework* ***Spring Boot***, com ***Kotlin*** a ser utilizado como linguagem de programação. ***Spring Boot*** apenas facilita a configuração *Spring*, sendo ***Spring MVC*** [19] a *framework* que oferece a arquitetura utilizada no servidor, *Model–View–Controller (MVC)* [9]. A componente, de forma a receber pedidos externos, utiliza o conceito de ***Controller***, para a definição dos *handlers* para cada recurso do sistema. Com este conceito, foram desenvolvidos *controllers* relacionados com o recurso utilizador e senha, de forma a permitir a entrada no sistema de forma segura, e permitindo aos utilizadores e serviços externos realizarem operações que permitem a gestão das senhas disponibilizadas pelos postos dos serviços. É utilizado o conceito de ***Services***, de forma a definir as interfaces e respetivas implementações que permitem ter uma camada com a lógica de negócio. De forma a aceder aos dados presentes nas entidades do modelo físico, é implementado em Spring uma camada de acesso a dados (DAL). A camada de acesso a dados foi implementada, utilizando o *Spring Java Database Connectivity (JDBC)* [7], que permite flexibilidade e controlo sobre o mapeamento das entidades e sobre *queries* executadas.

Na Figura 5.2, é possível observar os conceitos utilizados na componente servidor.

Figura 5.2: Conceitos utilizados no servidor

No servidor, foram utilizados métodos assíncronos, recorrendo à ativação e configuração do processo assíncrono *Spring*, que permite detetar as operações assíncronas utilizando a anotação @Async. Na configuração do processo, é criado um executor de tarefas, definindo o número de *threads* sempre disponíveis para receber tarefas, o número máximo de *threads* que podem ser criadas para processamento, assim como o máximo de tarefas que podem estar numa fila à espera para serem processadas. Quando o contexto do *Spring* é inicializado, marca os métodos que têm a anotação @Async como assíncronos, o que permite utilizar um gestor de tarefas quando um método marcado é invocado, para ser executado numa nova *thread* em pararelo. Com isso é possível ter respostas mais rápidas ao cliente, realizando o processamento de outras ações mais complexas em *threads* diferentes. O cálculo da previsão de tempo de espera e o envio de notificações, são realizadas em processos assíncronos visto que são mais demorados, pela necessidade de obtenção de mais dados e consequente processamento, e por ser necessário no caso do envio de notificações, a comunicação com o serviço externo

FCM.

Na camada de acesso a dados, foi utilizada uma cache simples o que permite obter dados que não são alterados constantemente (e.g. sessão) de forma mais rápida, sem a necessidade de realizar um pedido à base de dados. Tendo em conta que se está no contexto de uma prova de conceito, não é realizada a implementação de um mecanismo que permite inserir dados com um tempo de vida e remove-los quando o tempo expira.

A componente servidor, à medida que foi desenvolvida, foi-se realizando o ***deployment*** de versões estáveis para a ***cloud***. Para hospedagem, foi utilizada a plataforma Heroku. O *deployment* é realizado de forma automática, recorrendo a *pipelines* presentes no repositório git do projeto, tendo sido necessário realizar configurações do lado do Heroku e do Bitbucket (onde se encontra o repositório do projeto) e a criação de um ficheiro bitbucket-pipelines.yml. No ficheiro estão indicados os passos a serem dados quando é detetado que houve *commits* no *branch master* (versão estável), sendo elas, a criação de um artefato que contém o projeto em *Spring Boot* e o envio para o Heroku. A plataforma de hospedagem ao receber o artefacto faz o *build* do projeto e em caso de sucesso, ocorre o *deploy*, ficando automaticamente disponível para utilização por parte dos utilizadores. O servidor de base de dados está também hospedado na plataforma Heroku.

A **Aplicação Móvel** cliente do sistema implementada em Android, utilizando a arquitetura recomendada para desenvolvimento Android [1], baseada no padrão *Model–View–ViewModel (MVVM)* [10], fornece ao utilizador uma interface que permite a realização da autenticação no sistema, sendo apresentado um conjunto de serviços, em caso de sucesso. O utilizador pode escolher o serviço pretendido, sendo apresentado todos os postos de serviço disponibilizados pela entidade institucional. Em cada posto de serviço, é possível observar um conjunto de filas, que permite ao utilizador a visualização do estado das filas assim como o tempo de espera previsto. O estado das filas é atualizado em tempo real, ao receber *push notifications* com informação atualizada. Isto é possível com uma subscrição implícita sobre o posto por parte da aplicação. Ao selecionar uma fila, o utilizador pode optar por retirar uma senha. Com a obtenção de uma senha, a aplicação recorre a serviços *Android* de forma a disponibilizar ao utilizador uma notificação que apresenta o estado da fila e do tempo de espera. À medida que a aplicação vai recebendo atualizações do servidor em combinação com FCM, o serviço recebe eventos de forma a manipular a notificação com os dados corretos.

A aplicação que serve de **simulador de serviço**, desenvolvido em Flutter Web, utiliza o mesmo padrão (MVVM) que a aplicação móvel, tendo três modos de utilização,

sendo eles administrador, administrador de serviço e servidor de serviço. Cada modo, depende do tipo de utilizador que realiza a autenticação, com o simulador a receber informação suficiente de forma a encaminhar o utilizador. Os modos estão separados em módulos no projeto, onde cada um contêm as interfaces gráficas e gestores de estados específicos, permitindo uma navegação diferente para cada tipo de utilizador.

5.2.2 Funcionalidades

Nesta secção é descrito as funcionalidades base do sistema.

5.2.2.1 Registo

O registo de utilizadores é diferente para cada tipo de utilizador do sistema, sendo que o administrador é que faz a gestão dos administradores e servidores de serviços, não tendo a possibilidade de realizar um registo autónomo. No entanto, o utilizador cliente da aplicação móvel pode realizar o registo, utilizando a aplicação, fornecendo os dados necessários para o registo com sucesso. Por este projeto ser uma prova de conceito, a ação é realizada de forma simples, sem a necessidade de confirmação por parte do cliente (e.g. confirmação por email) e sem aplicar o Regulamento Geral sobre a Proteção de Dados (RGPD) [16]. Visto que existe recolha e tratamento de dados do utilizador, num cenário real seria necessário aplicar o RGPD no projeto.

5.2.2.2 Autenticação e Autorização

A autenticação e autorização é realizada pelo próprio servidor sem a utilização de protocolos e servidores de autenticação externos, o que permite ter um sistema com maior autonomia. O utilizador ao estar autenticado, utiliza um *token* de acesso, gerado e fornecido pelo sistema, para realizar as operações pelas quais está autorizado. De forma a garantir maior flexibilidade do sistema, esta componente integrada com outros serviços de identificação de utilizadores, permitindo assim acessos sem a necessidade de realizar um novo registo. A autorização é realizada recorrendo às entidades *User_Service*, *User_Post_Office* e *Path_Control_Access*.

No JSON abaixo é possível ver a estrutura enviada pelo servidor para as aplicações clientes no momento da atualização. O nome é utilizado meramente para dar as boas vindas ao utilizador e indicar as iniciais quando o utilizador entra no menu da aplicação móvel. A indicação do *role*, permite identificar o tipo de utilizador e enviar para

o módulo correto no simulador de serviço. O *token*, será utilizado em todas as operações que requerem o utilizador autenticado, sendo utilizado também para validar se o utilizador está autorizado para realizar a operação. O *userId* e *role* são utilizados apenas para simplificar e permitir operações mais rápidas, sem a necessidade de haver introspeção e pedidos extras no servidor para obtenção da informação para realização de ações.

```
{
    "class": [
        "session"
    ],
    "properties": {
        "name": "Ronilton Neves",
        "role": "ADMIN",
        "token": "f60cc0c2-a975-4c4b-8de1-c8e6f05754a5",
        "userId": 1
    }
}
```

5.2.2.3 Visualização de Estado e Obtenção de Senhas

O utilizador após a autenticação com sucesso consegue ver os serviços e os postos de serviços disponíveis, assim como as filas de cada posto. Ao entrar num posto de serviço, é realizado uma subscrição implícita de forma automática, o que representa uma declaração de interesse sobre os estados das filas do posto de serviço. Ao deixar de observar a informação de um posto a subscrição é retirada de forma automática. Durante o período em que o utilizador tem a informação disponível, é atualizado das alterações em tempo real.

O utilizador tem a opção para escolher uma fila do serviço e requisitar uma senha, ficando subscrito de forma explícita, com a aplicação móvel a lançar um serviço em primeiro plano que vai apresentando o estado da fila e o tempo de espera previsto, permitindo ao utilizador fechar a aplicação, e continuar a receber as atualizações até ser atendido.

No objeto JSON abaixo é possível observar um exemplo da estrutura que contém o estado das filas de espera de um posto de serviço. Neste exemplo é possível observar que o posto de serviço, tem duas filas (A e B), tendo o valor de atendimento presente

na propriedade *attendedNumber*, a última senha retirada *stateNumber*, com *name*, *letter* e *queueId* sendo as informações da fila e por fim a previsão do tempo de espera na propriedade *forecast*.

```
{
    "class": [
        "collection",
        "queueStates"
    ],
    "properties": {
        "queueStates": [
            {
                "class": [
                    "queueState"
                ],
                "properties": {
                    "stateNumber": 13,
                    "name": "Queue A",
                    "letter": "A",
                    "queueId": 4,
                    "attendedNumber": 5,
                    "forecast": 134
                }
            },
            {
                "class": [
                    "queueState"
                ],
                "properties": {
                    "stateNumber": 5,
                    "name": "Queue B",
                    "letter": "B",
                    "queueId": 5,
                    "attendedNumber": 5,
                    "forecast": 0
                }
            }
        ]
```

```
35        }
36    }
```

5.2.2.4 Atualizações em tempo real

Nesta secção está descrito o processo que permite a atualização em tempo real do estado das senhas. Na Figura 5.3 é possível observar o esquema de atualização.

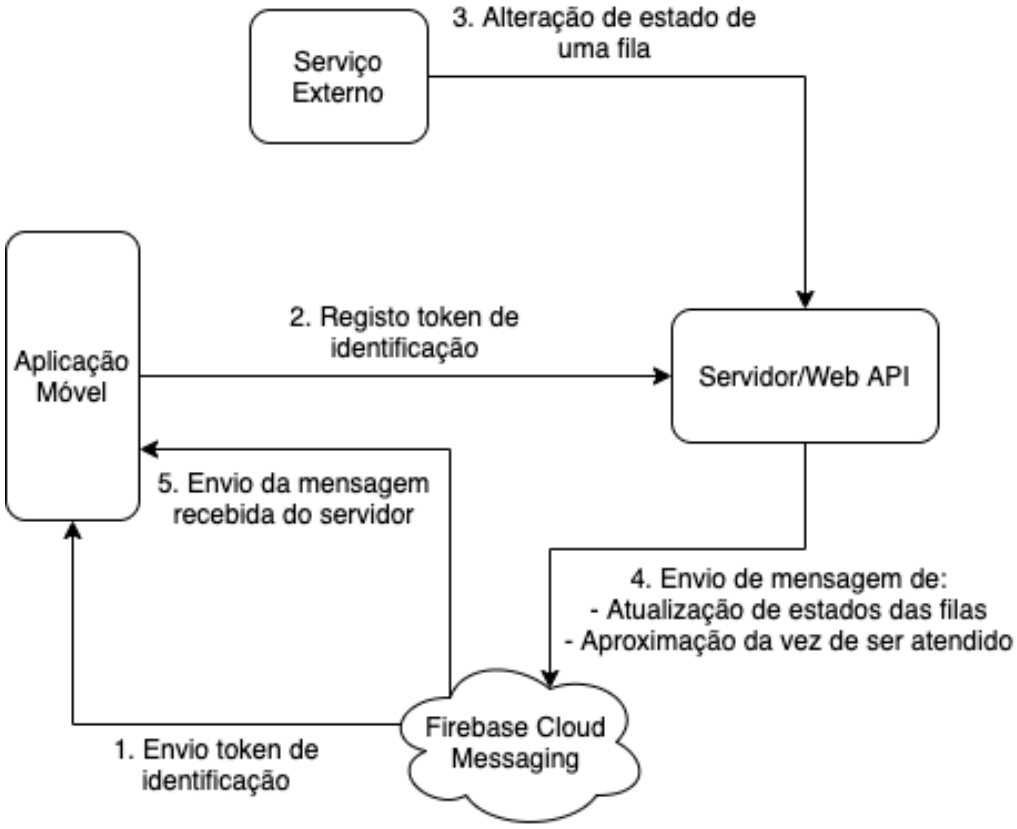

Figura 5.3: Esquema da atualização em tempo real

Da primeira vez que o utilizador inicia a aplicação móvel, um *token* de identificação que associa um dispositivo à aplicação é gerado pelo serviço de FCM, sendo enviado ao servidor quando o utilizador realiza a primeira autenticação, de modo a permitir enviar notificações/mensagens para os utilizadores corretos. No passo 1 e 2 da Figura 5.3 é possível verificar o registo do *token*. O processo de geração e consequente registo, pode ser repetido diversas vezes, visto que existe um mecanismo de rotação de *tokens*, sendo atualizado pelo FCM e enviado para a aplicação móvel.

Com o registo do *token* e as subscrições sobre serviços e senhas, o utilizador fica habilitado a receber atualizações em tempo real, com o desencadeamento dos passos 3,

4 e 5. Num serviço externo ao sistema, ao ser alterado o estado da fila, este tem que informar o sistema, permitindo a este anunciar aos interessados o evento. O servidor, de acordo com a informação recebida, realiza um processamento de forma a identificar quem recebe a atualização do estado e quem recebe informação sobre aproximação da vez de ser atendido. Com o processamento completo, os dados são enviados para o FCM, com a identificação (*token*) de cada utilizador, permitindo o FCM a entrega ao destinatário correto da informação recebida.

No JSON abaixo é possível verificar um exemplo do formato de dados que é enviado a um utilizador que tenha uma subscrição sobre uma fila.

```
{
    "type": "ticket_for_list",
    "queueId" : 1,
    "letter" : "A",
    "stateNumber" : 5,
    "attendedNumber" : 2,
    "name": "Queue A",
    "forecast" : 120
}
```

A aplicação cliente pode receber diferentes formatos de mensagens do FCM, sendo responsável por realizar o processamento adequado de forma a apresentar corretamente os dados ao utilizador.

5.2.3 Modelo de Dados

O modelo de dados utilizado pelo sistema, é baseado no diagrama Entidade Associação (EA) apresentado na Figura 5.4. De forma a garantir a persistência dos dados utilizados, este utiliza um servidor de base de dados PostgreSQL.

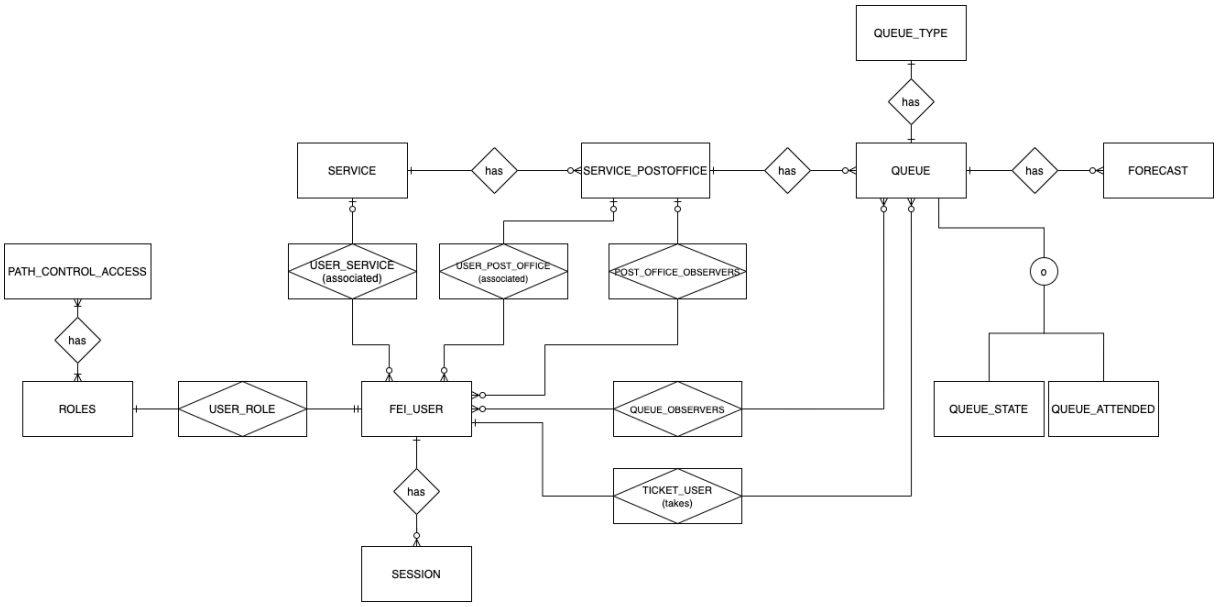

Figura 5.4: Diagrama EA do modelo de dados

Fei_user(*id*, *username*, *name*, *nif*, *password*, *salt*, *fcm_token*, *phone*, *email*, *state*) - Entidade que contém os dados dos utilizadores do sistema.

Service(*id*, *name*, *description*, *state*) - Entidade que tem a informação dos serviços integrados no sistema.

Service_Postoffice(*id*, *latitude*, *longitude*, *description*, *serviceId[FK]*, *address*, *state*) - Tem informação de cada posto pertencente a um determinado serviço.

Queue_Type(*id*, *description*, *state*) - Tem a informação sobre os tipos de fila.

Queue(*id*, *activeServers*, *letter*, *name*, *description*, *type*, *maxAvailable*, *servicePostOfficeId[FK]*, *tolerance*, *state*) - Tem a informação relaciona com as filas de cada posto de serviço.

Queue_State(*id*, *number*, *letter*, *queueId[FK]*) - Tem a informação do estado atual das filas.

Queue_Attended(*id*, *number*, *letter*, *queueId[FK]*) - Tem a informação do estado de atendimento das filas por parte dos servidores.

Ticket_User(*id*, *queueId[FK]*, *number*, *userId[FK]*, *createdAt*, *attended*, *duration*, *state*, *canceled*, *forecast*, *updatedAt*) - Esta entidade associação, é utilizada quando um utilizador obtém a senha de um determinado serviço.

Session(*id, userId[FK], token, createdAt*) - Entidade responsável por armazenar as sessões dos utilizadores logados.

Forecast(*id, queueId[FK], alpha, flimit, forecast*) - Entidade que contêm a informação que deve ser aplicada a cada fila de espera para obtenção da previsão.

Roles(*id, role*) - Tem as funções dos utilizadores do sistema.

User_Role(*userId, role*) - Contêm a associação entre utilizador e função.

Queue_Observers(*id, queueId[FK], userId[FK]*) - Entidade associação responsável por armazenar os observadores de cada fila.

Post_Office_Observers(*id, postId[FK], userId[FK]*) - Entidade associação responsável por armazenar os observadores de cada posto de serviço.

User_Service(*userId, serviceId[FK]*) - Tem a informação de que serviços é que um utilizador pode aceder.

User_Post_Office(*userId, servicePostOfficeId[FK]*) - Tem a informação de que postos de serviço é que um utilizador tem acesso.

Path_Control_Access(*id, method, regex, roles*) - É a entidade onde está definido as regras de acesso de cada função no sistema.

Em algumas entidades do modelo, é possível verificar o atributo *state* (tipo *bit*) que serve para representar se um registo está válido ou não, permitindo assim manter histórico sem a necessidade da criação de novas entidades. Com isso, ao ocorrer uma tentativa de eliminação de um recurso, o sistema apenas faz a atualização do atributo *state* para zero (0) e o recurso deixa de estar visível.

De forma a auxiliar na inserção e atualização de dados, foram implementadas algumas funções, *procedures* e *triggers*. Um dos *procedures* existentes é o *SetUser_Ticket_Attended* que é responsável por indicar que um cliente já foi atendido, realizando também o cálculo do tempo de atendimento, que é utilizado posteriormente para o cálculo de previsão de tempo da fila. As funções *Create_Queue_Tables* e *Update_Queue_Tables* são utilizadas por *triggers* de forma a inserir e atualizar as entidades *Queue_State* e *Queue_Attended* quando existe uma inserção ou atualização na entidade *Queue*. Uma outra função importante é o *Get_Next_In_Line* que permite obter o próximo cliente na fila, para que este seja notificado que a vez de ser atendido se aproxima.

6

Conclusões e Trabalho Futuro

Neste capitulo são apresentadas, de forma sintética, as conclusões do trabalho realizado.

São apresentadas ainda, algumas propostas para futuros desenvolvimentos.

6.1 Conclusões

O presente projeto foi realizado com o objetivo de desenvolver um sistema que permite a gestão centralizada de vários serviços de forma a otimizar o tempo gasto em filas de espera e nos atendimentos dos serviços institucionais.

O projeto visa suportar diversos serviços em simultâneo, o que requer bastante processamento em paralelo (e.g. cálculos de previsão diferentes para cada tipo de fila dos vários serviços) bem como grande capacidade de armazenamento. No entanto, não é realizado o estudo de viabilidade sobre a capacidade e disponibilidade do sistema, visto que o projeto está num contexto de prototipagem, onde não existe a necessidade para o dimensionamento.

Para que o sistema seja bem sucedido, a adesão de clientes e instituições é importante. A utilização do sistema proposto apresenta, para as instituições, vantagens em termos de custos, permitindo a cada continuar com o estabelecimento de *trade-offs* entre o custo do serviço e o custo do tempo despendido pelos clientes na fila de espera. A gestão realizada pelo sistema permite uma melhoria nos atendimentos sem recursos a

grandes custos, comparado com o investimento que seria necessário fazer em recursos humanos e materiais, pelas instituições para os vários postos de serviço, permitindo ainda filas de espera mais eficazes, sem grandes aglomerações, contribuindo assim para aumentar a produtividade e satisfação dos diversos clientes.

A utilização de uma abordagem de prototipagem evolutiva foi importante durante o processo de desenvolvimento, com a implementação de cada funcionalidade por completo. Assim foi possível minimizar os riscos do projeto, tendo sido feitas pequenas correções/adaptações, visando melhorias/otimizações no sistema.

De forma a melhorar os dados de entrada nas métricas, como por exemplo o *alpha* (α) utilizado no cálculo exponencial, é sempre necessário haver uma extração dos dados para componentes ou aplicações externas que permitem a análise dos mesmos, de modo realizar testes e verificações para alterar os valores manualmente. Assim sendo, o método *Moving Average*, para a previsão de tempos, pode devolver resultados mais precisos aos utilizadores do sistema.

Este projeto utiliza o simulador de serviço, para substituir e permitir simular o funcionamento dos sistemas de senhas utilizados pelos serviços institucionais. Assim sendo, o comportamento do sistema deverá permanecer igual após integração com diversos sistemas de senhas. No entanto isto faz com que seja sempre necessário o desenvolvimento de novas componentes/interfaces para ligar os serviços ao sistema.

6.2 Trabalho Futuro

Com o projeto finalizado, identificam-se alguns aspetos que podem ser implementados no sistema de forma a torná-lo melhor e mais robusto.

Apesar do projeto estar focado em serviços institucionais, a evolução do sistema, permite abrir portas para outros tipos de serviços que recorrem a senhas e filas de espera para satisfazer aquilo que os clientes pretendem. Por exemplo, para serviços como a restauração (que permitem take-away), pode-se ter uma componente que permita a realização de pagamentos através da aplicação cliente do sistema.

Para além das funcionalidades já disponibilizadas, recorrendo à utilização da localização geográfica, pode-se sugerir a um cliente a que posto de serviço é que deve ir. Isto permite realizar uma distribuição equilibrada de clientes para os diversos postos.

Outra funcionalidade seria a troca de senha entre clientes que, recorrendo também à localização geográfica e o número da senha de cada cliente, pode identificar os indivíduos ideais para a troca, com duas vertentes:

1. O utilizador solicita a troca caso não seja possível chegar a tempo.
2. O sistema detetar que o utilizador não consegue chegar a tempo para o atendimento.

Em relação à primeira vertente, poderia ser possível oferecer ao utilizador a possibilidade de indicar a estimativa de tempo de chegada ao serviço. Com esta possibilidade, existe sempre o risco de não haver cliente na fila, surgindo assim a oportunidade para o desenvolvimento de uma fila de espera que permite ao sistema de forma automática retirar senhas em nome do utilizador para o momento ideal. Na segunda vertente, seria necessário o acesso à localização geográfica do cliente de forma a detetar um possível atraso.

Recorrendo a algoritmos de análise de dados e de apoio à decisão, será possível desenvolver componentes integrantes do sistema que, de acordo com os resultados, altere dados de entrada nas métricas de forma automática visando conseguir os melhores resultados, sem intervenção humana.

Referências

[1] (2020). Guide to app architecture | android developers @ONLINE, URL: https://developer.android.com/jetpack/guide.

[2] José Fernando Oliveira FEUP, *Filas de espera*, 1998.

[3] (2020). Firebase cloud messaging @ONLINE, URL: https://firebase.google.com/docs/cloud-messaging.

[4] (2020). Flutter @ONLINE, URL: https://flutter.dev.

[5] Rob J Hyndman & George Athanasopoulos. (2016). Forecasting: Principles and practice @ONLINE, URL: https://otexts.com/fpp2/.

[6] (2021). Heroku @ONLINE, URL: https://www.heroku.com.

[7] (2014). Data access with jdbc @ONLINE, URL: https://docs.spring.io/spring-framework/docs/4.0.x/spring-framework-reference/html/jdbc.html.

[8] C. E. M. Pearce, "Moving average processes in queueing theory", Tese de Doutoramento, The Australian National University Canberra, 1965.

[9] (2006). Model view controller @ONLINE, URL: https://martinfowler.com/eaaDev/uiArchs.html#ModelViewController.

[10] (Jul. de 2017). The model-view-viewmodel pattern @ONLINE, URL: https://docs.microsoft.com/en-us/xamarin/xamarin-forms/enterprise-application-patterns/mvvm.

[11] (2020). Node.js @ONLINE, URL: https://nodejs.org/en/.

[12] (2017). Observer design pattern @ONLINE, URL: https://docs.microsoft.com/en-us/dotnet/standard/events/observer-design-pattern.

[13] Frederick S. Hillier / Gerald J. Lieberman, *Introduction To Operations Research*. McGraw-Hill, 2001.

[14] Robert Kissell e Jim Poserina. (2017). Poisson distribution in optimal sports math, statistics, and fantasy @ONLINE, URL: https://www.sciencedirect.com/topics/mathematics/poisson-distribution.

[15] (2018). Publisher-subscriber pattern @ONLINE, URL: https://docs.microsoft.com/en-us/azure/architecture/patterns/publisher-subscriber.

[16] (2020). Proteção de dados pessoais @ONLINE, URL: https://eur-lex.europa.eu/legal-content/PT/TXT/HTML/?uri=LEGISSUM:310401_2.

[17] (2020). Siren: A hypermedia specification for representing entities @ONLINE, URL: https://github.com/kevinswiber/siren.

[18] (2020). Spring boot @ONLINE, URL: https://spring.io/projects/spring-boot.

[19] (2020). Spring mvc @ONLINE, URL: https://docs.spring.io/spring-framework/docs/3.2.x/spring-framework-reference/html/mvc.html.

[20] Yun Yang. (2017). Temporal data mining @ONLINE, URL: https://www.sciencedirect.com/topics/computer-science/temporal-data-mining.

[21] (Mar. de 2018). Teoria das filas @ONLINE, URL: https://pt.wikipedia.org/wiki/Teoria_das_filas.

[22] Jason Brownlee. (ago. de 2019). What is time series forecasting? @ONLINE, URL: https://machinelearningmastery.com/time-series-forecasting/.

[23] (Nov. de 2010). Waiting line management: Costs function and service level @ONLINE, URL: https://www.shmula.com/waiting-line-management/7217/.