Universidade Federal
do Rio de Janeiro
Escola Politécnica

SISTEMA DE CONTROLE DE ACESSO E AGENDAMENTO PARA O RESTAURANTE UNIVERSITÁRIO

Manoel Fernando de Sousa Domingues Júnior

Projeto de Graduação apresentado ao Curso de Engenharia Eletrônica e de Computação da Escola Politécnica, Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários para obtenção do título de Engenheiro.

Orientadores: Fernando Luiz Bastos Ribeiro e Flávio Luis de Mello

Rio de Janeiro
Fevereiro de 2019

SISTEMA DE CONTROLE DE ACESSO E AGENDAMENTO PARA O RESTAURANTE UNIVERSITÁRIO

Manoel Fernando de Sousa Domingues Júnior

PROJETO DE GRADUAÇÃO SUBMETIDO AO CORPO DOCENTE DO CURSO DE ENGENHARIA ELETRÔNICA E DE COMPUTAÇÃO DA ESCOLA POLITÉCNICA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE ENGENHEIRO ELETRÔNICO E DE COMPUTAÇÃO.

Autor:

Manoel Fernando de Sousa Domingues Júnior

Orientador:

Prof. Fernando Luiz Bastos Ribeiro, D. Sc.

Orientador:

Prof. Flávio Luis de Mello, D. Sc.

Examinador:

Prof. Heraldo Luís Silveira de Almeida, D. Sc.

Examinador:

Prof. Celso Alexandre Souza de Alvear, D. Sc.

Rio de Janeiro

Fevereiro de 2019

Declaração de Autoria e de Direitos

Eu, *Manoel Fernando de Sousa Domingues Júnior* CPF *116.260.437-95*, autor da monografia *SISTEMA DE CONTROLE DE ACESSO E AGENDAMENTO PARA O RESTAURANTE UNIVERSITÁRIO*, subscrevo para os devidos fins, as seguintes informações:

1. O autor declara que o trabalho apresentado na disciplina de Projeto de Graduação da Escola Politécnica da UFRJ é de sua autoria, sendo original em forma e conteúdo.

2. Excetuam-se do item 1. eventuais transcrições de texto, figuras, tabelas, conceitos e ideias, que identifiquem claramente a fonte original, explicitando as autorizações obtidas dos respectivos proprietários, quando necessárias.

3. O autor permite que a UFRJ, por um prazo indeterminado, efetue em qualquer mídia de divulgação, a publicação do trabalho acadêmico em sua totalidade, ou em parte. Essa autorização não envolve ônus de qualquer natureza à UFRJ, ou aos seus representantes.

4. O autor pode, excepcionalmente, encaminhar à Comissão de Projeto de Graduação, a não divulgação do material, por um prazo máximo de 01 (um) ano, improrrogável, a contar da data de defesa, desde que o pedido seja justificado, e solicitado antecipadamente, por escrito, à Congregação da Escola Politécnica.

5. O autor declara, ainda, ter a capacidade jurídica para a prática do presente ato, assim como ter conhecimento do teor da presente Declaração estando ciente das sanções e punições legais, no que tange a cópia parcial, ou total, de obra intelectual, o que se configura como violação do direito autoral previsto no Código Penal Brasileiro no art.184 e art.299, bem como na Lei 9.610.

6. O autor é o único responsável pelo conteúdo apresentado nos trabalhos acadêmicos publicados, não cabendo à UFRJ, aos seus representantes, ou ao(s) orientador(es), qualquer responsabilização/ indenização nesse sentido.

7. Por ser verdade, firmo a presente declaração.

Manoel Domingues Junior

Manoel Fernando de Sousa Domingues Júnior

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
Escola Politécnica - Departamento de Eletrônica e de Computação
Centro de Tecnologia, bloco H, sala H-217, Cidade Universitária
Rio de Janeiro - RJ CEP 21949-900

DEDICATÓRIA

Dedico este trabalho ao povo brasileiro que contribuiu de forma significativa à minha formação e estada nesta Universidade. Este projeto é uma pequena forma de retribuir o investimento e confiança em mim depositados.

AGRADECIMENTO

Inicialmente agradeço aos meus pais e familiares que me proporcionaram meios de conseguir uma educação de qualidade.

Agradeço de forma extremamente profunda aos professores do corpo docente do Departamento de Engenharia Eletrônica e de Computação por terem dedicado um pouco do seu tempo à minha orientação durante o curso. Cada conversa, cada dúvida respondida e cada palavra de motivação me faz capaz de acreditar que podemos mudar a realidade de nossa sociedade. Em especial, não posso deixar de agradecer ao Prof. Jomar Gozzi, Prof. Luiz Wagner Pereira Biscainho e Prof. Carlos José Ribas d'Ávila.

Agradeço também à Decania do Centro de Tecnologia que me deu recursos e autonomia para a realização do projeto tema deste trabalho. Em especial, não posso deixar de agradecer ao Prof. Fernando Luiz Bastos Ribeiro, ao Chefe da Seção de Informática, Diego Chiappetta e a ex-Superintendente Wilma Almeida. Sem o apoio e competência dessa equipe certamente esse trabalho não seria colocado em prática.

Por fim, agradeço ao corpo administrativo do Sistema de Alimentação da UFRJ que permitiu a utilização do sistema desenvolvido em suas unidades. Em especial, agradeço a colaboração da nutricionista Renata Machado e da Prof. Lúcia Pereira de Andrade.

RESUMO

O presente trabalho trata do desenvolvimento de um conjunto de softwares com a finalidade de reduzir as filas no acesso ao Restaurante Universitário da Universidade Federal do Rio de Janeiro, de forma a garantir que os usuários do Restaurante possam planejar suas atividades acadêmicas de forma mais eficiente. Este trabalho também pretende demonstrar que o uso de soluções tecnológicas pode aumentar a eficiência de processos de atendimento já consolidados, além de diminuir a incidência de fraudes sem colocar os operadores do sistema em risco. O trabalho aborda os detalhes de arquitetura e implementação assim como os objetivos intrínsecos do sistema, que envolvem desde o fornecimento de uma solução com baixo custo de manutenção como requisitos robustos de segurança de sistemas. Outros detalhes sobre os desafios enfrentados, o impacto com a utilização do sistema e possibilidades futuras também são discutidos no presente trabalho.

Palavras-Chave: cliente-servidor, sistemas-distribuídos, API HTTP, autenticação mobile.

ABSTRACT

The present work deals with the development of a set of softwares with the purpose of reducing the access queues to the University Restaurant of the Federal University of Rio de Janeiro, in order to guarantee that the users of the Restaurant can plan their academic activities in a more efficient way. This paper also aims to demonstrate that the use of technological solutions can increase the efficiency of already consolidated processes, as well as reduce the incidence of fraud without putting system operators at risk. The work addresses the details of system architecture and implementation and the system's intrinsic goals, ranging from providing a low-cost maintenance solution to robust systems security requirements. Further details on the challenges faced, the impact with the use of the system and future possibilities are also discussed in this paper.

Key-words: client-server, distributed-systems, HTTP API, mobile authentication.

SIGLAS

UFRJ - Universidade Federal do Rio de Janeiro

CCMN - Centro de Ciências Matemáticas e da Natureza

CT - Centro de Tecnologia

API - *Application Programming Interface*

CPF - Cadastro de Pessoas Físicas

CAPTCHA - *Completely Automated Public Turing test to tell Computers and Humans Apart*

JSON - *JavaScript Object Notation*

JWT - *JSON Web Tokens*

SIGA - Sistema Integrado de Gestão Acadêmica

SMS - *Short Message Service*

DNS - *Domain Name Service*

VRRP - *Virtual Router Redundancy Protocol*

HTTP - *Hypertext Transfer Protocol*

HTML - *Hypertext Markup Language*

IP - *Internet Protocol*

ACID - Atomicidade, Consistência, Isolamento e Durabilidade

TOTP - *Time-based One Time Password*

RGB - *Red, Green e Blue*

UML - *Unified Modeling Language*

Sumário

1 Introdução **1**

1.1 Tema . . . 1

1.2 Delimitação . . . 1

1.3 Justificativa . . . 1

1.4 Objetivos . . . 2

1.5 Metodologia . . . 3

1.6 Organização do trabalho . . . 3

2 O Sistema de Alimentação da UFRJ **5**

2.1 História . . . 5

2.2 Funcionamento dos restaurantes . . . 6

2.3 Problemas de atendimento . . . 6

2.4 Conclusão . . . 7

3 Engenharia do Sistema **8**

3.1 Introdução . . . 8

3.2 Concepção do sistema . . . 8

3.3 Conceitos utilizados no sistema . . . 9

3.4 Diagramas de casos de uso . . . 10

3.4.1 Interface-cliente . . . 11

3.4.2 Interface-administrativa . . . 16

3.5 Conclusão . . . 37

4 Implementação do Sistema **38**

4.1 Suporte ao sistema . . . 41

4.2 Relação do sistema com a plataforma como serviço . . . 42

4.3 Comunicação entre componentes do sistema 43
4.4 Linguagens de programação utilizadas 47
4.5 Disponibilização de interfaces para o usuário e cliente 48
4.6 Interface da API 48
4.6.1 Autenticação 49
4.6.2 Permissionamento 50
4.6.3 Auditoria 51
4.6.4 Condições de corrida: agendamento 53
4.6.5 Logout de operadores 55
4.6.6 Comunicação com outros recursos: banco de dados e CAPTCHA 56
4.6.7 Recursos disponibilizados 57
4.7 Interface do Cliente 59
4.7.1 Navegação inicial no sistema 59
4.7.2 Visualização de atendimentos disponíveis 61
4.7.3 Agendamento de horário para refeição 63
4.7.4 Busca de agendamento 66
4.7.5 Cancelamento de agendamento 68
4.7.6 Uso do telão 69
4.8 Interface de Administração 69
4.8.1 Autenticação no sistema 71
4.8.2 Listagem de atendimentos 72
4.8.3 Criação de atendimentos 73
4.8.4 Edição de atendimentos 75
4.8.5 Atendimento de agendamentos 76
4.8.6 Cadastro de clientes e operadores 78
4.9 Documentação do sistema 79

5 Impactos, conclusões e trabalhos futuros **83**
5.1 Impacto na utilização do sistema 83
5.2 Conclusão e trabalhos futuros 83

Bibliografia **86**

Lista de Figuras

3.1 Diagrama com casos de uso do cliente do sistema. 12
3.2 Diagrama com casos de uso do operador do sistema: Autenticação e atendimentos. 17
3.3 Diagrama com casos de uso do operador do sistema: agendamentos. 18
3.4 Diagrama com casos de uso do operador do sistema: relatórios, clientes e operadores. 19

4.1 Diagrama de arquitetura do sistema. 40
4.2 Diagrama de sequência para comunicação entre partes do sistema. 45
4.3 Árvore de permissões hierárquicas dos atendimentos. 51
4.4 Árvore de permissões hierárquicas dos clientes. 51
4.5 Árvore de permissões hierárquicas dos operadores. 52
4.6 Página inicial da interface-cliente do sistema. 60
4.7 Página inicial da interface-cliente do sistema. 60
4.8 Página inicial da interface-cliente do sistema com 3 botões (desativada). . 61
4.9 Página da interface-cliente do sistema com listagem de atendimentos disponíveis. 62
4.10 Página da interface-cliente do sistema com formulário para agendamento. . 63
4.11 Página da interface-cliente do sistema com resposta para agendamento no horário solicitado pelo cliente. 64
4.12 Página da interface-cliente do sistema com resposta para agendamento em horário diferente do solicitado pelo cliente. 64
4.13 Página da interface-cliente do sistema com resposta para agendamento com QR Code. 65
4.14 Página da interface-cliente do sistema para clientes com pendências cadastrais. 66

4.15 Página da interface-cliente do sistema com formulário para busca de agendamento. 67
4.16 Página da interface-cliente do sistema com resultado da busca realizada no mesmo dispositivo do agendamento. 67
4.17 Página da interface-cliente do sistema com teste CAPTCHA para cancelamento de agendamento. 68
4.18 Página da interface-cliente do sistema com resposta ao pedido de cancelamento de agendamento. 69
4.19 Página da interface-cliente do sistema preparada para uso nos telões informativos. 70
4.20 Página da interface-administrativa do sistema com formulário para autenticação. 71
4.21 Página da interface-administrativa do sistema para autenticação multifator e tela de um dispositivo móvel para confirmação de autenticação, ambos com Duo Mobile. 72
4.22 Página da interface-administrativa do sistema após autenticação realizada com sucesso. 73
4.23 Página da interface-administrativa do sistema com listagem de atendimentos. 74
4.24 Página da interface-administrativa do sistema com listagem de atendimentos fechados. 74
4.25 Página da interface-administrativa do sistema com formulário para criação de atendimento. 75
4.26 Página da interface-administrativa do sistema com edição de atendimento. 76
4.27 Página da interface-administrativa do sistema listando agendamentos de determinado atendimento. 77
4.28 Página da interface-administrativa do sistema listando operadores cadastrados. 79
4.29 Página da interface-administrativa do sistema listando clientes cadastrados. 80
4.30 Página da interface-administrativa do sistema exibindo erro de permissão. 80
4.31 Página inicial da documentação de interoperabilidade do sistema. 81
4.32 Página da documentação de interoperabilidade mostrando as rotas HTTP disponíveis no sistema. 82

4.33 Página da documentação de interoperabilidade referente a rota POST /queues/:uid. 82

5.1 Foto da fila do restaurante antes implantação do sistema de agendamento. 84

5.2 Foto da fila do restaurante após implantação do sistema de agendamento. . 84

Capítulo 1

Introdução

1.1 Tema

O tema do trabalho é o estudo da arquitetura e implementação do sistema de controle de acesso e agendamento utilizado no Restaurante Universitário da UFRJ. Dessa forma, o problema consiste em projetar um sistema com a finalidade de diminuir a formação de filas durante o atendimento no restaurante, assim como proporcionar aos usuários do restaurante uma forma de fazer seu planejamento diário com base na capacidade de atendimento do restaurante.

1.2 Delimitação

O objeto de estudo é o sistema de controle de acesso e agendamento utilizado no Restaurante Universitário do Centro de Tecnologia, seus protótipos e sua versão em funcionamento em 2017 e 2018.

1.3 Justificativa

O objetivo do Restaurante Universitário da UFRJ é "oferecer alimentação de qualidade, equilibrada, e acessível de forma a favorecer a permanência dos estudantes no espaço universitário, permitindo-lhes dedicação integral aos estudos, sendo importante meio de combate à evasão escolar" [1]. Em conjunto com esse objetivo, o Restaurante procurou nos últimos anos aumentar sua capacidade de produção de

alimentos, superando a meta do Plano de Desenvolvimento Institucional 2020, e em 2018 sendo capaz de produzir cerca de 10.000 refeições diárias [2].

Contudo, somente aumentar a capacidade de produzir refeições para o Restaurante não é suficiente para o seu funcionamento eficiente. A forma de atendimento utilizada em conjunto com a ausência de um efetivo controle de acesso, fez com que longas filas e tempos de espera superiores à 1 hora ocorressem na unidade do Centro de Tecnologia. Embora uma das questões centrais relaciona-se com o fato da demanda ser muito superior à oferta na unidade do CT, o esgotamento das refeições não era um item controlado e era passível de ocorrência em todos os dias de funcionamento da unidade. Esse cenário dificulta a dedicação integral aos estudos, tornando habitual o abandono precoce de atividades e atrasos em aulas para assegurar a permanência nas filas de atendimento.

Por isso, o presente trabalho aborda a arquitetura e implementação de um sistema que permite o agendamento do horário pretendido para as refeições, que auxilie no planejamento de capacidade do restaurante e no controle de acesso ao mesmo. O sistema atua de modo que o usuário do restaurante possa fazer o uso de acordo com sua disponibilidade, sem gerar impactos inesperados nas atividades acadêmicas.

O trabalho não pretende solucionar o problema central da demanda ser muito superior à capacidade de produção e distribuição de refeições. A proposta é que o trabalho demonstre como um sistema de controle de acesso e agendamento pode ter impactos positivos nesse cenário de limitações.

1.4 Objetivos

O objetivo é mostrar a arquitetura e os métodos utilizados durante a implementação do sistema de controle de acesso no Restaurante Universitário.

Para isso, serão abordadas: (1) a especificação dos principais cenários de uso do sistema proposto e (2) a plataforma de computação em nuvem utilizada. Em seguida, o presente trabalho tratará: (3) os requisitos de segurança do sistema e (4) a arquitetura utilizada nos principais cenários de uso. Finalmente, serão abordados: (5) a elaboração da documentação de cada parte do sistema e (6) os impactos do

sistema na comunidade acadêmica.

1.5 Metodologia

Para atender ao objetivo, foi criada uma infraestrutura baseada em uma plataforma como serviço [3] na qual foi prototipado o sistema de agendamento. Essa infraestrutura utilizou excedentes de processamento da infraestrutura corporativa da Decania do Centro de Tecnologia na sua fase de testes e utilizou os serviços de 2 funcionários da Seção de Informática como apoio logístico e operacional, 1 aluno de graduação de Engenharia Eletrônica como arquiteto e desenvolvedor principal do sistema e outro aluno de graduação de Engenharia da Computação como desenvolvedor dos aplicativos *mobile* [4].

Durante o ano de 2016, através de reuniões de alinhamento de demandas, nutricionistas, professores e funcionários da Universidade ajudaram a construir a visão dos problemas que o sistema deveria se propor a resolver. Juntamente com a visão dos problemas, uma arquitetura de software foi planejada, documentada e prototipada para uso durante o recesso acadêmico de 2016.

Após a prototipagem do sistema e validação da proposta, uma versão final foi produzida, documentada e disponibilizada à Universidade em 2017. Ambas as versões conservam as mesmas funcionalidades, tendo sua maior diferença na documentação de interoperabilidade para integração com outros sistemas.

1.6 Organização do trabalho

No segundo capítulo, faz-se uma contextualização do Sistema de Alimentação da Universidade Federal do Rio de Janeiro, seu funcionamento e características que motivaram a criação do sistema objeto do trabalho.

No terceiro capítulo, caracterizam-se os cenários de uso com base nas reuniões com o público alvo do sistema. Para cada cenário de uso, observações relativas a usabilidade e segurança são realizadas de maneira a fornecer um panorama completo de cada funcionalidade chave do sistema.

No quarto capítulo, aborda-se uma introdução aos conceitos de plataformas como serviço e discute-se a tomada de decisões técnicas relativas ao desenvolvimento do sistema. No mesmo capítulo abordam-se a arquitetura de funcionamento do sistema e os principais tópicos envolvidos em cada uma das partes do sistema. As interfaces são detalhadas e ilustradas, incluindo as principais operações possíveis de serem realizadas no sistema.

No quinto e último capítulo, concluí-se o trabalho e discute-se possibilidades de trabalhos futuros.

Capítulo 2

O Sistema de Alimentação da UFRJ

2.1 História

O Sistema de Alimentação da UFRJ surgiu através do Plano de Desenvolvimento Institucional 2020 [5] e de um esforço de reinaugurar os restaurantes anteriormente fechados na década de 90 [6].

Durante o movimento da década de 90 intitulado "Roleta da fome", os funcionários da universidade protestavam devido ao aumento de preços e logo após o fechamento de algumas unidades, passavam a protestar também sobre as "filas intermináveis e uma espera de horas"no restaurante do CCMN. Logo após esse movimento, os restaurantes foram fechados em toda Universidade.

Com a reabertura da unidade satélite na Faculdade de Letras em 2008 [7], a unidade central no mesmo ano [8] e a abertura da unidade satélite do Centro de Tecnologia em 2012 [9] chegamos à distribuição de refeições contextualizada no âmbito inicial deste trabalho.

Em 2016, houve o início da distribuição de refeições nos Pólos do Centro e da Praia Vermelha [10] e a inauguração do restaurante no Pólo de Xerém em 2018 [11].

2.2 Funcionamento dos restaurantes

Desde a concepção inicial, os restaurantes possuem uma unidade Central responsável pela produção das refeições para atender toda a Universidade. As outras unidades, conhecidas como unidades-satélite, ficam distribuídas pelos campus e recebem as refeições produzidas pela unidade Central. Nesse contexto, as unidades-satélite apenas distribuem as refeições.

Inicialmente, as refeições eram produzidas por uma empresa terceirizada e transportadas até a Universidade onde era realizada a distribuição. Após a conclusão das obras da cozinha industrial no Restaurante Central, as refeições passaram a ser produzidas dentro do campus da Ilha do Fundão e distribuídas para as demais unidades.

O restaurante central, com capacidade de atender 4000 refeições por dia, segundo o PDI 2020 [5], é uma unidade separada, no qual o prédio é destinado inteiramente para o serviço de refeições. Esse cenário difere do ambiente das unidades satélite, que estão dentro do espaço físico compartilhado por salas de aula, laboratórios e escritórios da Universidade.

Nesse cenário de limitações físicas devido ao compartilhamento de local, as unidades satélites possuem uma capacidade de atendimento menor. Sendo que, no Centro de Tecnologia, escopo do presente trabalho, o número de refeições diárias varia entre 1300 e 1500 refeições por dia [12] frente ao corpo social de aproximadamente de 12000 pessoas [13].

2.3 Problemas de atendimento

Desde a inauguração das unidades-satélite, o acesso aos restaurantes foi um ponto criticado devido as longas filas de espera. Segundo Patrícia Pereira, que realizou um trabalho de pesquisa cênica na UFRJ [14, p. 329], protestos com faixas faziam parte do cotidiano da unidade satélite da Faculdade de Letras.

Esse cenário fez com que os movimentos de permanência estudantil se articulassem e produzissem manifestos, como o do Movimento JUNTOS-UFRJ de 2013 [15] que expõe a situação no Campus da Ilha do Fundão: “na Ilha do Fundão, é nítido que as longas filas dos três restaurantes não atendem à demanda. Os bolsis-

tas muitas vezes ficam sem almoçar, já que a comida sempre acaba antes do horário previsto".

A situação das filas se tornou tão cotidiana que existe uma folclorização da mesma, como se fosse uma situação inevitável sua formação. Em outubro de 2017, o Sistema de Alimentação através de seu canal no Youtube "Você no R.U."publicou um vídeo [16] destacando o tamanho da fila de atendimento na unidade central[1].

2.4 Conclusão

A presença de filas e o término das refeições disponíveis durante o horário de atendimento revelam problemas de planejamento de capacidade no Sistema de Alimentação e em suas unidades-satélite que poderiam ser resolvidos com tecnologia.

Dessa forma, a Decania do Centro de Tecnologia, motivada por uma melhor utilização do espaço, em conjunto com o autor começaram a desenvolver em 2016 uma solução tecnológica a ser aplicada na unidade satélite do CT, mas que pudesse ser utilizada em toda Universidade.

[1]No momento da confecção do vídeo, a unidade central não utilizava o sistema.

Capítulo 3

Engenharia do Sistema

3.1 Introdução

Segundo Pfleeger [17, p. 116], a utilização de casos de usos é uma maneira conveniente de mapear os requisitos funcionais de um sistema. Essa forma é especialmente interessante uma vez que o sistema apresentaria mais de uma maneira de funcionamento.

Como houve o envolvimento da equipe de nutrição do Sistema de Alimentação da UFRJ, essa abordagem permite uma comunicação eficiente entre todos os envolvidos [17, p. 216], sejam os nutricionistas seja o corpo técnico da Decania do CT.

3.2 Concepção do sistema

A concepção do sistema foi realizada tendo algumas premissas como base para seu desenvolvimento. Essas premissas foram levantadas pelo corpo técnico da Decania do CT em conjunto com as preocupações levantadas pela equipe do Sistema de Alimentação durante os encontros no ano de 2016.

Controle de acesso: o sistema deve garantir que somente o pessoal autorizado pelo Sistema de Alimentação seja permitido utilizar as unidades em suas refeições.

Capacidade de atendimento: o sistema deve garantir que a capacidade de atendimento da unidade não seja excedida.

Mínimo impacto às atividades acadêmicas: o uso do sistema deve gerar o menor impacto possível nas atividades acadêmicas, de modo que o cenário onde os alunos/funcionários deixam suas atividades para ficar na fila de espera seja evitado.

Facilidade de uso: o sistema deve ter sua funcionalidade principal orientada à atividade principal, de forma que seja de fácil utilização e não houvesse a necessidade de treinamento.

Adicionalmente, o corpo técnico da Decania do CT, adicionou outras preocupações, como:

Privacidade: o sistema deve ter formas de garantir que a privacidade dos clientes que o utilizassem em cada unidade seja resguardada.

Auditoria: o sistema deve possuir uma maneira de auditar as operações realizadas no mesmo.

Autenticação segura: o sistema deve ter mecanismos para conter o vazamento de credenciais, evitando o comprometimento do sistema dessa forma.

Ainda, baseado no conceito de micro-serviços [18] o sistema foi dividido em três serviços, chamados para o propósito desse trabalho de interface-cliente, interface-administrativa e API.

3.3 Conceitos utilizados no sistema

De maneira a normalizar o significado de cada palavra utilizada na descrição dos casos de uso, elucida-se a seguir uma lista com os principais conceitos do sistema, seu significado e exemplos de forma a clarificar o entendimento do leitor.

Atendimento: refere-se à um ciclo de funcionamento de cada unidade, caracterizado por um intervalo de tempo (por exemplo, das 10:30 até às 14:00) e outras características como capacidade de atendimento (por exemplo, 1000

refeições)[1]. Por exemplo, um almoço na unidade do CT é um atendimento diferente de um almoço na unidade de Letras, assim como um almoço na unidade do CT é um atendimento diferente do jantar na mesma unidade. Esse conceito, delimitado com as presentes restrições por unidade e tipo de refeição, possibilita que o sistema trate de forma independente as diferentes características de capacidade, horário de atendimento e atrasos respeitando as limitações da localidade onde cada unidade está presente.

Cliente: refere-se aos utilizadores do sistema que tem por objetivo conseguir um horário para uma refeição. Alunos, professores e técnicos-administrativos são exemplos de clientes do sistema pois compõem o corpo social da Universidade e, portanto, podem utilizar os Restaurantes Universitários.

Agendamento: refere-se à um determinado horário (com precisão de minuto) reservado dentro do espaço de tempo do atendimento para um cliente, ou seja, um agendamento é uma vaga associada à um horário. Por exemplo, quando um cliente utiliza o sistema para agendar o horário das 12:00 para seu almoço, ele realiza um agendamento que garante um vaga para o mesmo às 12:00 caso esse horário esteja disponível de acordo com as limitações de atendimento.

Operador: refere-se aos utilizados do sistema com privilégios administrativos, tais como a capacidade de receber pagamentos, verificar um agendamento ou cadastrar um atendimento. Um exemplo de operadores do sistema são os técnicos terceirizados pela Universidade para atuar no atendimento dos clientes nas unidades do Sistema de Alimentação.

3.4 Diagramas de casos de uso

Os diagramas de caso de uso foram montados de acordo com o objetivo de cada interface. Os objetivos são listados a seguir.

Interface-cliente: interface utilizada pelos alunos e servidores que constituem a

[1] Mais detalhes sobre as informações que caracterizam um atendimento podem ser encontradas no item 3.4.2.4 (p. 23).

comunidade acadêmica da UFRJ para realizar o agendamento de horário da refeição.

Interface-administrativa: interface utilizada pelos operadores do Sistema de Alimentação para verificação do horário agendado e outras tarefas administrativas.

API: serviço utilizado por ambas as interfaces para implementar a lógica de negócio do sistema como um todo. Uma vez que a mesma está presente em todos os casos de uso, a mesma não será representada nos diagramas.

3.4.1 Interface-cliente

A interface-cliente é o serviço acessado pela comunidade acadêmica da Universidade e seu papel principal é servir de porta de entrada para a maior parte dos utilizadores. Como seu acesso é público e irrestrito, todos os casos de uso incluem um CAPTCHA de forma a evitar que a mesma seja usada de maneira automatizada.

Na figura 3.1 temos um mapa dos casos de uso tratados na interface-cliente. Eles serão detalhados a seguir com as pré-condições, fluxos e pós-condições. Cabe destacar que embora o SIGA[2] apareça como um ator nessa ilustração, o mesmo é apenas consultado pela API, de modo que todas as ações feitas na interface são protagonizadas pelo ator "Cliente"e executadas pelo ator "API".

3.4.1.1 Verificar atendimentos disponíveis

O cenário de uso em questão surge da necessidade de tornar o sistema compatível com o uso por todas as unidades dentro da Universidade. Caso não fosse possível obter todos os atendimentos disponíveis, os clientes do sistema não teriam a possibilidade de verificar quais unidades já iniciaram o atendimento e escolher quais delas melhor lhe atende.

Pré-Condições: O cliente acessa a interface-cliente a partir da internet ou da rede interna da UFRJ.

[2]O SIGA é o sistema responsável por manter e prover os dados de toda comunidade acadêmica da Universidade.

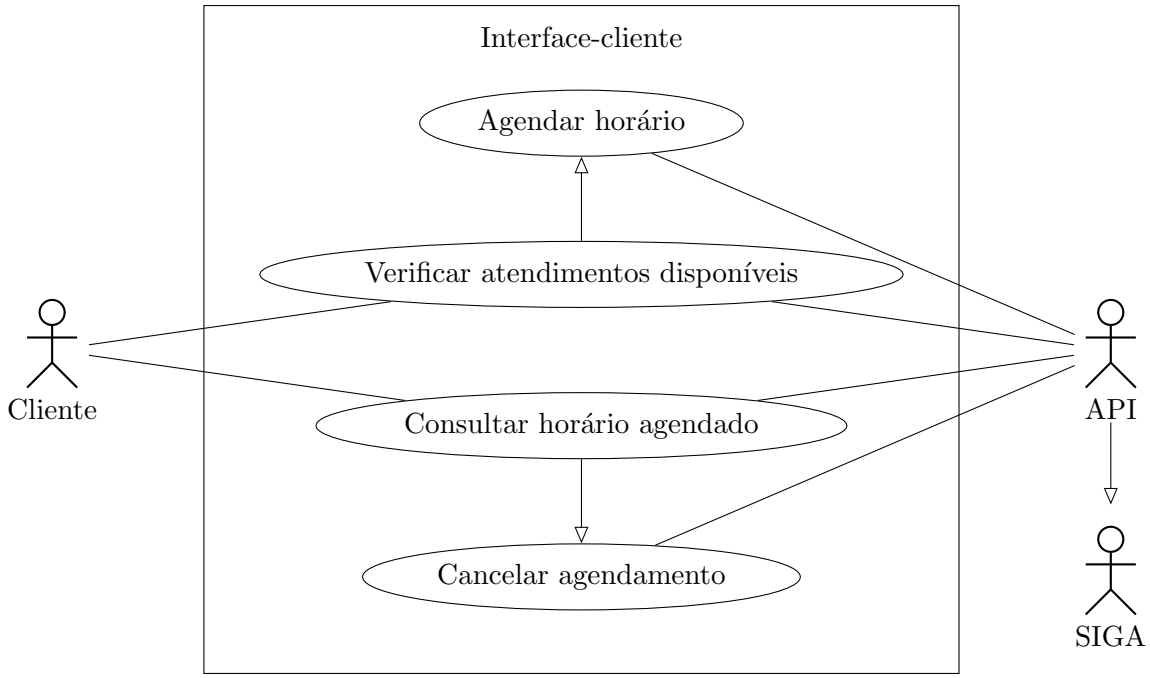

Figura 3.1: Diagrama com casos de uso do cliente do sistema.

Fluxo Principal: 1. O cliente acessa a interface-cliente

2. O cliente seleciona a opção correspondente para verificar os atendimentos disponíveis.
3. A interface-cliente solicita à API os atendimentos disponíveis, caso algum exista.
4. A interface-cliente mostra os dados retornados pela API ao cliente.

Fluxos Alternativos: Não existem.

Fluxo de Exceção: Caso nenhum agendamento exista (após passo 3)

- A interface-cliente não exibe nenhum atendimento ao cliente.

Caso a API esteja indisponível (após passo 3)

- A interface-cliente exibe uma mensagem ao cliente e solicita ao mesmo que refaça a solicitação em alguns instantes.

Pós-Condições: Não existem.

3.4.1.2 Agendar horário

O cenário de uso em questão é o mais utilizado pelos clientes. Ele ocorre como dependência do cenário de uso anterior (Verificar atendimentos disponíveis) apenas como forma de garantir que os clientes escolham de forma adequada o atendimento a qual desejam agendar.

Nesse caso de uso, existe um novo ator que é o Sistema Integrado de Gestão Acadêmica - SIGA, responsável por verificar se um CPF faz efetivamente parte da comunidade acadêmica. Embora essa verificação não seja realizada pela interface-cliente, mas pela API, ilustramos o SIGA nesse caso de uso pois é um recurso utilizado apenas nesse e no caso de uso 3.4.2.9 (p. 29).

Pré-Condições: O cliente acessa a interface-cliente a partir da internet ou da rede interna da UFRJ. Adicionalmente é necessário que algum atendimento esteja disponível.

Fluxo Principal:

1. O cliente acessa a interface-cliente
2. O cliente seleciona a opção correspondente para verificar os atendimentos disponíveis.
3. A interface-cliente solicita à API os atendimentos disponíveis.
4. A interface-cliente mostra os atendimentos retornados pela API ao cliente.
5. O cliente seleciona a opção correspondente para agendar um horário.
6. A interface-cliente mostra um formulário solicitando o CPF, o horário pretendido para refeição e um teste de segurança do tipo CAPTCHA.
7. O cliente preenche os dados solicitados e seleciona a opção correspondente para enviar.
8. A interface-cliente solicita à API o agendamento. Nesse passo, a API verifica no SIGA o CPF do cliente, assim como demais verificações tratadas em 4.6.4 (p. 53).
9. A interface-cliente mostra os dados retornados pela API ao cliente.

Fluxos Alternativos: Caso o cliente esteja em dúvida sobre qual agendamento escolher (após passo 6)

- O cliente pode selecionar a opção correspondente para listar os atendimentos disponíveis novamente.

Caso o cliente desista do agendamento (após passo 9)

- O cliente pode selecionar a opção correspondente para cancelar seu agendamento.

Fluxo de Exceção: Caso o horário solicitado não esteja disponível (após passo 9)

- A interface-cliente retorna o agendamento no próximo horário disponível.

Caso o CAPTCHA não tenha sido preenchido corretamente (após passo 7)

- A interface-cliente mostra um erro referente ao preenchimento incorreto do CAPTCHA.

Pós-Condições: O horário solicitado pelo cliente passará a ser atribuído ao seu CPF, respeitando as limitações administrativas do sistema. Adicionalmente, o navegador do cliente passa a armazenar um identificador de cancelamento aleatório na forma de *cookie* [19] [3].

3.4.1.3 Consultar horário agendado

O presente cenário de uso se torna necessário pela possibilidade dos clientes esquecerem seu horário de agendamento e, portanto, ter uma maneira de recuperar essa informação de forma independente, sem ter que procurar a unidade para a qual ele fez o agendamento.

Pré-Condições: O cliente acessa a interface-cliente a partir da internet ou da rede interna da UFRJ. O cliente deve ter agendado previamente um horário em um atendimento disponível.

[3] Uma pós-condição desejada era o envio de notificações para os clientes próximo de seu horário agendado como forma de lembrete. Como a utilização da interface-cliente é através de navegadores web, demandaria muito esforço integrar a interface com o mecanismo utilizado por cada fabricante de navegador web. Também não é possível o envio de notificações por e-mail uma vez que o SIGA não disponibiliza essa informação. Esse ponto está mapeado como uma possibilidade de trabalho futuro no item 5.2 (p. 83).

Fluxo Principal: 1. O cliente acessa a interface-cliente

2. O cliente seleciona a opção correspondente para buscar o horário agendado.
3. A interface-cliente mostra um formulário solicitando o CPF, o atendimento para a busca e um CAPTCHA.
4. O cliente preenche os dados solicitados e seleciona a opção correspondente para enviar.
5. A interface-cliente solicita à API se existe algum horário agendado para aquele CPF no atendimento em questão.
6. A interface-cliente mostra os dados retornados pela API ao cliente.

Fluxos Alternativos: Caso o cliente esteja realizando a consulta antes do seu horário agendado, no mesmo navegador que agendou (após passo 6)

- A interface-cliente mostra uma opção para cancelar agendamento ao lado do horário agendado. Esse caso de uso será tratado a seguir.

Fluxo de Exceção: Caso o CAPTCHA não tenha sido preenchido corretamente (após passo 4)

- A interface-cliente mostra um erro referente ao preenchimento incorreto do CAPTCHA.

Pós-Condições: Não existem.

3.4.1.4 Cancelar agendamento

A finalidade do presente caso de uso é possibilitar ao cliente que o mesmo possa realizar o cancelamento de seu agendamento prévio caso o mesmo esteja impossibilitado de comparecer.

Uma vez que o cancelamento possa ser realizado, ele libera a vaga antes ocupada pelo cliente para outros possíveis clientes que desejam fazer a refeição no mesmo horário. Adicionalmente, por se tratar de um mecanismo que pode ser abusado (onde um cliente poderia ter a capacidade de cancelar o agendamento de outro), foi incluído durante o caso de uso do agendamento de horário (item 3.4.1.2,

p. 13) um identificador de cancelamento aleatório que é armazenado no navegador do cliente de forma a mitigar essa possibilidade (conforme descrito nas pós-condições do respectivo item). Uma vez que esse identificador está armazenado no navegador do cliente, ele é enviado junto com o resultado do teste do tipo CAPTCHA à API.

Pré-Condições: O cliente acessa a interface-cliente a partir da internet ou da rede interna da UFRJ. O cliente deve ter agendado previamente um horário em um atendimento disponível. O cliente deve ter realizado o caso de uso referente à consultar horário agendado.

Fluxo Principal:

1. A interface-cliente mostra uma opção para cancelar agendamento ao lado do horário agendado.
2. Caso o cliente selecione a opção referente ao cancelamento, a interface-cliente mostrará um formulário composto de apenas um CAPTCHA para preenchimento do cliente. O cliente deverá preencher o CAPTCHA e selecionar a opção correspondente à enviar.
3. A interface-cliente solicita à API o cancelamento, informando o resultado do teste do tipo CAPTCHA e o identificador de cancelamento aleatório.
4. A interface-cliente mostra os dados retornados pela API ao cliente.

Fluxos Alternativos: Não existem.

Fluxo de Exceção: Caso o cliente não possua o identificador de cancelamento referente ao horário que está tentando cancelar, nenhuma modificação é realizada no sistema.

Pós-Condições: O horário anteriormente agendado para o cliente passará a possuir o estado disponível.

3.4.2 Interface-administrativa

A interface-administrativa é o ponto de controle do sistema por parte do Sistema de Alimentação da UFRJ. Seu papel principal é oferecer agilidade no atendimento dos clientes das unidades e um conjunto de funcionalidades que possibilite uma gestão integrada das unidades.

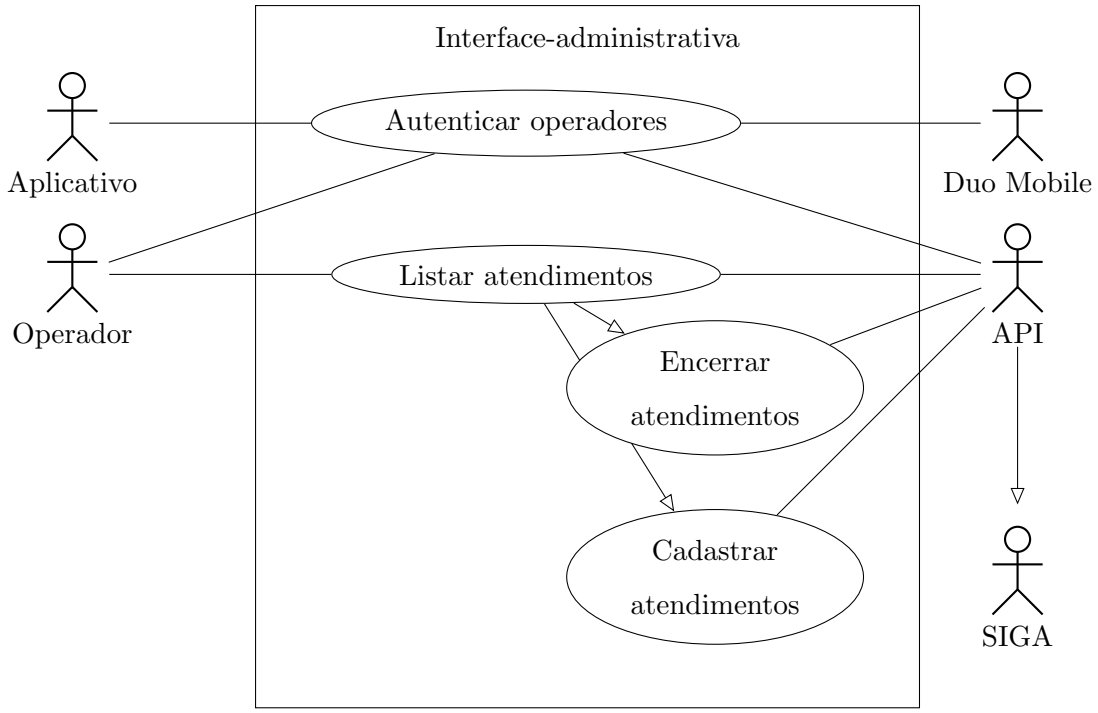

Figura 3.2: Diagrama com casos de uso do operador do sistema: Autenticação e atendimentos.

Devido ao grande número de casos de uso que a mesma atende, seus diagramas foram separados em 3 partes. Os casos de uso mais cotidianos e que fazem parte do dia-a-dia dos operadores está representado na figura 3.2. O diagrama com os casos de uso envolvendo as operações de atendimento aos agendamentos e controle de acesso está representado na figura 3.3. Por fim, as tarefas administrativas realizadas ocasionalmente pelos operadores estão registradas no diagrama da figura 3.4.

No diagrama de casos de uso representado na figura 3.2, o ator Aplicativo representa um software instalado no dispositivo móvel do ator Operador. Esse software possui a finalidade de servir como meio de confirmar a autenticação do operador utilizando o sistema Duo Mobile. Esse cenário de uso será detalhado a seguir.

3.4.2.1 Autenticar operadores

A finalidade do presente caso de uso é possibilitar aos operadores do sistema acesso seguro às funcionalidades administrativas do mesmo.

Este caso de uso caracteriza-se por ser um dos mais sensíveis uma vez que um indivíduo mal-intencionado consiga se autenticar no sistema de forma indevida,

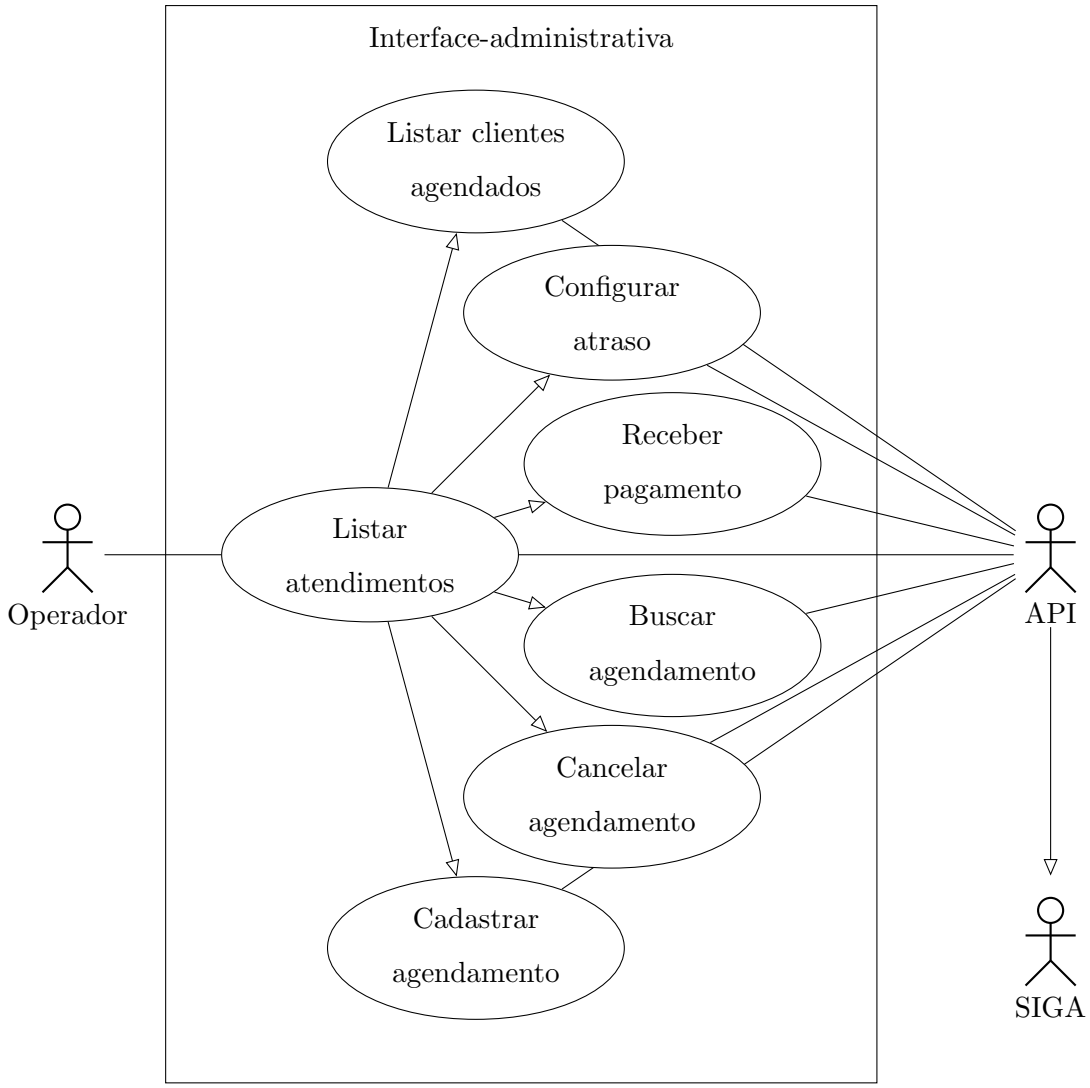

Figura 3.3: Diagrama com casos de uso do operador do sistema: agendamentos.

o mesmo pode colocar o funcionamento do sistema em risco. Para mitigar os problemas mais comuns da autenticação entre entidades [20, p. 203], foi implementado nesse caso de uso a autenticação multi-fator. Dessa forma, a interface-administrativa passa a validar através do dispositivo móvel do operador (previamente cadastrado e verificado) se o operador que está tentando se autenticar é quem diz ser.

A necessidade de um dispositivo verificado decorre da utilização da API por aplicativos em dispositivos móveis. Conforme demonstrado pelo autor em [21], APIs de acesso externo com uso através de aplicativos móveis devem se autenticar fazendo uso de um canal separado para envio de identificadores de autenticação. De forma a tornar essa arquitetura compatível com a utilização da interface-administrativa, a mesma é previamente cadastrada nos operadores como um dispositivo verificado,

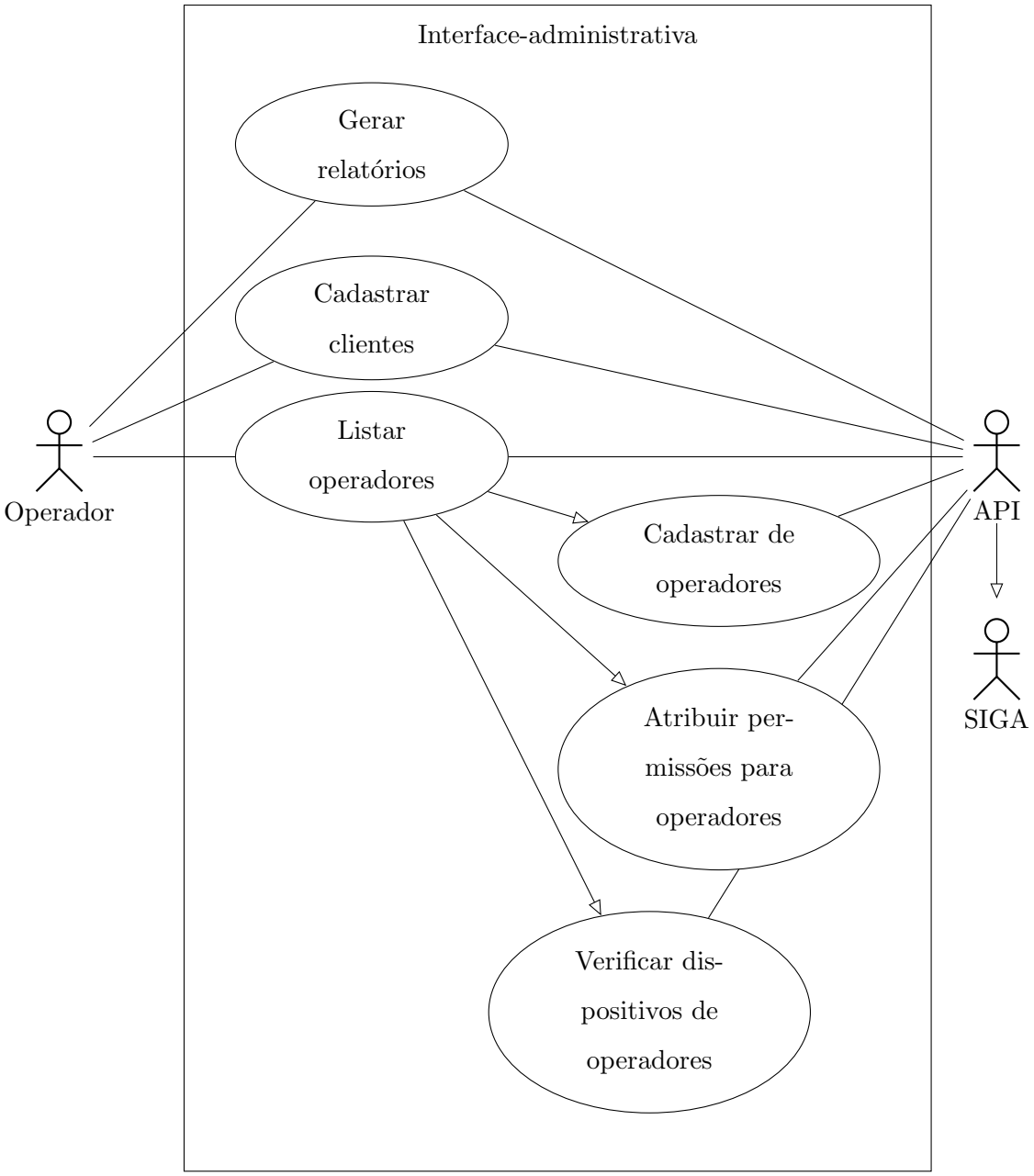

Figura 3.4: Diagrama com casos de uso do operador do sistema: relatórios, clientes e operadores.

dessa forma dispensando o uso de um canal separado para o envio do identificador de autenticação.

Podolan, em seu trabalho sobre o desenvolvimento de aplicativos [4, p. 37], explica como ocorre a comunicação no caso de um aplicativo com as funções da interface-administrativa instalado em um dispositivo móvel.

A interface-administrativa utiliza o ator externo Duo Mobile [22] como provedor da verificação de autenticação multi-fator.

Conforme pode ser observado na descrição do caso de uso a seguir, utiliza-se um JWT [23] para controlar a autenticação na API. A escolha pelo uso de um JWT como mecanismo para controlar a autenticação dos operadores se baseia na possibilidade de integração futura do sistema com provedores de autenticação com suporte ao protocolo OpenID Connect [24]. Adicionalmente, utilizar JWTs possibilita que as sessões não tenham a necessidade de ser verificadas na base de dados a cada requisição, trazendo ganhos de performance ao sistema[4].

Pré-Condições: O operador acessa a interface-administrativa a partir da internet ou da rede interna da UFRJ. O operador deve possuir credenciais previamente cadastradas no sistema, com seu dispositivo validado e o Aplicativo instalado em seu dispositivo móvel.

Fluxo Principal:

1. A interface-administrativa mostra um formulário onde o operador deve inserir suas credenciais e preencher um CAPTCHA.
2. A interface-administrativa solicita à API que seja realizada a operação de login para o operador em questão.
3. Uma vez que a API responde com sucesso às credenciais do operador, a interface-administrativa redireciona o navegador do operador para um formulário. Nesse formulário o operador deve validar seu acesso através de uma mensagem no seu dispositivo móvel.
4. O operador, de posse de seu dispositivo móvel, valida a mensagem.
5. A interface-administrativa mostra o menu com as opções disponíveis ao operador.

Fluxos Alternativos: Caso o operador não esteja de posse do seu dispositivo móvel (após passo 3)

- A interface-administrativa retorna para a tela inicial com o formulário para submissão das credenciais.

[4]Existe um conflito de escolha na utilização de JWTs como mecanismo de verificação da autenticidade de sessões, discutido no item 4.6.5, p. 55 quando é abordado em detalhes o problema de *logout* de operadores.

Caso o dispositivo móvel do operador esteja sem sinal (após passo 3)

- A interface-administrativa fornece a opção de digitação de um código sincronizado com o dispositivo móvel que não depende de sinal telefônico.

Caso o dispositivo móvel do operador não consiga abrir o aplicativo para validação ou geração do código sincronizado (após passo 3)

- A interface-administrativa fornece a opção de envio de um código via SMS para o dispositivo móvel do operador.

Fluxo de Exceção: Caso o operador apresente credenciais incorretas (após passo 2)

- A interface-administrativa mostra a mensagem de erro enviada pela API.

Caso o operador não valide a mensagem no seu dispositivo móvel em até 1 minuto (após passo 3)

- A interface-administrativa fornece a opção do operador enviar uma nova mensagem de validação.

Pós-Condições: O operador passa a ter acesso a interface-administrativa. Adicionalmente, o navegador do operador passa a armazenar um JWT de sessão na forma de *cookie*.

3.4.2.2 Listar atendimentos

Esse caso de uso tem a finalidade de oferecer um panorama geral dos atendimentos atualmente cadastrados no sistema, possibilitando uma gestão centralizada e as opções de escolha para cada operador iniciar suas atividades de atendimento aos agendamentos.

Breve Descrição: Listar atendimentos

Atores: Operadores do Sistema, API

Pré-Condições: O operador deve estar autenticado na interface-administrativa e ter permissão para listar atendimentos.

Fluxo Principal:
1. A interface-administrativa mostra as opções disponíveis para o operador.
2. O operador seleciona a opção correspondente aos atendimentos no menu.
3. A interface-administrativa solicita à API os atendimentos registrados.
4. A interface-administrativa mostra os atendimentos registrados ao operador.

Fluxos Alternativos: Não existem.

Fluxo de Exceção: Caso o operador não possua permissão para listar os atendimentos (após passo 3)

- A interface-administrativa API retorna para a interface-administrativa um erro de permissão.
- A interface-administrativa mostra a mensagem de erro enviada pela API.

Pós-Condições: É registrado para auditoria que o operador listou os atendimentos.

3.4.2.3 Encerrar atendimento

Uma vez que o atendimento é finalizado, ou seja, todos os agendamentos foram atendidos ou que o horário máximo de funcionamento tenha sido atingido, é necessário registrar no sistema que o atendimento correspondente está encerrado.

Pré-Condições: O operador deve estar autenticado na interface-administrativa e ter permissão para encerrar atendimentos.

Fluxo Principal:
1. A interface-administrativa mostra as opções disponíveis para o operador.
2. O operador seleciona a opção correspondente a encerrar o atendimento que deseja.
3. A interface-administrativa solicita à API o encerramento do atendimento.
4. A interface-administrativa lista os atendimentos registrados.

Fluxos Alternativos: Não existem.

Fluxo de Exceção: Caso o operador não possua permissão para encerrar um atendimento (após passo 3)

- A interface-administrativa API retorna para a interface-administrativa um erro de permissão.
- A interface-administrativa mostra a mensagem de erro enviada pela API e nenhuma modificação é realizada.

Pós-Condições: O atendimento encerrado tem seu estado modificado para a opção correspondente a encerrado e não é mais mostrado ao executar o caso de uso listar agendamentos. Adicionalmente, é registrado para auditoria que o operador encerrou o atendimento em questão.

3.4.2.4 Cadastrar atendimentos

O cadastro de atendimentos possibilita que seja possível a realização do controle de acesso e do agendamento nas unidades do Sistema de Alimentação. A funcionalidade de agendamento está diretamente relacionada a opção "Capacidade de atendimento da unidade por minuto"que caso esteja com valor 0, ativa somente o controle de acesso, impossibilitando o agendamento por parte dos clientes.

A funcionalidade de controle de acesso funciona de forma que a unidade não disponibiliza, via interface-cliente, a possibilidade de agendamentos por parte de seus clientes. Ou seja, o atendimento será controlado através dos operadores na unidade, onde eles utilizarão o sistema para verificar se os clientes fazem parte da comunidade acadêmica da Universidade. Essa verificação é realizada cadastrando-se agendamentos conforme explica o caso de uso 3.4.2.9. Adicionalmente, o sistema fornecerá a capacidade de emitir relatórios para a unidade (caso de uso 3.4.2.11).

Pré-Condições: O operador deve estar autenticado na interface-administrativa e ter permissão para cadastrar atendimentos.

Fluxo Principal:

1. A interface-administrativa mostra as opções disponíveis para o operador.
2. O operador seleciona a opção correspondente a cadastrar um novo atendimento.

3. A interface-administrativa mostra um formulário para o operador com os campos Descrição, Quantidade prevista de refeições, Capacidade de atendimento da unidade por minuto, Horário abertura do sistema de agendamento, Horário de início de funcionamento da unidade, Horário do fim do atendimento da unidade e Tolerância[5] em minutos para atrasos.
4. O operador preenche os dados e seleciona a opção correspondente a enviar.
5. A interface-administrativa solicita à API o cadastro do atendimento.
6. A interface-administrativa lista os atendimentos registrados.

Fluxos Alternativos: Não existem.

Fluxo de Exceção: Caso o operador não possua permissão para cadastrar um atendimento (após passo 5)

- A API retorna para a interface-administrativa um erro de permissão.
- A interface-administrativa mostra a mensagem de erro enviada pela API e nenhuma modificação é realizada.

Pós-Condições: O atendimento solicitado é criado e passa a ser listado para todos os operadores. Adicionalmente, é registrado para auditoria que o operador criou o atendimento em questão.

3.4.2.5 Listar clientes agendados

O presente caso de uso demonstra a funcionalidade que permite aos operadores verificar se um cliente está de fato agendado no horário de atendimento da unidade.

Para evitar violações relativas à privacidade dos clientes, a interface-administrativa mostra somente o CPF dos clientes e a foto do mesmo somente é exibida até que o cliente dê entrada na unidade. Ou seja, após a entrada do cliente, sua foto não é mais exibida caso a unidade esteja com o agendamento habilitado.

[5] A tolerância, quando com valor diferente de zero, permite que o sistema periodicamente verifique quais clientes não compareceram aos agendamentos. Com isso, caso um cliente não compareça sua vaga retorna ao quantitativo total (que é limitado pela quantidade prevista de refeições), permitindo que outros clientes utilizem essa vaga em um horário diferente para realizar sua refeição.

Nos casos onde a unidade está utilizando o sistema apenas como controle de acesso, a foto pode ser exibida durante todo o atendimento uma vez que é necessário conferir a identidade do cliente após sua entrada.

Pré-Condições: O operador deve estar autenticado na interface-administrativa e ter permissão para listar os clientes agendados.

Fluxo Principal:

1. A interface-administrativa mostra as opções disponíveis para o operador.
2. O operador seleciona a opção correspondente a mostrar os clientes agendados de um atendimento.
3. A interface-administrativa solicita à API a listagem dos clientes agendados no atendimento selecionado.
4. A interface-administrativa lista os clientes agendados no atendimento em questão.

Fluxos Alternativos:

Fluxo de Exceção: Caso o operador não possua permissão para listar os clientes de um atendimento (após passo 3)

- A API retorna para a interface-administrativa um erro de permissão.
- A interface-administrativa mostra a mensagem de erro enviada pela API.

Pós-Condições: Não existem.

3.4.2.6 Configurar atraso

Nas unidades que utilizam o agendamento, é recomendado a presença de telões para auxiliar a orientação dos clientes acerca do horário que eles podem começar a entrar na unidade. Porém, podem ocorrer atrasos no atendimento que fazem com que a tolerância configurada para a chegada dos clientes não seja efetiva.

Por exemplo, no caso de um agendamento para às 12:00 com tolerância de 15 minutos, um cliente possui um horário máximo de entrada até às 12:15. Caso ocorra um atraso interno à unidade de 10 minutos, e o cliente compareça às 12:10, ele seria atendido às 12:20 (devido ao atraso interno de 10 minutos) e teria seu

agendamento expirado, estando impossibilitado de ingressar na unidade até efetuar um novo agendamento.

Dessa forma, cada atendimento possui uma informação de atraso em minutos, inicialmente configurada com o valor padrão igual a zero. A necessidade desse caso de uso surgiu com a utilização de telões[6] para orientar os clientes sobre quais horários agendados estavam sendo chamados para adentrar no restaurante.

Uma vez que a configuração de atraso é realizada com um valor como 10 minutos, por exemplo, o horário exibido nos telões atrasa em 10 minutos, evitando aglomeração e questionamentos por parte dos clientes. Adicionalmente, a configuração de atraso evita que os agendamentos sejam expirados devido à atrasos ou problemas internos nos restaurantes.

Pré-Condições: O operador deve estar autenticado na interface-administrativa e ter permissão para editar os atendimentos.

Fluxo Principal:

1. A interface-administrativa mostra as opções disponíveis para o operador.
2. O operador seleciona a opção correspondente para editar o tempo de atraso de um atendimento.
3. A interface-administrativa solicita à API a edição do tempo de atraso no atendimento selecionado.
4. A interface-administrativa lista os clientes agendados no atendimento em questão[7].

Fluxos Alternativos: Não existem.

Fluxo de Exceção: Caso o operador não possua permissão para editar um atendimento (após passo 3)

- A API retorna para a interface-administrativa um erro de permissão.

[6] Os telões são utilizados apenas na unidade do CT e são compostos de monitores de computadores que mostram a interface-cliente em uma tela especialmente configurada. Um exemplo da tela é ilustrado na figura 4.19, p. 70.

[7] A interface-administrativa volta para a tela com a listagem de clientes agendados de forma a dar mais agilidade aos operadores durante o reestabelecimento do atendimento.

- A interface-administrativa mostra a mensagem de erro enviada pela API.

Pós-Condições: O atendimento tem sua informação de atraso alterada. Adicionalmente, é registrado para auditoria que o operador editou o atendimento em questão.

3.4.2.7 Receber pagamento

Esse caso de uso ilustra a funcionalidade de registrar o pagamento realizado por um cliente ao operador no momento de sua entrada na unidade.

Pré-Condições: O operador deve estar autenticado na interface-administrativa e ter permissão para editar os agendamentos.

Fluxo Principal:

1. A interface-administrativa mostra as opções disponíveis para o operador.
2. O operador seleciona a opção correspondente para receber o pagamento de um cliente.
3. A interface-administrativa solicita à API o registro de recebimento de pagamento para o cliente selecionado.
4. A interface-administrativa lista os clientes agendados no atendimento em questão.

Fluxos Alternativos: Caso o operador selecione que o cliente possui acessibilidade especial, ou seja, é isento de pagamento (durante passo 2)

- A interface-administrativa solicita à API o registro de isenção de pagamento para o cliente selecionado.
- A interface-administrativa lista os clientes agendados no atendimento em questão.

Fluxo de Exceção: Caso o operador não possua permissão para editar um registro de atendimento (após passo 3)

- A API retorna para a interface-administrativa um erro de permissão.
- A interface-administrativa mostra a mensagem de erro enviada pela API.

Pós-Condições: O agendamento associado ao cliente tem seu estado alterado para o correspondente a pago ou isento conforme o fluxo escolhido. Adicionalmente, é registrado para auditoria que o operador alterou o agendamento em questão.

3.4.2.8 Buscar agendamento

Esse caso de uso é baseado no cenário onde clientes comparecem às unidades sem lembrar seu horário de agendamento. Sem esse cenário, um operador teria que buscar pelas fotos o cliente em questão, o que poderia causar atrasos ou formação de filas. Logo, esse caso de uso apresenta uma forma de fornecer a informação do horário agendado ao cliente pelo operador[8].

Adicionalmente, esse caso de uso também tem a possibilidade de fornecer ao operador indícios de irregularidades, como clientes que utilizem as unidades repetidas vezes em um mesmo atendimento.

Pré-Condições: O operador deve estar autenticado na interface-administrativa e ter permissão para listar os clientes agendados.

Fluxo Principal:

1. A interface-administrativa mostra as opções disponíveis para o operador.
2. O operador seleciona a opção correspondente a busca de clientes agendados em um atendimento.
3. O operador digita o CPF do cliente que deseja buscar e seleciona a opção correspondente a enviar.
4. A interface-administrativa solicita à API o registro de agendamento do cliente informado.
5. A interface-administrativa mostra somente o registro de agendamento do cliente informado, caso ele exista. Caso não exista, a interface-administrativa não mostra nenhum registro.

Fluxos Alternativos: Não existem.

[8] O ideal é que os clientes utilizem a interface-cliente para esse tipo de consulta, conforme demonstra o caso de uso 3.4.1.3, p. 14. Porém, como forma de dar maior controle aos operadores, esse caso de uso também foi mapeado na interface-administrativa.

Fluxo de Exceção: Caso o operador não possua permissão para listar os clientes agendados (após passo 4)

- A API retorna para a interface-administrativa um erro de permissão.
- A interface-administrativa mostra a mensagem de erro enviada pela API.

Pós-Condições: Não existem.

3.4.2.9 Cadastrar agendamento

Esse caso de uso fornece ao operador a possibilidade de cadastrar um agendamento para um cliente. Nesse caso específico, esse agendamento utiliza o horário de sua requisição e não respeita as limitações de horário e capacidade configuradas no atendimento.

Essa funcionalidade apresenta essas características pois seu cenário de utilização é nas unidades que ainda estejam em processo de adaptação para disponibilizar o agendamento. Essas unidades podem utilizar o sistema apenas como mecanismo de controle de acesso, de forma que o operador, de posse do CPF do cliente, passa a ter a capacidade de cadastrar os agendamentos e realizar de maneira integrada o registro de entrada e pagamento.

Pré-Condições: O operador deve estar autenticado na interface-administrativa e ter permissão para cadastrar agendamentos nos atendimentos.

Fluxo Principal:

1. A interface-administrativa mostra as opções disponíveis para o operador.
2. O operador seleciona a opção correspondente ao cadastro de um agendamento.
3. O operador digita o CPF do cliente que deseja buscar, o estado do registro que deseja fazer (“Pendente”,“Pago”, “Residência Estudantil”) e seleciona a opção correspondente a enviar.
4. A interface-administrativa solicita à API o cadastro de um agendamento associado ao cliente informado.
5. A interface-administrativa lista os clientes agendados no atendimento em questão, inclusive o recém-cadastrado.

Fluxos Alternativos: Não existem.

Fluxo de Exceção: Caso o operador não possua permissão para listar os clientes agendados (após passo 4)

- A API retorna para a interface-administrativa um erro de permissão.
- A interface-administrativa mostra a mensagem de erro enviada pela API.

Pós-Condições: O horário atual é associado à um novo agendamento e este novo agendamento ao cliente. Adicionalmente, é registrado para auditoria que o operador cadastrou novo agendamento em questão.

3.4.2.10 Cancelar agendamento

O cancelamento do registro de agendamento é um caso de uso caracterizado pela necessidade de alguns clientes de tornar disponível o horário previamente registrado mas que se encontram impossibilitados de fazer isso por meios próprios[9].

Pré-Condições: O operador deve estar autenticado na interface-administrativa e ter permissão para cancelar clientes nos atendimentos.

Fluxo Principal:

1. A interface-administrativa mostra as opções disponíveis para o operador.
2. O operador seleciona a opção correspondente ao cancelamento de um registro de agendamento referente ao cliente em questão.
3. O operador seleciona a opção correspondente a enviar.
4. A interface-administrativa solicita à API o cancelamento do registro de agendamento do cliente informado.
5. A interface-administrativa lista os clientes agendados no atendimento em questão.

Fluxos Alternativos: Não existem.

[9] O caso de uso 3.4.1.4, p. 15, demonstra o cenário onde o cliente pode realizar o cancelamento através da interface-cliente.

Fluxo de Exceção: Caso o operador não possua permissão para cancelar os clientes agendados (após passo 4)

- A API retorna para a interface-administrativa um erro de permissão.
- A interface-administrativa mostra a mensagem de erro enviada pela API.

Pós-Condições: O registro de agendamento possui seu estado alterado para o correspondente a cancelado. Adicionalmente, é registrado para auditoria que o operador cancelou o registro de agendamento em questão.

3.4.2.11 Gerar relatórios

A confecção de relatórios é um caso de uso caracterizado pela necessidade da administração de cada unidade emitir relatórios que relacionem a quantidade de refeições servidas em um atendimento, tanto para fins de planejamento como com a finalidade de realizar sua prestação de contas.

Pré-Condições: O operador deve estar autenticado na interface-administrativa e ter permissão para listar registros de agendamento nos atendimentos.

Fluxo Principal:

1. A interface-administrativa mostra as opções disponíveis para o operador.
2. O operador seleciona a opção correspondente aos atendimentos que deseja emitir um relatório.
3. O operador seleciona a opção correspondente a gerar o relatório.
4. A interface-administrativa solicita à API a geração do relatório pedido.
5. A interface-administrativa renderiza os dados do relatório solicitado.

Fluxos Alternativos: Não existem.

Fluxo de Exceção: Caso o operador não possua permissão para gerar relatórios (após passo 4)

- A API retorna para a interface-administrativa um erro de permissão.
- A interface-administrativa mostra a mensagem de erro enviada pela API.

Pós-Condições: É registrado para auditoria que o operador gerou um relatório com os atendimentos informados.

3.4.2.12 Cadastrar clientes

O cadastro de clientes é um caso de uso caracterizado pela dificuldade da Universidade em fornecer informações de maneira centralizada sobre os clientes do restaurante, gerando fluxos de atendimentos diferenciados para alunos que participam de programas internos, como os alunos que residem na residência estudantil.

Dessa forma, para atender essa necessidade, o sistema possui a capacidade de registrar a identificação de clientes de forma priorizada, ou seja, é uma base que possui prioridade em detrimento do SIGA, possibilitando que dados adicionais sejam incluídos. Por exemplo, seria possível incluir que um cliente reside na residência estudantil ou que participa de um programa relevante para o restaurante.

O cadastro de clientes possibilita alterar o tipo (ou seja, a colocação do cliente na Universidade: Graduação, Servidor, Mestrado, entre outros) e o próximo estado (pago ou isento) de um agendamento após uma ação do operador. Assim, um cliente da residência estudantil pode fazer sua refeição subsidiada pela Universidade de forma integrada com o sistema, uma vez que o SIGA não possui essa informação.

Pré-Condições: O operador deve estar autenticado na interface-administrativa e ter permissão para cadastrar clientes.

Fluxo Principal:

1. A interface-administrativa mostra as opções disponíveis para o operador.
2. O operador seleciona a opção correspondente ao cadastro de um cliente.
3. O operador digita o CPF do cliente que deseja cadastrar, seleciona o tipo de cliente, o estado que os agendamentos devem possuir após uma ação do operador e seleciona a opção correspondente a enviar.
4. A interface-administrativa solicita à API o cadastro do cliente informado.
5. A interface-administrativa lista os clientes cadastrados, inclusive o recém-cadastrado.

Fluxos Alternativos: Não existem.

Fluxo de Exceção: Caso o operador não possua permissão para cadastrar clientes (após passo 4)

- A API retorna para a interface-administrativa um erro de permissão.
- A interface-administrativa mostra a mensagem de erro enviada pela API.

Pós-Condições: É registrado para auditoria que o operador cadastrou determinado cliente.

3.4.2.13 Listar operadores

A listagem de operadores é um caso de uso criado devido a necessidade de garantir um efetivo controle de acesso do sistema por parte da administração do restaurante. Assim, através da listagem de operadores, a administração do Sistema de Alimentação pode garantir que somente o mínimo pessoal necessário possui acesso ao mesmo.

Pré-Condições: O operador deve estar autenticado na interface-administrativa e ter permissão para listar operadores.

Fluxo Principal:
1. A interface-administrativa mostra as opções disponíveis para o operador.
2. O operador seleciona a opção correspondente a listagem de operadores.
3. A interface-administrativa solicita à API a listagem dos operadores cadastrados.
4. A interface-administrativa lista os operadores cadastrados.

Fluxos Alternativos: Não existem.

Fluxo de Exceção: Caso o operador não possua permissão para listar os operadores (após passo 3)

- A API retorna para a interface-administrativa um erro de permissão.
- A interface-administrativa mostra a mensagem de erro enviada pela API.

Pós-Condições: É registrado para auditoria que o operador cadastrou determinado cliente.

3.4.2.14 Cadastrar operadores

O cadastro de operadores é um caso de uso decorrente da necessidade do próprio sistema ter a possibilidade de cadastro, mesmo que seja de forma básica, de seus operadores.

Pré-Condições: O operador deve estar autenticado na interface-administrativa e ter permissão para cadastrar operadores.

Fluxo Principal:

1. A interface-administrativa mostra as opções disponíveis para o operador.
2. O operador insere uma identificação (por exemplo, o e-mail) para o novo operador e uma senha[10] para o mesmo.
3. A interface-administrativa solicita à API o cadastro do novo operador.
4. A interface-administrativa lista os operadores cadastrados.

Fluxos Alternativos: Não existem.

Fluxo de Exceção: Caso o operador não possua permissão para cadastrar operadores (após passo 3)

- A API retorna para a interface-administrativa um erro de permissão.
- A interface-administrativa mostra a mensagem de erro enviada pela API.

Pós-Condições: O operador com e-mail e senha informados é criado. Adicionalmente, é registrado para auditoria que o operador atualmente autenticado cadastrou o novo operador.

3.4.2.15 Atribuir permissões para operadores

Quando um operador é criado, o mesmo apenas possui um identificador e uma senha, caracterizando suas credenciais de acesso. Apenas de posse das credenciais, um operador não consegue operar o sistema, pois cada funcionalidade realizada pelo sistema necessita de uma ou mais permissões[11] para ser executada.

[10] A senha informada será enviada para a API. Na API, o armazenamento da senha será realizado através da função *scrypt* [25].

[11] As permissões disponíveis no sistema são detalhadas no item 4.6.2, p. 50.

Portanto, esse caso de uso tem a finalidade de permitir que um operador configure permissões para si e para outros operadores.

Pré-Condições: O operador deve estar autenticado na interface-administrativa e ter permissão para modificar operadores.

Fluxo Principal:

1. A interface-administrativa mostra as opções disponíveis para o operador.
2. O operador seleciona a opção correspondente ao operador que deseja atribuir permissões.
3. A interface-administrativa solicita à API as permissões do operador selecionado.
4. A interface-administrativa lista as permissões atuais do operador selecionado.
5. O operador seleciona as permissões que deseja atribuir (ou remover) do operador previamente selecionado.
6. A interface-administrativa solicita à API alterações nas permissões do operador previamente selecionado.
7. A interface-administrativa retorna o resultado da operação.

Fluxos Alternativos: Não existem.

Fluxo de Exceção: Caso o operador não possua permissão para listar operadores (após passo 3)

- A API retorna para a interface-administrativa um erro de permissão.
- A interface-administrativa mostra a mensagem de erro enviada pela API.

Pós-Condições: O operador selecionado possui suas permissões alteradas. Adicionalmente, é registrado para auditoria que o operador atualmente autenticado alterou as permissões do operador selecionado.

3.4.2.16 Verificar dispositivos de operadores

Ao efetuar a autenticação pela primeira vez na API, é necessário cadastrar a interface-administrativa como uma interface válida para aquele usuário conforme explicado no caso de uso Autenticar Operadores na página 17.

Pré-Condições: O operador deve estar autenticado na interface-administrativa e ter permissão para modificar dispositivos de operadores.

Fluxo Principal:

1. A interface-administrativa mostra as opções disponíveis para o operador.
2. O operador seleciona a opção correspondente ao operador que deseja verificar dispositivos.
3. A interface-administrativa solicita à API os dispositivos do operador selecionado.
4. A interface-administrativa lista os dispositivos atuais do operador selecionado.
5. O operador seleciona o dispositivo que deseja verificar para o operador previamente selecionado.
6. A interface-administrativa solicita à API a verificação do dispositivo para operador previamente selecionado.
7. A interface-administrativa retorna o resultado da operação.

Fluxos Alternativos: Não existem.

Fluxo de Exceção: Caso o operador não possua permissão para listar os dispositivos atuais de operadores (após passo 3)

- A API retorna para a interface-administrativa um erro de permissão.
- A interface-administrativa mostra a mensagem de erro enviada pela API.

Pós-Condições: O operador selecionado tem seu dispositivo marcado como verificado. Adicionalmente, é registrado para auditoria que o operador atualmente autenticado verificou o dispositivo do operador selecionado.

3.5 Conclusão

Com os casos de uso apresentados nesse capítulo tangibiliza-se o funcionamento do sistema tendo em vista as preocupações introduzidas pela equipe do Sistema de Alimentação da Universidade e do corpo técnico da Decania do CT (controle de acesso, capacidade de atendimento, mínimo impacto às atividades acadêmicas, facilidade de uso, privacidade, auditoria e autenticação segura).

A seguir, trabalha-se a arquitetura implementada, trazendo ao leitor detalhes das escolhas técnicas de implementação.

Capítulo 4

Implementação do Sistema

A arquitetura escolhida para o desenvolvimento do sistema foi de cliente-servidor [17, p. 170]. Além de ser adequada para a utilização em um sistema distribuído possui as vantagens de possibilitar ter diferentes clientes apresentando as informações conforme cada necessidade, centraliza a lógica de negócio em apenas um ponto e possibilita um melhor gerenciamento de segurança.

Dessa forma, para atender a todas as funcionalidades descritas nos casos de uso, o sistema foi dividido em partes, nas quais, respeitando a arquitetura cliente-servidor, a interface-cliente e a interface-administrativa desempenham o papel de clientes. A API, por sua vez, desempenha o papel de servidor de acordo com o modelo de arquitetura proposto.

Com a finalidade de aumentar a disponibilidade do sistema, cada uma das partes do sistema foi provisionada em mais de uma instalação. Ou seja, ao contrário de ter apenas uma máquina[1] com a interface-cliente instalada e configurada, foram preparadas 3 máquinas de modo que na falha de uma delas, outras estivessem disponíveis para atender os usuários do sistema. Essas máquinas estavam localizadas no Centro de Processamento de Dados da Decania do Centro de Tecnologia. Um cenário semelhante também foi realizado na aplicação responsável pela interface-

[1]Entende-se por máquina um conjunto de recursos como CPU e memória capaz de executar um programa de computador. No contexto desse trabalho, computadores especialmente preparados para funcionamento ininterrupto ou máquinas virtuais instaladas nesses computadores podem ser entendidos como máquinas. Evitou-se o uso da palavra "servidor" uma vez que a mesma poderia se confundir com um dos papeis na arquitetura escolhida.

administrativa, observando o cenário de utilização de cada uma delas.[2]

Contudo, para permitir que as 3 máquinas estivessem disponíveis e atuassem de forma integrada com a finalidade de manter o sistema disponível caso alguma viesse a falhar, utilizaram-se protocolos próprios para tal função. Dentro do desenvolvimento do sistema, houve a proposta de usar o protocolo DNS[3] para essa função, mas esse caminho foi impossibilitado devido a restrições no serviço utilizado pela Decania do CT. Finalmente, foi implantada a ferramenta KeepAlived [26] que implementa essa funcionalidade através do protocolo VRRP [27].[4]

Ainda assim, caso fosse necessário aumentar a capacidade de atendimento através de uma nova instalação, uma nova configuração das ferramentas de redundância teria que ser realizada. Para evitar esse cenário, uma camada de *proxies* foi provisionada tornando seu número de instalações fixo. Os *proxies* atuam como intermediários entre o servidor e o cliente no protocolo HTTP [28, p. 9], permitindo que outras funcionalidades sejam implementadas devido seu posicionamento lógico. No contexto do projeto, os *proxies* implementaram a terminação da conexão criptografada com o cliente e também o balanceamento[5] entre as máquinas com a aplicação provisionada.

Esse cenário permite que novas instalações para cada uma das aplicações (interface-cliente, interface-administrativa e API) sejam realizadas de forma dinâmica, uma vez que cada nova instalação somente precisa ser adicionada ou removida na configuração do conjunto de *proxies* (o que é realizado de maneira automatizada).

---

[2] O cenário de utilização da interface-cliente é caracterizado por um pico de acesso de centenas de clientes no momento que o agendamento das refeições é disponibilizado. A interface-administrativa, por ser utilizada somente por operadores, possui um número de acessos distribuído de forma uniforme ao longo do dia.

[3] A proposta de utilizar o protocolo DNS consistia de incluir múltiplos registros do tipo *A* para cada nova máquina. A desvantagem é que os navegadores web tentam cada um dos endereços na ordem cadastrada, de modo que caso a primeira e/ou segunda máquina estivessem indisponíveis, haveria impacto no tempo de acesso do usuário.

[4] Nesse protocolo, o conjunto de máquinas elege aquela que responderá pelo serviço. Em caso de falha nessa máquina, outra assume passando a responder pelo serviço no lugar da anterior.

[5] O balanceamento é responsável dividir uniformemente a utilização das aplicações entre os clientes. Logo, o balanceamento evita que um servidor com alguma aplicação fique sobrecarregado enquanto outro esteja mais disponível.

A figura 4.1 ilustra como cada parte se comunica com outras partes e recursos externos ao sistema. Pode-se observar o uso de *proxies*, no qual o *Proxy 2* é aquele que responde pelo serviço. Como o mesmo conjunto de *proxies* é utilizado tanto para a interface-cliente quanto pela interface-administrativa, é utilizado o cabeçalho *Host* do protocolo HTTP [28, p. 118] para diferenciar as conexões para cada uma das interfaces. Utilizar esse cabeçalho é conveniente pois é transparente para o usuário final, uma vez que esse cabeçalho é derivado da URL utilizada para o acesso.

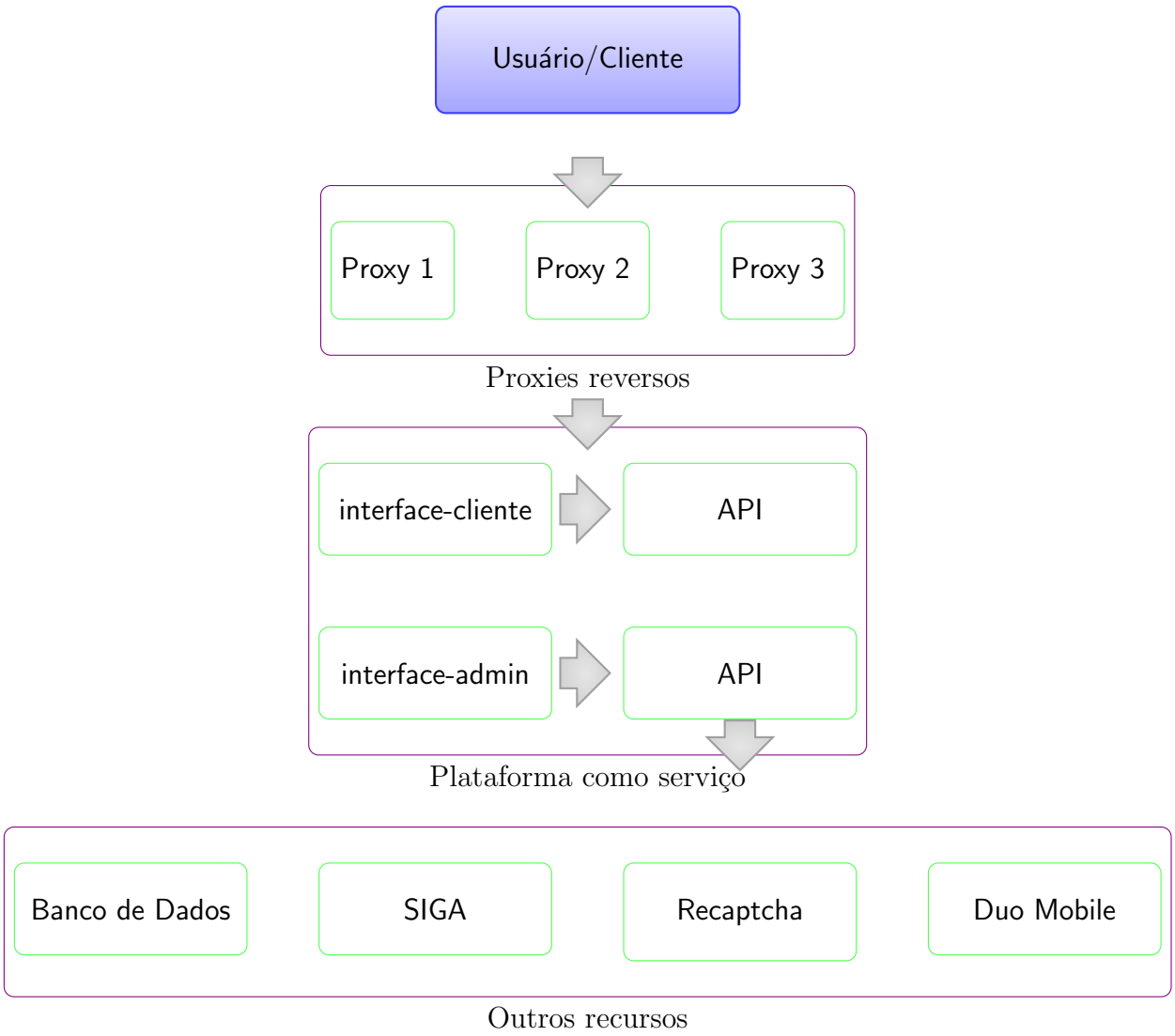

Figura 4.1: Diagrama de arquitetura do sistema.

Mais adiante, na figura 4.2 (p. 45), aborda-se em detalhes como a comunicação é realiza entre as partes do sistema.

4.1 Suporte ao sistema

Antes de detalharmos cada uma das partes do sistema, faz-se necessário introduzir a estrutura que fornece suporte para a execução do sistema.

Segundo Rentea [29, p. 7], durante o desenvolvimento de um software existe o risco de parte do esforço concentrar-se em atividades de manutenção, como o esforço em manter um ambiente consistente para as aplicações. Esse esforço pode ser descrito como aquele responsável por:

- instalar máquinas virtuais
- atualizar máquinas virtuais
- instalar dependências/bibliotecas para utilização das aplicações
- monitorar utilização de recursos (memória/CPU)
- redimensionar recursos (memória/CPU)
- realizar novas instalações sob demanda (devido ao aumento de tráfego, por exemplo)
- instalar novas versões das aplicações

Para evitar que parte do esforço fosse concentrado em atividades de manutenção e suporte, o sistema foi construído utilizando uma Plataforma como Serviço. Essa plataforma como serviço é responsável por fornecer de maneira estruturada as demandas mais comuns relacionadas com a manutenção do ambiente, seja ele o ambiente onde a aplicação é desenvolvida e validada, ou o ambiente onde a aplicação é disponibilizada para uso.

Para realizar a escolha entre as Plataformas como Serviço existentes, a familiaridade do autor e a necessidade de uma plataforma de fácil instalação foram critérios decisivos. Dessa forma, a plataforma escolhida foi o Tsuru [30].

Ainda sobre o suporte ao sistema, a utilização de software-livre foi priorizada em detrimento do uso de soluções proprietárias, levando-se em conta o equilíbrio entre custo de licenciamento, custo de manutenção e benefício para a Universidade. Os principais componentes utilizados pelo sistema são softwares disponíveis de forma

aberta, como as linguagens e *frameworks* utilizadas (item 4.4, p. 47), os protocolos escolhidos (todos com as especificações disponíveis) e a tecnologia de banco de dados (MySQL, com mais detalhes no item 4.6.6, p. 56). Os principais componentes com tecnologia proprietária são o sistema Duo Mobile e o *Google Recaptcha v2*, no qual o desenvolvimento de funcionalidades equivalentes no sistema demandaria um esforço desproporcional ao custo da solução proprietária.

4.2 Relação do sistema com a plataforma como serviço

Como o sistema não possui nenhum caso de uso mapeado relacionado com funcionalidades providas pela plataforma como serviço, o mesmo não possui nenhuma integração explícita com a mesma.

Mesmo assim, devido as tecnologias utilizadas pela Plataforma como serviço, um conjunto de práticas chamado *The Twelve Factor App* [31] foi adotado no desenvolvimento do sistema. Algumas dessas práticas incluem características como:

1. Uma base de código única com rastreamento através do controle de revisão, no qual foi adotado para o sistema a utilização do Github[6] como provedor desse serviço.

2. Dependências declaradas e isoladas, no qual cada parte do projeto utilizou o mecanismo de dependências mais apropriado, como o Go dep [32] para o código em Go e o Carton [33] para o código em Perl.

3. Configuração no ambiente, no qual caracteriza que as configurações das aplicações deve ser realizada de maneira estritamente separada do código. Nesse caso o projeto utilizou variáveis de ambiente.

4. Serviços de apoio como recursos anexados, no qual cada serviço utilizado (como banco de dados, por exemplo) possa ser trocado com mínimo impacto sobre o uso do sistema. Nesse caso o projeto trata suas conexões com o SIGA, banco

[6] O repositório utilizado para o controle de versão é https://github.com/decania-ct/ct, mas seu acesso deve ser solicitado através do e-mail informatica@ct.ufrj.br.

de dados e Duo Mobile como recursos que podem ser trocados por serviços equivalentes.

5. Separação do código de desenvolvimento do código em utilização, no qual uma versão do sistema é empacotada e enviada para os servidores de forma imutável. Essa prática garante um ambiente homogêneo e reproduzível.

6. Utilização de processos sem estado e sem compartilhamento de informações, no qual o sistema não armazena dados localmente (para isso ele utiliza recursos anexados) e nem depende de informações sobre seu estado anterior.

A utilização dessas práticas não apresenta nenhum tipo de aprisionamento tecnológico para a plataforma escolhida, Tsuru, mas sua utilização reduz o esforço de desenvolvimento das aplicações envolvidas. São através dessas práticas que decisões de arquitetura relacionadas a configuração, por exemplo, foram realizadas.

4.3 Comunicação entre componentes do sistema

De acordo com as interfaces levantadas nos casos de uso, uma arquitetura do tipo cliente-servidor mostrou-se mais relevante para o desenvolvimento da solução.

Dessa forma, os clientes e servidores precisam comunicar-se de forma eficiente, levando em conta suas respectivas funções especializadas e as necessidades de manutenção e integração entre clientes. O protocolo escolhido para lidar com a comunicação entre serviços foi o protocolo HTTP [28] devido sua relação com o modelo REST apresentado por Fielding [34]. O modelo REST possui uma série de vantagens das quais o sistema pode se aproveitar, sendo as principais:

Baseada em recursos/objetos: essa representação está de acordo com os dados trabalhados pelo sistema. Vimos que nos casos de uso, as principais operações estarão relacionadas aos agendamentos e atendimentos.

Sem estado: essa representação não necessita de um controle de estado, simplificando a implementação.

Cache: essa representação permite o uso de cache dos objetos, dessa forma evitando o uso exagerado de recursos de processamento e diminuindo a complexidade da aplicação.

Interface uniforme: essa representação utiliza os métodos HTTP como verbos para as ações que podem ser executadas no sistema. Essa interface permite a fácil adaptação e extensão do sistema.

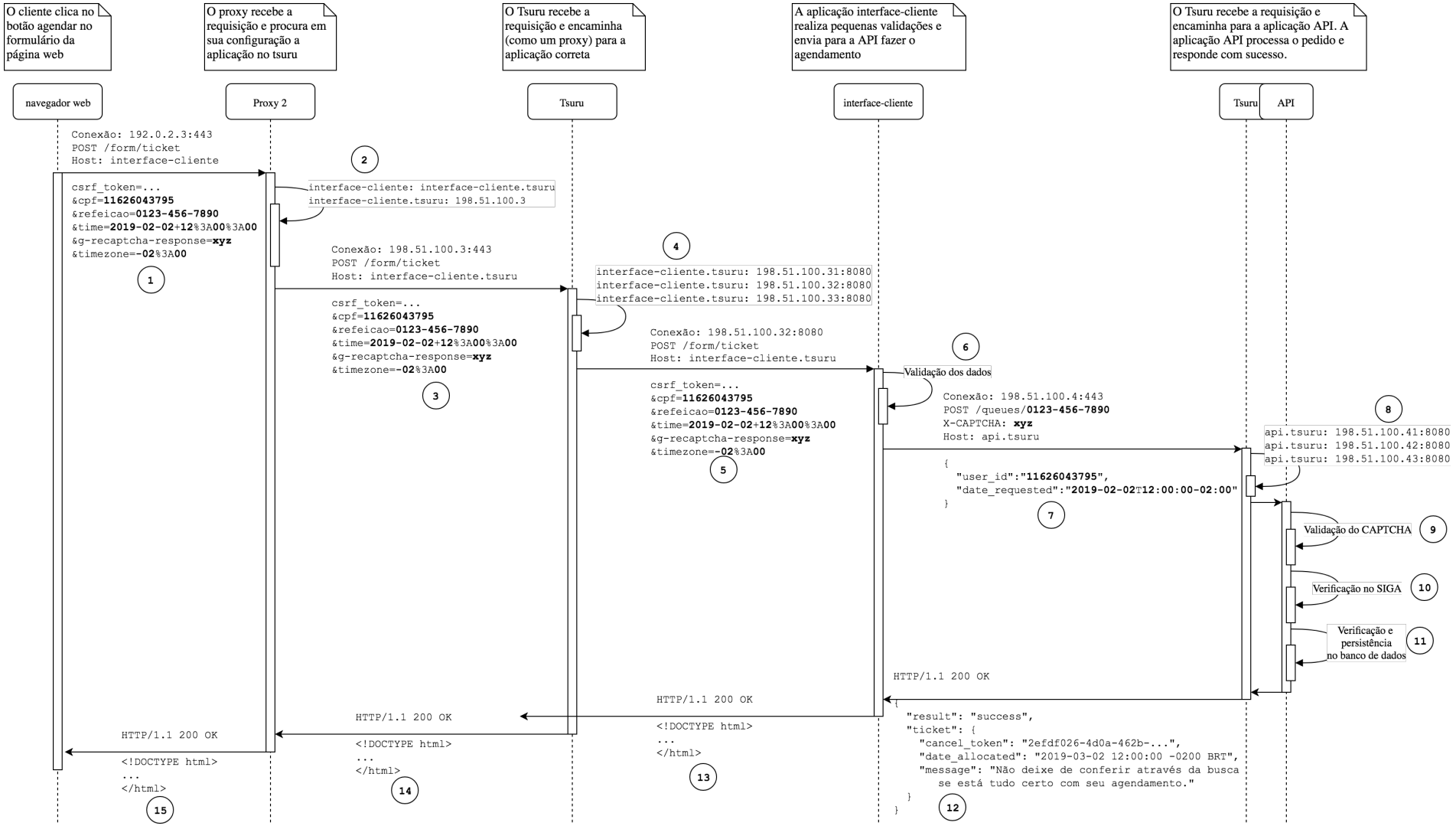

Figura 4.2: Diagrama de sequência para comunicação entre partes do sistema.

Um exemplo dessa comunicação utilizando o modelo REST e a arquitetura cliente-servidor em uma plataforma como serviço está representado na figura 4.2. Nessa figura, ilustra-se o caso de uso "Agendar horário" (3.4.1.2, p. 13).

Na parte 1 da figura 4.2, a requisição é realizada a partir do navegador web do cliente após o preenchimento do formulário (figura 4.10, p. 63) referente à solicitação de um agendamento. Verifica-se que os dados são enviados seguindo os padrões de formulário HTML e a conexão é realizada utilizando-se o protocolo HTTPS para um endereço IP (192.0.2.3[7]) tal que um dos *proxies* responda.

Na parte 2 da figura, temos a recepção da requisição pelo *Proxy 2*. Ele verifica em sua configuração qual aplicação na plataforma como serviço está configurada para que ele encaminhe a requisição. Para tal decisão, o *Proxy 2* verifica o cabeçalho *Host* enviado pelo navegador do cliente. O valor do cabeçalho é verificado na configuração e o *proxy* realiza uma conexão para o novo endereço (198.51.100.3). Verifica-se no passo 3 que o conteúdo da requisição não é modificado.

No passo 4, a plataforma como serviço (Tsuru) realiza a recepção da requisição oriunda do *Proxy 2*. O Tsuru[8] atua como um novo *proxy*, verificando em quais endereços IP a interface-cliente está disponível (no exemplo, 198.51.100.31, 198.51.100.32 e 198.51.100.33). Dessa forma, o Tsuru envia a requisição para uma das instâncias em funcionamento da interface-cliente, como pode-se observar no passo 5.

Posteriormente, no passo 6, a instância da interface-cliente que recebeu a requisição realiza uma validação dos dados enviados pelo cliente. Essa validação tem a finalidade de verificar se o tipo dos dados está de acordo com o esperado. Por exemplo, o campo CPF é composto apenas por números, de forma que a interface-cliente verifica se a composição desse campo é apenas numérica. Cabe ressaltar que a interface-cliente não verifica se o CPF informado pelo cliente faz parte ou não da comunidade acadêmica da Universidade.

No passo 7, a interface-cliente transforma os dados enviados via formulário para o padrão de formato utilizado pela API. Dessa forma, os dados são reorganizados e enviados como uma nova requisição à API no formato JSON (repare que a

[7] Os endereços IP mencionados são apenas para fins de exemplo.

[8] O Tsuru em si possui um componente chamado *router* que executa essa função.

API se encontra em outro endereço IP - 198.51.100.4).

Como a API também está na plataforma como serviço, o Tsuru recebe a requisição direcionada à API e atua no passo 8 de forma semelhante ao passo 4. Uma vez que uma instância da API recebe a requisição da interface-cliente, a API prossegue com mais validações (passos 9, 10 e 11), inclusive a verificação relativa ao pertencimento do CPF à comunidade acadêmica, e responde à interface-cliente os dados necessários para informar o cliente sobre sua solicitação, conforme ilustrado no passo 12.

A interface-cliente por sua vez, recebe os dados em JSON e converte-os para um página HTML. Essa página é repassada por toda a cadeia de comunicação (passos 13, 14 e 15) até chegar ao navegador do cliente (figura 4.11, p. 64).

4.4 Linguagens de programação utilizadas

Para desenvolver cada uma das interfaces, foi escolhida uma linguagem de programação tendo como critérios o conhecimento do autor e a economia de recursos. Para desenvolver as interfaces (interface-cliente e interface-administrativa), foi escolhida a linguagem Perl. Para desenvolver a API foi escolhida a linguagem Go.

Perl foi escolhido por ser uma linguagem interpretada, possibilitando um rápido desenvolvimento e uma flexível edição dos componentes utilizados. O mecanismo de modelos de páginas do *framework* Mojolicious idealizado por Riendel [35] possibilita o compartilhamento de código entre as interfaces diminuindo o esforço de desenvolvimento. De forma adicional ao Perl, as interfaces também utilizam JavaScript através do conjunto de componentes Bootstrap disponibilizado pelo Twitter [36].

A linguagem de programação Go foi escolhida devido o alto desempenho obtido pela mesma com um baixo esforço de desenvolvimento, o que são pontos relevantes para a API, uma vez que a mesma possui o papel de servidor na arquitetura escolhida. Outros pontos que embasaram a escolha dessa linguagem é a produção de binários estaticamente compilados por padrão e o uso facilitado de *coroutines* e *threads*.

Essas características são importantes uma vez que o uso de binários torna o

lançamento de novas versões mais rápido assim como a inicialização do sistema de forma geral. Outro efeito do uso de binários estaticamente compilados é que o consumo total de recursos do componente de API é reduzido em comparação com o uso de uma linguagem interpretada. Além disso, o uso facilitado de *coroutines* e *threads* possibilita um melhor uso dos recursos do processador para funções assíncronas, como verificações de consistência e envio de eventos de auditoria.

4.5 Disponibilização de interfaces para o usuário e cliente

Para ambas as interfaces, ao invés de disponibilizar as aplicações como aplicativos *desktop* com necessidade de instalação ou mesmo *download*, as interfaces foram disponibilizadas através de aplicações web que funcionam no navegador de computadores e dispositivos móveis. A disponibilização via web foi realizada para evitar o problema de atualização de versões em um ambiente distribuído de computadores, assim como para tornar a interface independente do sistema operacional utilizado.

Cabe ressaltar que como a API que executa todas as operações, outras interfaces podem ser implementadas. O trabalho apresentado por Podolan [4] implementa interfaces para aplicativos compatíveis com os sistemas operacionais Android e iOS para utilização em dispositivos móveis.

4.6 Interface da API

A interface para comunicação com a API, utiliza o protocolo HTTP com criptografia e é baseada no trabalho de Fielding [34] ao tratar recursos diretamente através dos métodos do protocolo.

Alguns desafios relacionados com APIs está relacionado ao controle de acesso, especificamente na autenticação de usuários e no permissionamento. Outros pontos desafiadores na API é na geração de eventos de auditoria, nas condições de corrida no agendamento e no *logout* de operadores.

4.6.1 Autenticação

A API realiza a autenticação dos usuários via web utilizando o conceito de autenticação multi-fator. Essa autenticação realiza uma verificação complementar ao fornecimento das credenciais (tradicionalmente, usuário e senha[9]) utilizando o dispositivo móvel do usuário.

Esse é um mecanismo compatível ao apresentado pelo autor na 29ª reunião do Grupo de Trabalho em Segurança [21]. No caso específico da interface-administrativa, como não é possível enviar uma mensagem *out-of-band* para o navegador, a alternativa foi criar um mecanismo de confiança para interface-administrativa.

Para fazer com que isso seja possível obedecendo às premissas de uma API REST, a interface-administrativa é cadastrada na API como uma interface confiável para autenticação. Dessa forma, a interface administrativa é marcado como confiável na API para autenticar usuários e, portanto, recebe os identificadores de autenticação diretamente (sem o envio *out-of-band*).

Uma vez que a autenticação multi-fator é realizada na interface-administrativa, a mesma pode se comparar com uma proteção equivalente, sendo que o meio apresentado não é o móvel e sim a web.

Finalmente, as características de autenticação do usuário são codificadas usando um JWT que é fornecido como identificador da sessão através interface-administrativa ao navegador. Esse identificador possui informações relativas à sessão, como a identificação do usuário (e-mail), suas permissões e um período de validade.

No cenário específico da interface-cliente, a mesma possui um JWT gerado apenas para ela, com um conjunto mínimo de permissões que representam as funcionalidades possíveis através de seu uso. Ou seja, como a interface-cliente precisa se comunicar com a API quando clientes fazem seus agendamentos, buscas e cancelamentos, ela possui um JWT válido na API em suas configurações. Esse JWT fixo nas configurações não fornece prejuízo à segurança do sistema uma vez que ele contém apenas as permissões mínimas para as funcionalidades expostas na interface-cliente.

[9] A API realiza o armazenamento das senhas utilizando a função *scrypt* [25].

4.6.2 Permissionamento

A API implementa um sistema granular de permissionamento, onde cada ação para cada tipo de recurso possui uma permissão específica.

Por exemplo, existe uma permissão específica para criar atendimentos (chamada *queues.create*) e outra específica para removê-los (chamada *queues.delete*), logo um determinado usuário pode ter permissão para criar atendimentos mas pode não ter permissão para excluí-los.

Como cada permissão é granular e específica, um mecanismo de hierarquia foi implementado. A proposta é que usuários que sejam gestores de atendimentos possam ter todas as permissões relativas a elas sem ter que adicionar uma por uma em seu controle de permissões. Assim, usuários que possuem a permissão para realizar a gestão de atendimentos (*queues*), já possuem (devido à hierarquia) a permissão para criar atendimentos (*queues.create*) e remover atendimentos (*queues.delete*).

Para obedecer às premissas REST e o modelo *The Twelve Factor*, o identificador de sessão (que contém as permissões) deve ser enviado em todas as solicitações para API. Caso alguma ação tente ser realizada sem o usuário ter as permissões necessárias, um erro é retornado pela API e nenhuma ação é realizada.

Na figura 4.3, ilustra-se a hierarquia de permissões dos atendimentos. Essa hierarquia permite que operadores com a permissão *queues* possam executar qualquer operação nos agendamentos. As operações da sub-árvore *tickets* chamadas de *schedule* e *search* se referem aos casos de uso de agendar horário (p. 13) e consulta de horário agendado (p. 14), respectivamente.

As permissões cuja árvore possui o nó *validated* são permissões que exigem a resolução de um teste do tipo CAPTCHA. Logo, para exemplificar, a permissão *queues.tickets.schedule.validated* exige que um teste do tipo CAPTCHA seja resolvido, enquanto a permissão *queues.tickets.schedule* não exige a resolução do teste.

Na figura 4.4 ilustra-se a árvore de permissões referente às operações com clientes do sistema. Como essas operações são realizadas apenas por operadores, nenhuma delas exige a solução de testes do tipo CAPTCHA (ou seja, nenhuma delas possui o nó mais restrito do tipo *.validated*).

Na figura 4.5 ilustra-se a hierarquia de permissões referentes às entidades dos operadores. Essas permissões são importantes pois permitem que um número

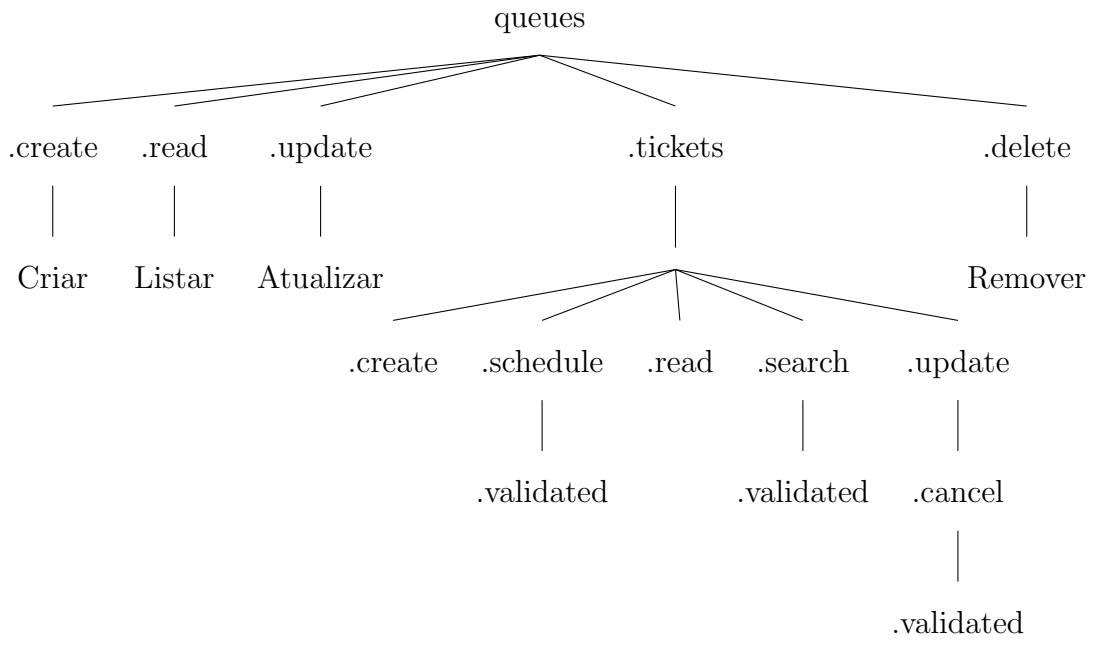

Figura 4.3: Árvore de permissões hierárquicas dos atendimentos.

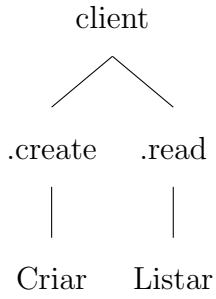

Figura 4.4: Árvore de permissões hierárquicas dos clientes.

reduzido de operadores possam criar novas credenciais para outros operadores, dessa forma controlando e centralizando o controle de acesso ao sistema. Como essas operações são realizadas apenas por operadores, nenhuma delas exige a solução de testes do tipo CAPTCHA (ou seja, nenhuma delas possui o nó mais restrito do tipo *.validated*).

4.6.3 Auditoria

Conforme levantado nos casos de uso, grande parte das funcionalidades expostas pela API possuem a necessidade de gerar registros de auditoria informando a

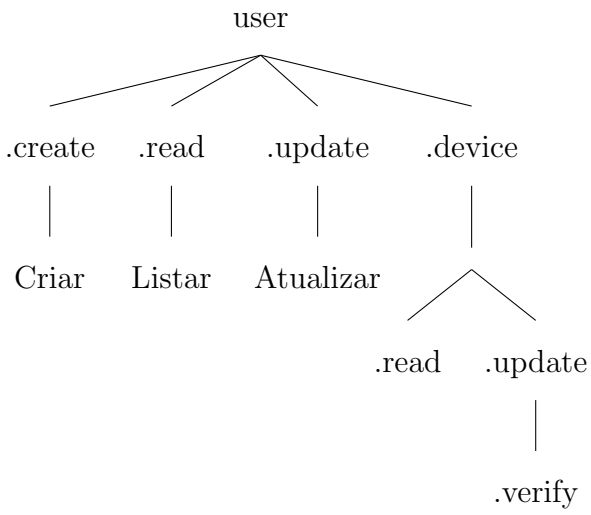

Figura 4.5: Árvore de permissões hierárquicas dos operadores.

ação correspondente. Isso significa que para cada ação na API, além das verificações de autenticação, permissões e consultas de validação na base de dados, mais uma operação de escrita teria de ser realizada.

Nas bases de dados compatíveis com o sistema (SQL), operações de escrita possuem um custo maior que operações de leitura [37] atuando como um limitador de performance para o sistema.

Uma vez que a preocupação relacionada com a auditoria tem como origem o corpo técnico do CT (item 3.2, p. 8), essa necessidade poderia ser retirada do sistema uma vez que a mesma realiza uma escrita no banco de dados e, consequentemente, possui um custo maior e degrada a performance geral do sistema. Ou seja, as funções de auditoria poderiam ser retiradas do sistema com a finalidade de diminuir os tempos de resposta do sistema.

De forma a evitar uma penalização da performance do sistema e consequente motivação para retirada da funcionalidade de geração de registros de auditoria, o sistema implementa a escrita desses registros de forma assíncrona para o banco de dados. Assim, cada processo em execução da API possui uma *coroutine* que recebe os eventos de auditoria (através de um endereço compartilhado de memória), enfileirando os registros e armazenando-os no banco de dados sem impactar na performance geral do sistema.

4.6.4 Condições de corrida: agendamento

O caso de uso Agendar horário descrito na página 13 possui algumas verificações que devem ser realizadas na API durante o passo 8 de seu fluxo principal. Essas verificações são:

1. Autenticação.
2. Permissionamento.
3. CAPTCHA.
4. Existência do atendimento informado.
5. Confirmação se o horário pedido está no intervalo de tempo permitido.
6. Confirmação que o atendimento selecionado está disponível para agendamentos.
7. Confirmação que ainda existem vagas no atendimento selecionado.
8. Verificação se o cliente informado está cadastrado como um cliente interno do sistema.
9. Confirmação que o cliente informado está cadastrado no SIGA.
10. Verificação se já existe algum outro agendamento pendente[10] para o cliente[11].
11. Verificação se ainda existem vagas no horário pedido pelo cliente ou nos horários seguintes.

Todas essas verificações são realizadas antes de se fazer qualquer alteração na base de dados. Essas condições introduzem um cenário de condição de corrida[38] uma vez que todas as verificações podem ocorrer de forma paralela (principalmente os itens 7, 10 e 11), possibilitando que um cliente realize um agendamento quando

[10]Entende-se como um agendamento pendente aquele que foi realizado pelo cliente, mas que o mesmo ainda não deu entrada no restaurante para consumir sua refeição.

[11]Foi estabelecido que seria desejável que um cliente do sistema não pudesse ter mais de um agendamento pendente por atendimento. Da mesma forma, foi estabelecido que um cliente, após a realização de sua refeição, poderia agendar novamente em outro horário.

não existam mais vagas em um horário ou que realize mais de um agendamento para determinado atendimento.

Uma forma de evitar o abuso do sistema é evitar criar um novo registro de agendamento e utilizar as propriedades de ACID do banco de dados. Essas propriedades seriam utilizadas para lidar com o problema da atomicidade dessa operação em um sistema distribuído e sem estado como é o caso da API.

Logo, para que a operação de agendar um horário fosse resistente à condições de corrida, o caso de uso relativo ao cadastro de atendimentos (p. 23) deveria ser capaz de inserir no banco de dados todas as possibilidades de horário para agendamentos. Dessa forma, quando um atendimento é cadastrado são geradas entidades internas no sistema chamadas de *tickets* que preenchem todas as posições de horário possíveis de um atendimento[12].

Para diferenciar *tickets* que podem ser atualizados de *tickets* que já estão associados à um determinado cliente, cada *ticket* possui um estado interno que inicia como disponível. Dessa maneira, quando um cliente solicita um agendamento, a API apenas atualiza[13] o próximo *ticket* disponível que possua o horário igual ou posterior ao horário solicitado pelo cliente. Essa operação é realizada apenas caso não exista nenhum outro *ticket* com estado pendente para tal cliente.

Como forma de garantir a consistência do banco de dados, a API executa uma *coroutine* periodicamente[14] verificando por inconsistências para os atendimentos ativos. Essa *coroutine* verifica os contadores internos do sistema[15] e se existem clientes com mais de um agendamento pendente cadastrado em um atendimento.

[12]Para exemplificar, um atendimento com capacidade de 10 pessoas por minuto e com duração de 1 hora teria 600 *tickets* criados assim que fosse cadastrado. Os *tickets* não podem ter seu horário alterado, logo existirão 10 *tickets* referentes ao 1º minuto do horário, mais 10 *tickets* referentes ao 2º minuto do horário, e assim até o último minuto do horário com os últimos 10 *tickets*, totalizando assim 600 *tickets*.

[13]Entende-se por atualizar o *ticket* como associar o mesmo ao cliente que solicitou o horário a qual o *ticket* corresponde e alterar seu estado para pendente.

[14]Essa verificação é executada a cada 10 minutos.

[15]Esses contadores são utilizados pelos operadores para verificar de forma rápida quantos agendamentos estão expirados ou cancelados.

4.6.5 Logout de operadores

Conforme mostrado no caso de uso Autenticar Operadores (p. 17), a gestão de sessões é realizada utilizando JWTs na interface-administrativa. Esse cenário é decorrente da API enviar um JWT no momento que o operador realiza a autenticação com sucesso. Ao invés da interface-administrativa tratar esse JWT internamente e expor somente um identificador aleatório como *cookie*, por simplicidade, a interface-administrativa expõe o JWT ao navegador assim não precisando realizar esse tratamento a cada chamada na API.

O uso do JWT fornece algumas garantias uma vez que o mesmo é um objeto assinado criptograficamente, logo não é passível de modificação por parte do usuário. Dessa forma, o JWT enviado pela API pode conter informações não sigilosas do usuário, como suas permissões, por exemplo. Algumas outras informações que um JWT contêm são a data de geração do mesmo, qual sistema fez sua emissão, uma identificação única e uma data de expiração.

É justamente na data de expiração do JWT que o problema de *logout* de operadores se manifesta. Como um JWT não pode ser modificado, se um operador realiza *logout* do sistema, seu JWT anterior continua válido até sua data de expiração. Da mesma forma, não seria possível revogar a autenticação de um operador, caso seja necessário[16].

Para exemplificar, quando um operador se autentica no sistema, a API gera um JWT válido por um determinado período de tempo (nesse exemplo, vamos considerar que é gerado um JWT válido até às 22:00 do dia atual). Caso o operador faça *logout* no sistema às 18:00, esse JWT continua sendo criptograficamente válido até às 22:00. Em uma situação hipotética onde algum dispositivo usado pelo operador tenha sido comprometido após seu uso, esse JWT poderia ser recuperado e colocaria a segurança do sistema em risco.

De forma a mitigar esse cenário sem penalizar a performance do sistema, a API possibilita que seja gerada uma lista negra de identificadores únicos de JWT. Porém, apenas esse cenário poderia degradar a performance do sistema uma vez que cada requisição precisaria ter o seu JWT verificado nessa lista. Logo, para evitar um

[16] Um exemplo de cenário onde uma revogação de autenticação possa ser necessária é caso um dispositivo móvel autenticado de um operador venha a ser comprometido.

cenário de consultas excessivas para verificação na lista negra da base de dados, foi criada uma *coroutine* que carrega a lista proveniente do banco de dados na memória do processo da API periodicamente.

Além disso, a *coroutine* verifica se o JWT cujo identificador único está na lista negra já seria inválido devido sua expiração criptográfica, mantendo na memória apenas os JWTs que sejam válidos criptograficamente, mas inválidos para uso no sistema.

4.6.6 Comunicação com outros recursos: banco de dados e CAPTCHA

A API possui a necessidade de se comunicar com outros sistemas para realizar o armazenamento e consulta de informações.

Durante a concepção da API, não houve a necessidade de nenhuma outra dependência específica de uma tecnologia de banco de dados, somente a necessidade de consistência e atomicidade conforme detalhado no item 4.6.4 (p. 53). Essa ausência de restrições possibilitou que a API utilizasse uma interface agnóstica para o armazenamento e recuperação de suas informações no banco de dados. Por causa dessa ausência, a API suporta uma ampla gama de implementações de banco de dados como MySQL, SQLite, PostgreSQL ou SQL Server.

Embora a API implemente uma conexão autenticada utilizando o protocolo relacionado ao banco SQL, a conexão não utiliza criptografia para os dados trafegados. Mesmo assim, a API deve ser resiliente à falhas temporárias de conexão com o banco de dados de forma a não gerar erros inesperados.

Para lidar com falhas de comunicação com o banco de dados, a API utiliza *timeouts* e tentativas de reconexões acumulativas no tempo. Ou seja, a API limita o tempo de espera de uma comunicação com o banco de dados e faz até 20 tentativas esperando 1 segundo entre a 1ª e a 2ª tentativa, 2 segundos entre a 2ª e 3ª tentativa, até esperar 19 segundos entre a 19ª e 20ª tentativa.

Outro sistema que a API depende é o SIGA a partir de onde os dados de clientes são verificados. Essa é uma verificação através do protocolo HTTP com criptografia, mas sem ser autenticada.

Finalmente, a API tem a necessidade de se comunicar com a API do *Google*

Recaptcha v2[17] para a validação de tentativas de agendamentos e proteção do sistema contra acesso automatizado. Essa conexão também é realizada utilizando o protocolo HTTP com criptografia e utiliza uma chave privada para autenticação.

4.6.7 Recursos disponibilizados

A API disponibiliza seus recursos através de rotas usando métodos HTTP. A seguir, listamos todas as rotas disponíveis com uma breve descrição da funcionalidade de cada uma delas.

Cada uma das rotas é acessível via protocolo HTTP com criptografia, utilizando o método descrito no início (GET, POST, PUT) e o *path* indicado logo depois (*/status*). A presença de campos iniciados por dois pontos (*:*) mostra que o valor desses campos é variável e portanto deve ser substituído por um identificador ou outra informação relevante. Todas as rotas exigem autenticação, exceto quando explicitado o contrário.

A lista de rotas a seguir tem o carater apenas informativo, mais detalhes podem ser obtidos na documentação atualizada pelo autor em [39].

4.6.7.1 Operação da aplicação

GET /status/live Rota que retorna se a aplicação está funcionando (se o processo está ativo). Essa rota não exige autenticação.

GET /status/ready Rota que verifica se a aplicação está pronta para o uso (se todos os recursos necessários para o seu funcionamento estão disponíveis). Essa rota não exige autenticação.

GET /status/token Rota que retorna 200 OK se o JWT enviado como cabeçalho estiver válido.

[17] Existe um conflito de escolha na utilização do *Google Recaptcha v2* como provedor da solução de CAPTCHA para o sistema. Esse conflito de escolha é justificado com base em preocupações de privacidade (os dados coletados pela Google durante a utilização da solução) e na utilização de um sistema proprietário para tal finalidade. A utilização dessa plataforma foi feita uma vez que existe um alto custo de desenvolvimento e manutenção de uma plataforma semelhante.

4.6.7.2 Manipulação de dados de atendimentos

GET /queues Rota que lista os atendimentos cadastrados no sistema.

POST /queues Rota que cria um atendimento.

GET /queues/:uid Rota que retorna os dados cadastrados no sistema para um determinado atendimento (identificado pelo *uid*).

POST /queues/:uid Rota que aloca um agendamento em determinado atendimento.

PUT /queues/:uid Rota que permite modificar atributos do atendimento.

GET /queues/:uid/tickets Rota retorna os agendamentos cadastrados para determinado atendimento.

POST /queues/:uid/tickets Rota que cria um novo agendamento no atendimento.

PUT /queues/:uid/tickets/:user_id Rota que atualiza o estado de um agendamento (onde *user_id* é o CPF de um cliente).

GET /queues/:uid/tickets/:user_id Rota que busca os agendamentos de um cliente.

4.6.7.3 Manipulação de dados de operadores

GET /users Rota que lista os operadores cadastrados no sistema.

POST /users Rota que cadastra um operador no sistema.

PUT /users/:id Rota que atualiza os dados de um operador no sistema (onde *id* é o identificador do operador).

GET /users/:id/devices Rota que retorna os dispositivos associados a um operador.

PUT /users/:id/devices/:device_id Rota que atualiza um dispositivo de um operador, tornando ele verificado (onde *device_id* é um identificador interno para o dispositivo).

POST /users/token Rota que envia um JWT para os dispositivos cadastrados do usuário.

4.6.7.4 Manipulação de dados de clientes

GET /clients Rota que lista os clientes atualmente cadastrados no sistema.

POST /clients Rota que cadastra clientes no sistema.

POST /clients/token Rota que envia um JWT para o dispositivo de um cliente ao utilizar um aplicativo móvel.

4.7 Interface do Cliente

A interface-cliente é a responsável por atender os pedidos dos clientes do sistema referentes a agendamentos em atendimentos de sua escolha. O acesso a essa interface ocorre via protocolo HTTP com criptografia através do endereço https://ru.ct.ufrj.br. Para o propósito deste trabalho, uma versão do sistema foi instalada em um domínio de posse do autor para que as demonstrações de algumas funcionalidades não afetasse o atendimento real dos restaurantes, isto é, separando o ambiente de desenvolvimento do ambiente de produção.

Embora as imagens mostradas nessa seção sejam realizadas utilizando navegadores de computadores de mesa, a interface-cliente foi projetada levando-se em consideração sua utilização em dispositivos móveis.

4.7.1 Navegação inicial no sistema

A navegação inicial no sistema começa por uma página (figura 4.6) que contém apenas duas opções. A primeira para os clientes visualizarem os atendimentos disponíveis e a segunda para buscar agendamentos já realizados em atendimentos ainda em atividade.

Além dessa interface, o cliente poderia deparar-se com uma mensagem indicando a indisponibilidade do sistema (figura 4.7) caso algum dos recursos necessários para o funcionamento do mesmo estivesse indisponível.

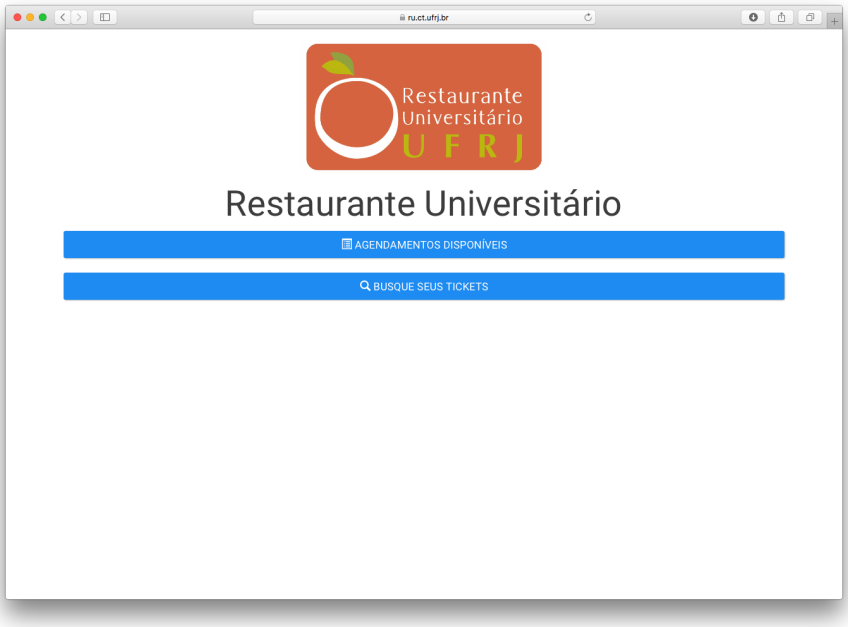

Figura 4.6: Página inicial da interface-cliente do sistema.

Figura 4.7: Página inicial da interface-cliente do sistema apresentando erro ao cliente.

Na primeira versão da interface-cliente (figura 4.8), existia uma terceira opção na página inicial do sistema que fornecia acesso direto do cliente ao formulário para cadastro de um agendamento. Essa opção foi removida pois os clientes deixavam de

verificar o cardápio do dia e o quantitativo de vagas e horários antes de realizar o agendamento.

Esse comportamento dos clientes gerava desentendimento com os atendentes das unidades e portanto a interface-cliente refletiu uma solução retirando o botão correspondente a essa opção. Assim, os clientes precisam realizar uma navegação forçada passando pelas informações dos atendimentos disponíveis antes de solicitar um agendamento.

Figura 4.8: Página inicial da interface-cliente do sistema com 3 botões (desativada).

4.7.2 Visualização de atendimentos disponíveis

Após a entrada do cliente na página inicial, o primeiro botão da interface leva o cliente à uma lista com os atendimentos disponíveis. Conforme ilustra a figura 4.9, a listagem de atendimentos traz informações que podem orientar o cliente na sua escolha pela unidade preferível.

Nessa listagem de atendimentos, somente os atendimentos com o estado "disponível"são exibidos pela interface-cliente. Isso ocorre pois os atendimentos possuem um estado interno que pode variar entre as opções "Disponível", "Privado"e "Fechado".

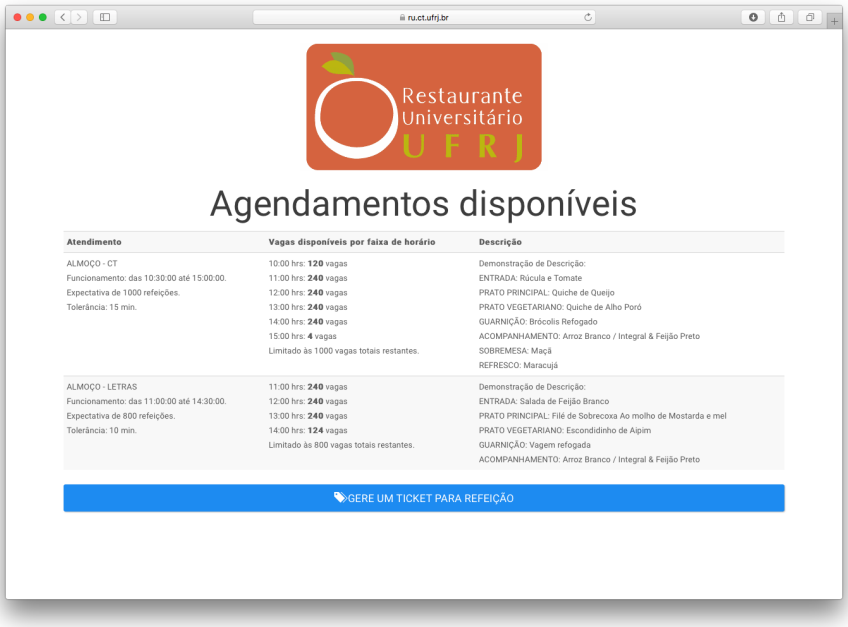

Figura 4.9: Página da interface-cliente do sistema com listagem de atendimentos disponíveis.

Essas informações são divididas em 3 colunas na tela, sendo a primeira coluna responsável por apresentar os dados referentes ao tipo de refeição (“Café da Manhã”, “Almoço” ou “Jantar”, por exemplo), à unidade do restaurante (na figura temos exemplos de atendimentos disponíveis na unidade de Letras e CT), aos horários de funcionamento de cada restaurante (cada unidade por ter horários diferentes conforme ilustrado na imagem). Por fim, caso exista alguma tolerância para a chegada do cliente, a mesma é mostrada (no exemplo temos que a unidade de Letras possui uma tolerância de 10 minutos enquanto a unidade do CT possui uma tolerância de 15 minutos).

A segunda coluna (conforme mostra a figura 4.9), apresenta estatísticas de preenchimento dos horários do atendimento. Essa informação pode ser utilizada pelos clientes para comparar os horários com mais vagas e sua respectiva disponibilidade em função das atividades acadêmicas.

Finalmente, a terceira coluna da figura 4.9, apresenta uma descrição configurada no momento de cadastro do atendimento. Essa descrição é um texto livre informado pelo operador e nos exemplos da figura, o operador colocou o cardápio referente ao atendimento em questão.

4.7.3 Agendamento de horário para refeição

Uma vez que o cliente tenha verificado os atendimentos disponíveis e as informações relativas às vagas em cada um deles, o mesmo pode prosseguir para a realização de um agendamento.

Nessa situação, o cliente deve informar seu CPF, o qual será utilizado para validar seu estado como integrante da comunidade acadêmica da Universidade. Após inserir o CPF, o cliente deve escolher o horário pretendido através de uma caixa que fornece opções com intervalos de 5 minutos. Por fim, o cliente seleciona o atendimento que deseja e resolve o teste do tipo CAPTCHA, podendo efetuar seu pedido ao sistema.

A figura 4.10 ilustra o formulário preenchido pelo cliente.

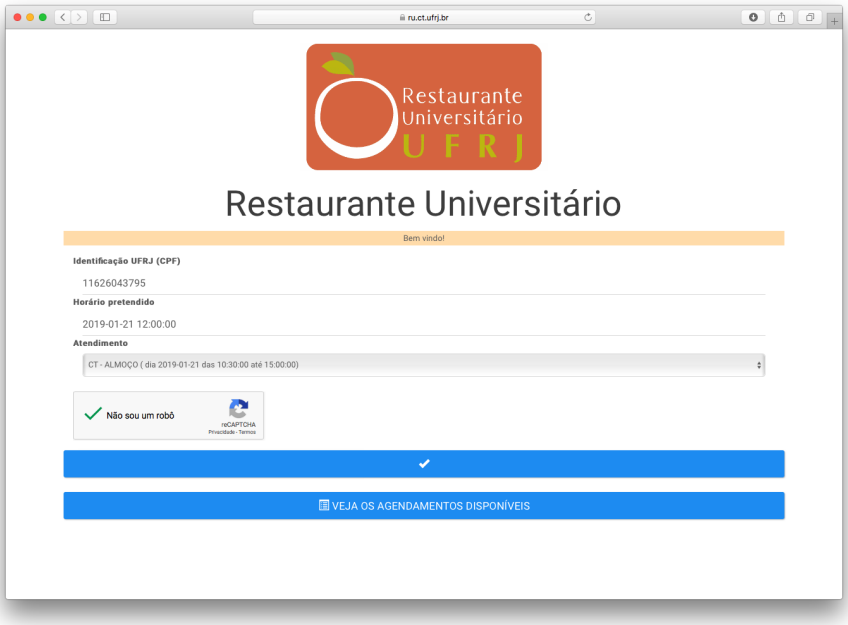

Figura 4.10: Página da interface-cliente do sistema com formulário para agendamento.

Após a solicitação do cliente, o sistema pode emitir duas respostas. Caso o horário solicitado pelo cliente esteja vago e ele tenha conseguido um agendamento no horário pedido, o cliente verá como resposta de sua solicitação a tela ilustrada na figura 4.11.

Caso o cliente não consiga um agendamento no horário solicitado por ele e ainda existam vagas, o sistema selecionará automaticamente o próximo horário

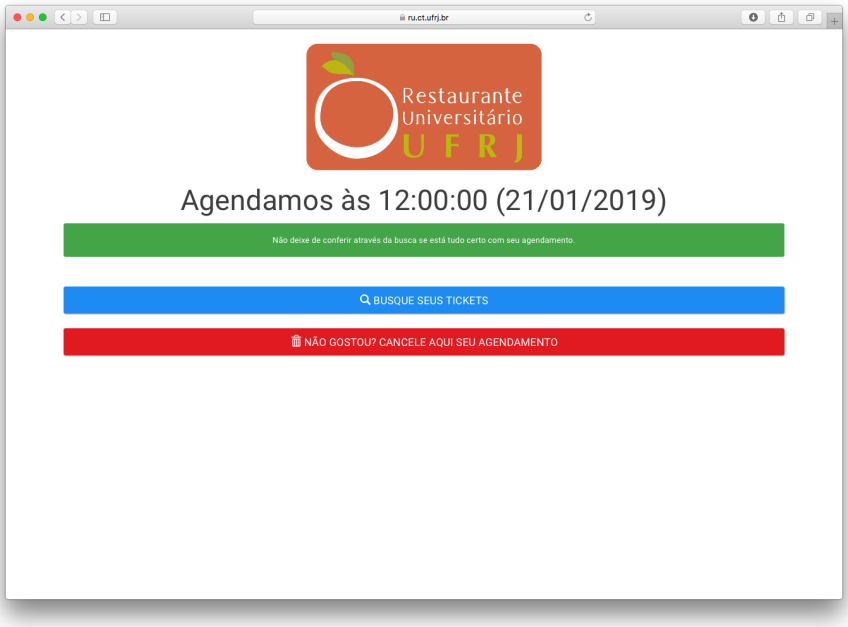

Figura 4.11: Página da interface-cliente do sistema com resposta para agendamento no horário solicitado pelo cliente.

disponível e apresentará para o usuário como resposta de sua solicitação a tela ilustrada na figura 4.12.

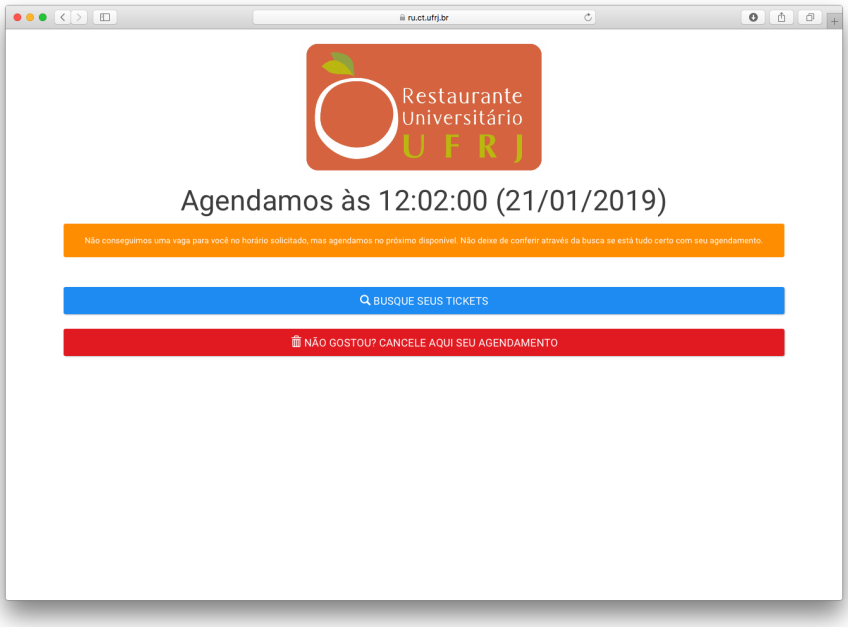

Figura 4.12: Página da interface-cliente do sistema com resposta para agendamento em horário diferente do solicitado pelo cliente.

Os elementos coloridos mostrados na figura 4.12 foram modificados para alertar o usuário que o horário agendado não é o mesmo solicitado por ele. A escolha da cor (correspondente à #FF9800 no sistema de cores aditivas RGB) foi realizada pelo *framework* Bootstrap utilizado no projeto.

Além dessas respostas, a interface-cliente suporta a geração de QR Codes [40] conforme ilustrado na figura 4.13. Essa funcionalidade permite a integração da interface-cliente com o aplicativo desenvolvido no trabalho de Podolan [4].

Figura 4.13: Página da interface-cliente do sistema com resposta para agendamento com QR Code.

Finalmente, existe a possibilidade do cliente não ser um membro da comunidade acadêmica da Universidade. Nesse caso, o sistema orienta o cliente com a seguinte mensagem: "Usuário não foi encontrado, verifique suas pendencias cadastrais no DRE (no CCMN) ou na sua secretaria.". Essa mensagem é ilustrada na figura 4.14 e possui o presente conteúdo para fornecer orientação aos clientes no caso de inconsistências em seus registros. No caso, a Divisão de Registro de Estudantes (DRE) ou a secretaria de curso foram os locais escolhidos para encaminhar os clientes nessa situação.[18]

[18]No final de 2018, esse cenário passou a ser tratado internamente pelo sistema através da

Figura 4.14: Página da interface-cliente do sistema para clientes com pendências cadastrais.

4.7.4 Busca de agendamento

A interface-cliente possibilita que um cliente realize uma busca para determinar o horário de seu agendamento. A partir da navegação da página inicial, o cliente é capaz de selecionar um atendimento e informar seu CPF. A figura 4.15 ilustra esse formulário.

Caso exista algum agendamento, o mesmo é listado para o cliente como resposta de sua solicitação. De maneira complementar, caso o cliente esteja solicitando a busca do mesmo dispositivo que realizou o agendamento, um botão de cancelamento é mostrado ao lado do resultado. A figura 4.16 ilustra a resposta recebida pelo cliente nesse cenário.

alteração da classificação do cliente ("Graduação", "Mestrado", "Servidor", por exemplo) para o valor "Ver documentos". Acredita-se que dessa forma seja possível um melhor tratamento para alunos que ainda não estejam cadastrados adequadamente no SIGA.

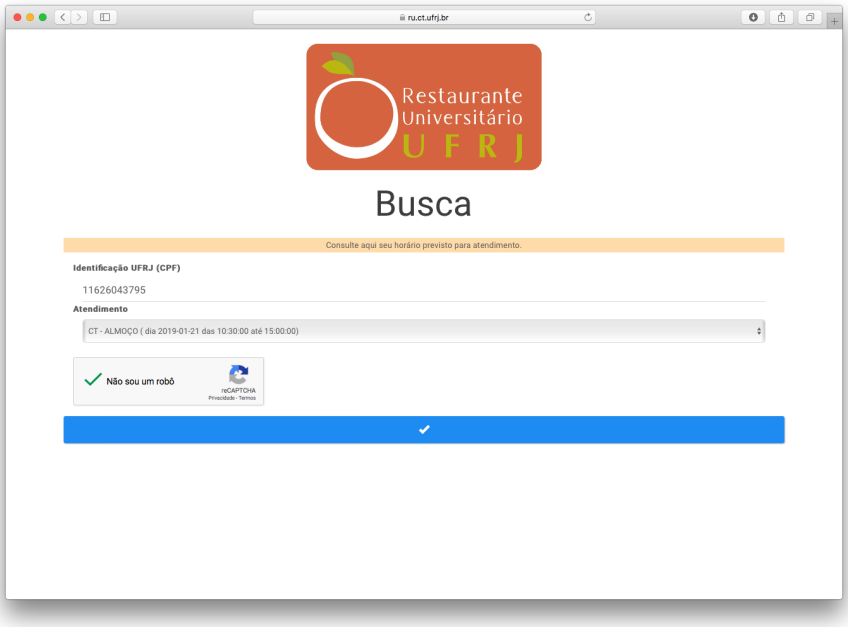

Figura 4.15: Página da interface-cliente do sistema com formulário para busca de agendamento.

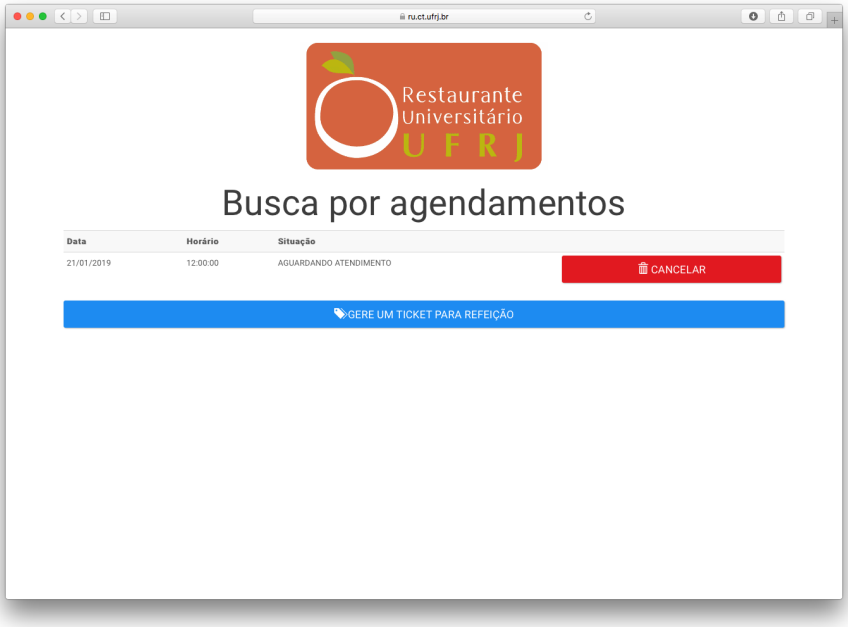

Figura 4.16: Página da interface-cliente do sistema com resultado da busca realizada no mesmo dispositivo do agendamento.

4.7.5 Cancelamento de agendamento

A interface-cliente possibilita que um cliente realize o cancelamento de seu agendamento em dois casos.

O primeiro caso disponibilizado pela interface-cliente é logo após o agendamento, através de um botão vermelho conforme ilustrado pela figura 4.11 e pela figura 4.12.

O segundo caso disponibilizado pela interface-cliente é logo após a realização de uma busca no mesmo dispositivo utilizado para o agendamento. A interface-cliente nesse cenário também exibe um botão vermelho conforme ilustrado pela figura 4.16.

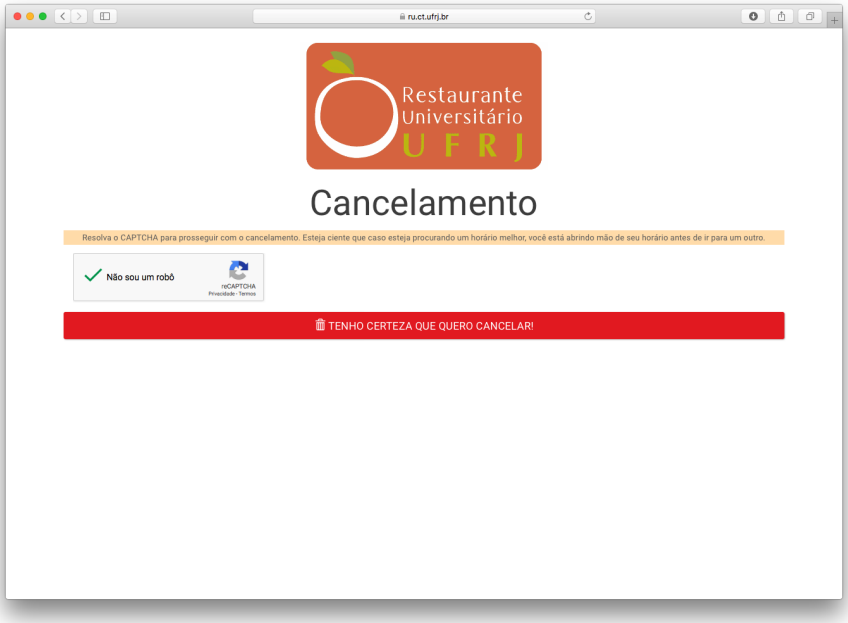

Figura 4.17: Página da interface-cliente do sistema com teste CAPTCHA para cancelamento de agendamento.

Em todos esses casos, para o cliente prosseguir com o cancelamento, o mesmo deve resolver um teste do tipo CAPTCHA conforme ilustrado pela figura 4.17. Uma vez que o agendamento seja cancelado com sucesso, o cliente recebe a resposta ilustrada na figura 4.18.

Figura 4.18: Página da interface-cliente do sistema com resposta ao pedido de cancelamento de agendamento.

4.7.6 Uso do telão

A interface-cliente possibilita que monitores ou televisões sejam utilizados de maneira a fornecer auxílio aos clientes durante sua entrada nas unidades. Embora essa funcionalidade não tenha sido mapeada nos casos de uso, a mesma representa uma adaptação no sistema que resulta na dispersão de aglomerações, principalmente quando os clientes chegam à unidade alguns minutos antes de seu horário.

A figura 4.19 ilustra como a interface-cliente disponibiliza às informações aos clientes. Em [41], Vilardo apresenta uma fotografia com a interface-cliente em um telão durante um atendimento.

4.8 Interface de Administração

A interface-administrativa é a responsável por permitir que os operadores manipulem o sistema, cadastrando atendimentos e verificando agendamentos, principalmente. É nessa interface que todas as atividades administrativas são realizadas e portanto a mesma deve ser uma ambiente de acesso restrito e seguro.

O acesso a essa interface ocorre via protocolo HTTP com criptografia através

Figura 4.19: Página da interface-cliente do sistema preparada para uso nos telões informativos.

do endereço https://admin.ru.ct.ufrj.br.

4.8.1 Autenticação no sistema

O fluxo de autenticação no sistema é iniciado por um formulário que solicita ao operador suas credenciais e a resolução de um teste do tipo CAPTCHA. A figura 4.20 ilustra o formulário onde o operador deve inserir suas credenciais e resolver o teste do tipo CAPTCHA.

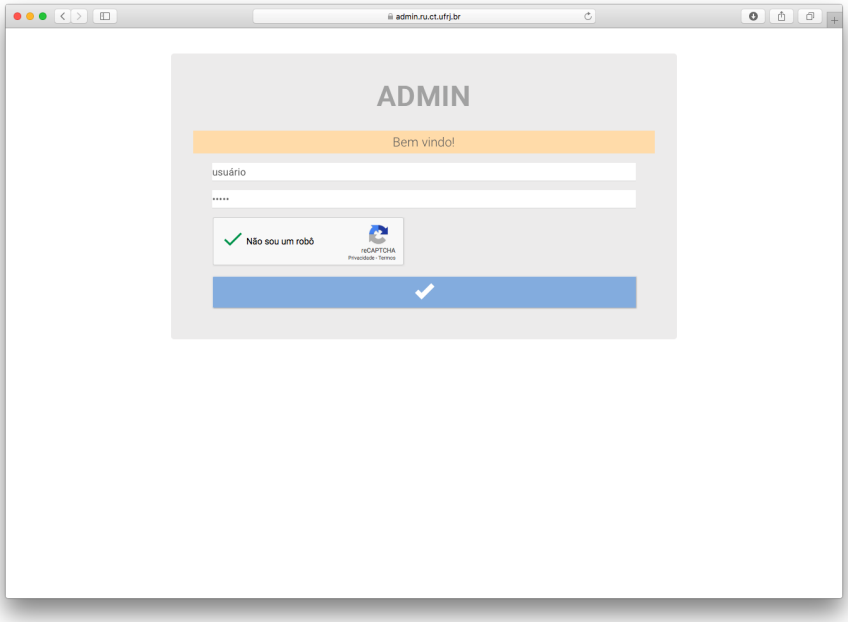

Figura 4.20: Página da interface-administrativa do sistema com formulário para autenticação.

Uma vez corretas e validadas as credenciais do usuário na API, a interface-administrativa exige a autenticação multi-fator utilizando o sistema Duo Mobile. Nesse momento, o operador deve selecionar o tipo de autenticação multi-fator (isso decorre do sistema Duo Mobile oferecer as possibilidades de *Push*, SMS e TOTP [42]). Conforme ilustrado na figura 4.21, temos um cenário onde o operador selecionou o envio de uma verificação do tipo *Push*. Nessa figura podemos observar a tela de confirmação que surge em seu dispositivo móvel para verificação.

Após a confirmação da autenticação via dispositivo móvel, o operador é direcionado para uma tela contendo o menu principal do sistema. Conforme ilustrado na figura 4.22, a nomenclatura utilizada nos menus da interface-administrativa difere da apresentada nesse trabalho. O motivo para essa divergência é que o uso

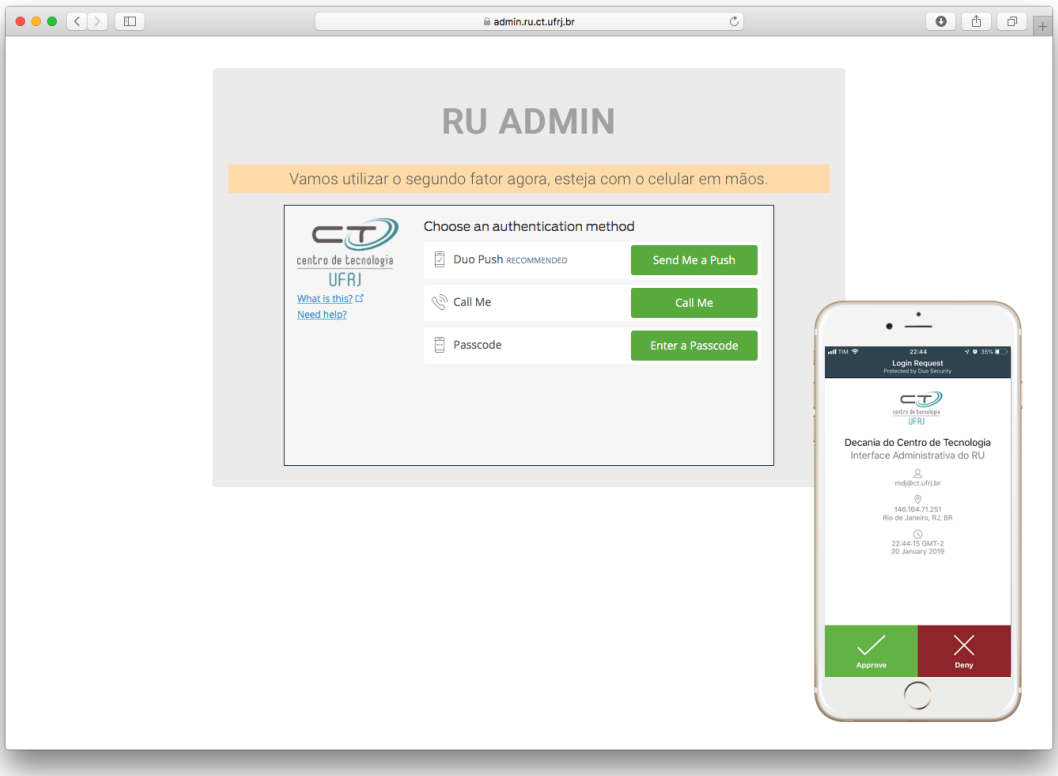

Figura 4.21: Página da interface-administrativa do sistema para autenticação multi-fator e tela de um dispositivo móvel para confirmação de autenticação, ambos com Duo Mobile.

cotidiano do sistema torna as entidades "atendimento"e "agendamento"facilmente confundíveis devido a proximidade entre os radicais e sufixos utilizados em ambas as palavras.

De forma a evitar possíveis erros operacionais, a interface-administrativa utiliza a palavra "fila"para se referir ao conceito de atendimento neste trabalho. De forma semelhante, a interface-administrativa faz uso da palavra *"ticket"* para se referir ao conceito de agendamento do presente texto.

4.8.2 Listagem de atendimentos

Uma vez autenticado, o operador para listar os agendamentos deve selecionar o menu "Filas". Caso o mesmo tenha autorização para essa funcionalidade, o mesmo verá a lista de atendimentos disponíveis conforme ilustrado na figura 4.23.

Nessa listagem de atendimentos, são mostrados todos os atendimentos que

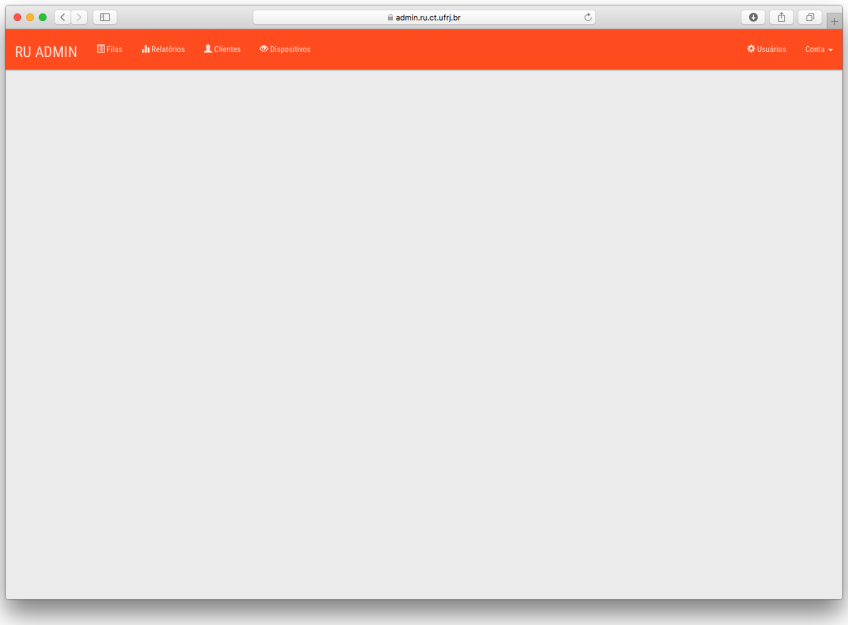

Figura 4.22: Página da interface-administrativa do sistema após autenticação realizada com sucesso.

não estejam com o estado fechado. Esse comportamento é diferente da interface-cliente (que somente exibe os atendimentos com estado disponível) pois as unidades podem preferir deixar os atendimentos com o estado privado, impossibilitando clientes de agendarem um horário.

Os atendimentos com o estado fechado podem ser obtidos através da opção do menu chamada "Relatórios", conforme mostrado na figura 4.24. Essa opção não apresenta diferença quanto ao permissionamento (para a API continua sendo uma operação de listagem), mas ela fica segregada na interface-administrativa pois pode ser utilizada para o levantamento histórico de informações de atendimentos passados.

4.8.3 Criação de atendimentos

Como cada ciclo de funcionamento de uma unidade é diferente de acordo com o tipo de refeição e horário, periodicamente novos atendimentos devem ser cadastrados no sistema. Para cadastrar um atendimento, o operador deve selecionar a opção "Cadastrar fila"na tela de listagem de atendimentos logo abaixo do menu do sistema (figura 4.23).

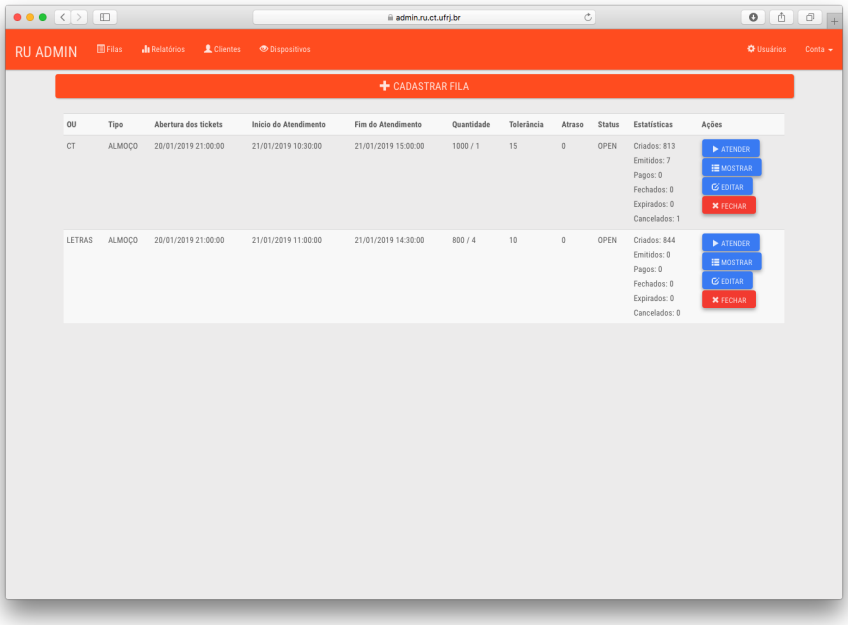

Figura 4.23: Página da interface-administrativa do sistema com listagem de atendimentos.

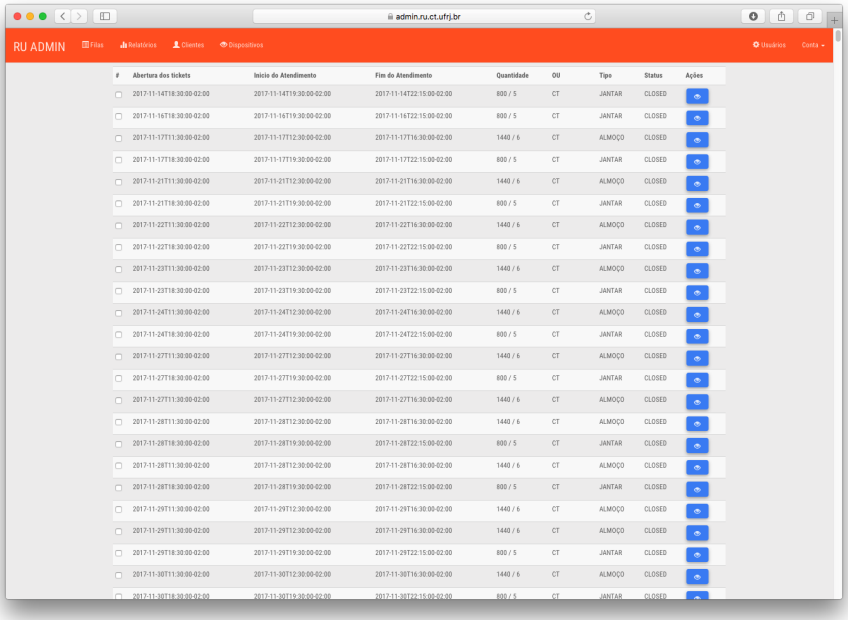

Figura 4.24: Página da interface-administrativa do sistema com listagem de atendimentos fechados.

Nessa tela (figura 4.25) o operador deve preencher as informações características do atendimento que o mesmo pretende abrir. Essas informações são relati-

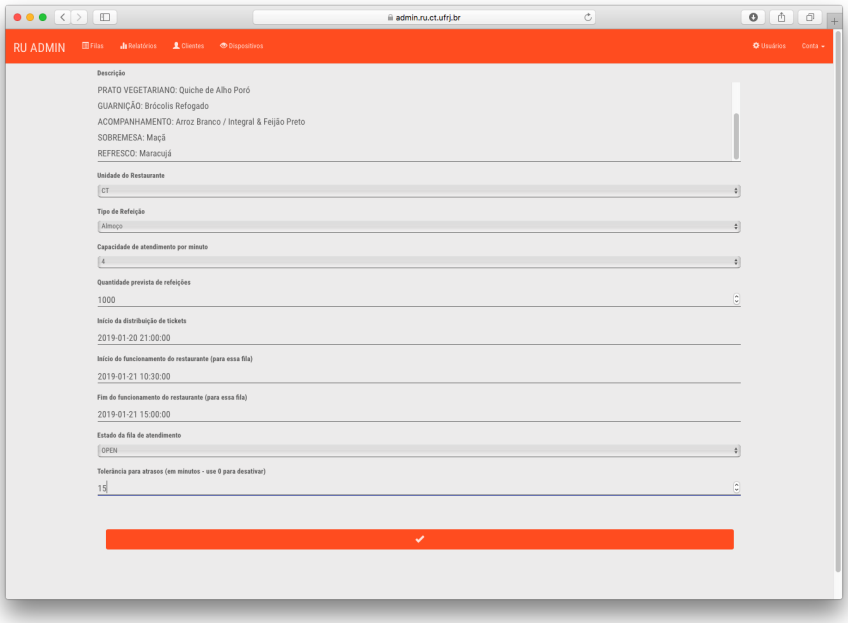

Figura 4.25: Página da interface-administrativa do sistema com formulário para criação de atendimento.

vas à descrição do atendimento, à unidade que servirá à refeição, ao tipo de refeição, à capacidade de atendimento por minuto (quantas pessoas por minuto a unidade consegue atender)[19], ao quantitativo total previsto de refeições, aos horários e à tolerância para aquele atendimento.

Quase todas essas informações podem ser editadas posteriormente, com exceção do horário de início e fim do atendimento na unidade.

4.8.4 Edição de atendimentos

No caso de um operador ter a necessidade de alterar as informações de um atendimento, a interface-administrativa fornece uma opção de edição. Essa opção está disponível através do botão "Editar"presente em cada atendimento conforme ilustrado na figura 4.23.

Não é possível editar os campos com os horários de início e fim do atendimento pois essas edições necessitam que agendamentos prévios sejam removidos ou

[19]Essa quantidade é configurável e normalmente varia de 4 à 10 clientes por minuto, de acordo com a quantidade de atendentes disponíveis na unidade em questão.

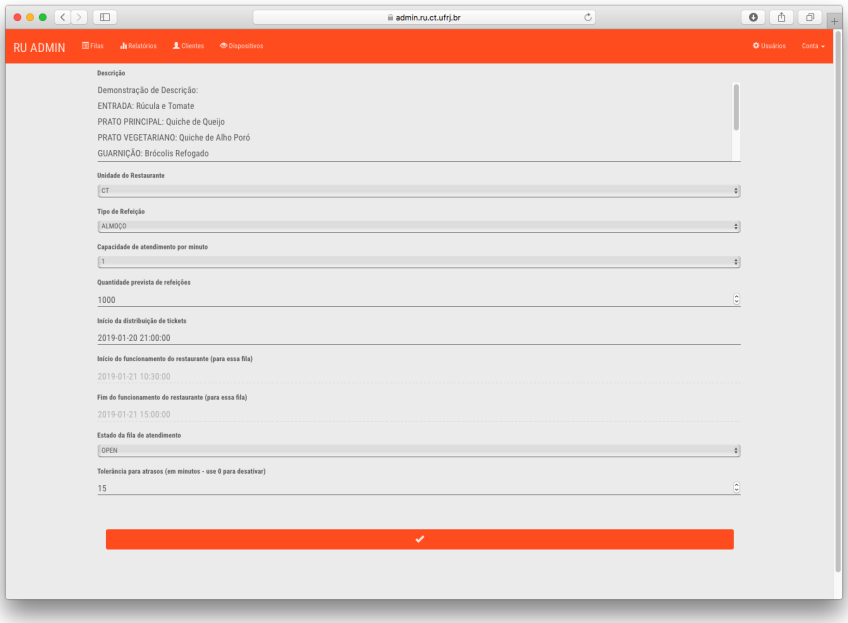

Figura 4.26: Página da interface-administrativa do sistema com edição de atendimento.

movidos, podendo causar estados inconsistentes no banco de dados[20].

Além da edição das características do atendimento, o estado de um atendimento pode ser alterado de duas formas. A primeira é utilizando-se do formulário mostrado na figura 4.26 e a segunda é utilizando-se do botão "Fechar" disponível ao se listar os atendimentos disponíveis (conforme ilustrado pela figura 4.23).

4.8.5 Atendimento de agendamentos

Com a proximidade do horário de início de um atendimento, o operador deve entrar no sistema para preparar o atendimento presencial dos clientes já agendados. A figura 4.27 ilustra a tela de atendimento da interface-administrativa.

Conforme ilustrado na figura 4.23 (p. 74), a interface-administrativa fornece duas opções de atendimento ao listar os atendimentos. A primeira opção, manifestada no botão "Atender" mostra apenas os atendimentos com horário próximo ao

[20] Um exemplo de estado de inconsistência é alterar o horário de início do atendimento com alguns agendamentos já realizados. Caso a data de início fosse adiada, não haveria como avisar aos clientes que marcaram um horário mais cedo. Assim, ao manter esses agendamentos na base, existiram agendamentos fora do horário do atendimento em questão, gerando inconsistência entre os agendamentos existentes e o atendimento cadastrado.

atual[21].

Outra opção disponibilizada pela interface-administrativa é manifestada no botão "Mostrar", conforme ilustrado na figura 4.23. Essa opção difere da opção "Atender"pois essa mostra todos os agendamentos cadastrados para o atendimento selecionado.

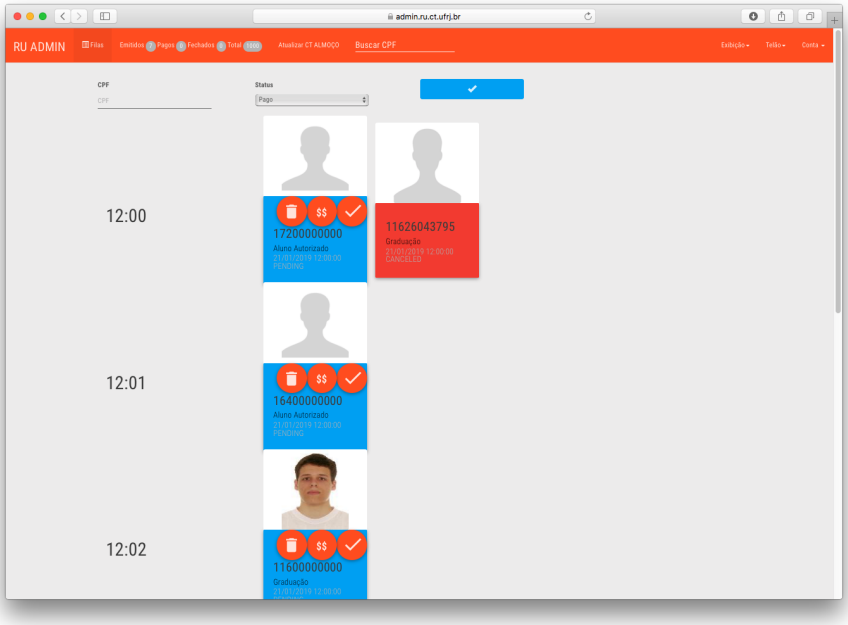

Figura 4.27: Página da interface-administrativa do sistema listando agendamentos de determinado atendimento.

Em ambas as opções, a tela mostrada pela interface-administrativa possui as mesmas características, variando somente o número de agendamentos exibidos. A figura 4.27 ilustra como os agendamentos são exibidos para um operador. As informações dos clientes são reduzidas ao mínimo possível para sua identificação visual (como a exibição da foto e do número do CPF). Além dessas informações, a classificação do cliente é exibida para auxiliar o atendimento local[22].

[21]Entende-se como horário próximo do atual os atendimentos agendados 10 minutos antes do horário atual e 5 minutos depois. Esse intervalo de tempo pode ser configurado através do menu "Exibição"disponível enquanto um atendimento estiver sendo realizado.

[22]Essa classificação se refere ao vínculo que o cliente possui com a Universidade. Por exemplo, "Graduação", "Mestrado"ou "Servidor"são valores possíveis para essa informação. Um caso especial ocorre quando o SIGA se encontra indisponível, onde essa informação passa a exibir o valor

Uma vez que um cliente dê entrada na unidade, sua foto deixa de ser exibida no sistema. O mesmo cenário ocorre caso o cliente cancele seu agendamento ou tenha o mesmo expirado em virtude de ter ultrapassado a tolerância configurada no atendimento.

Ainda nessa tela da interface-administrativa, cada agendamento possui três opções para manipulação de seu estado interno. Cada opção manifesta-se através de um botão circular localizado entre a foto do cliente e o seu CPF conforme ilustra a figura 4.27.

O primeiro botão, cujo símbolo é uma lixeira altera o estado do agendamento para "cancelado". O segundo botão com o símbolo de "$$" altera o estado do agendamento para "pago". Por fim, o terceiro botão com o símbolo de um "*check*" altera o estado do agendamento para o correspondente à "subsidiado".

Além de alterar o estado dos agendamentos, a tela ilustrada na figura 4.27 possibilita a configuração de atraso para o atendimento. Essa configuração é realizada através da opção "Atraso" disponível no menu. Adicionalmente, é possível realizar buscas pelos CPFs previamente agendados utilizando o campo de busca disponível no menu.

Finalmente, logo abaixo do menu existe um formulário simples para a entrada de um CPF e a configuração de um estado. Esse formulário cadastra novos agendamentos (conforme caso de uso 3.4.2.9, p. 29) e deve ser utilizado para cadastro de entrada de clientes quando uma unidade estiver utilizando o sistema apenas como mecanismo de controle de acesso, ou seja, sem a funcionalidade de agendamento[23].

4.8.6 Cadastro de clientes e operadores

Para os casos de uso referentes ao cadastro de operadores (3.4.2.13, 3.4.2.14, 3.4.2.15 e 3.4.2.16, p. 33) a interface-administrativa oferece uma tela para essas ações conforme ilustra a figura 4.28. Nessa tela é possível listar os operadores atualmente cadastrados, suas permissões e cadastrar novos operadores com o formulário presente logo abaixo do menu.

"Ver documentos"

[23] Esse cenário é utilizado na unidade Central e na unidade na Faculdade de Letras.

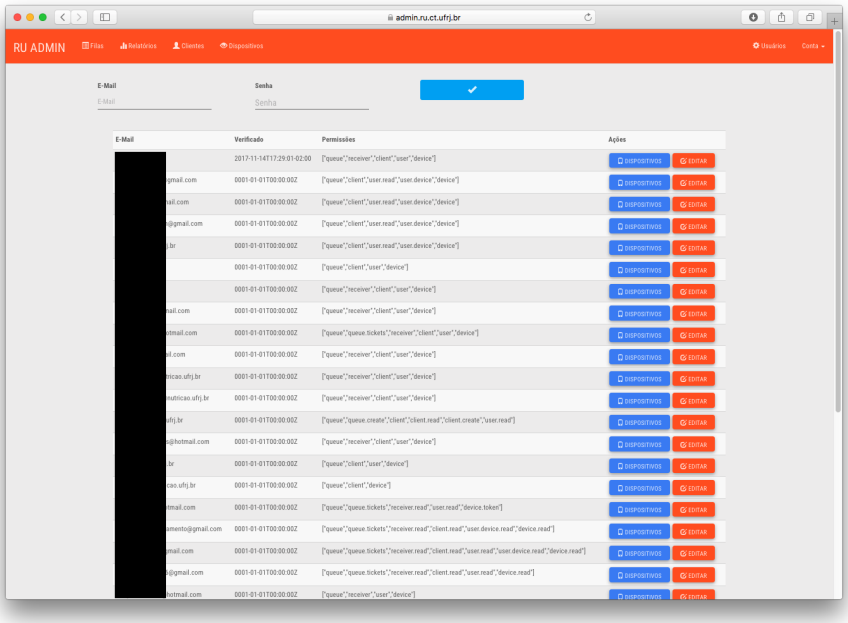

Figura 4.28: Página da interface-administrativa do sistema listando operadores cadastrados. Grifo realizado pelo autor para preservar os e-mails dos operadores.

Outras opções podem ser acessadas para verificar dispositivos e editar permissões e a senha de operadores[24].

No caso (3.4.2.12) onde um operador precisa cadastrar novos clientes a pedido da Universidade, a interface-administrativa fornece essa opção através do menu "Clientes". Conforme ilustra a figura 4.29, os clientes são listados e na mesma tela é possível cadastrar novos no formulário presente logo abaixo do menu.

Para todos os casos, os operadores devem ter permissão para realizar tais ações. Caso o operador não possua alguma permissão, ele receberá como resultado da interface-administrativa tela ilustrada na figura 4.30.

4.9 Documentação do sistema

Documentar um sistema distribuído ao longo de seu desenvolvimento mostrou-se um desafio para o autor do trabalho. Ferramentas tradicionais de modelagem como UML, se mostraram ineficientes em um cenário onde existia a necessidade de

[24] Essas outras telas não serão mostradas nesse trabalho pois estão fora do escopo das atividades principais deste capítulo.

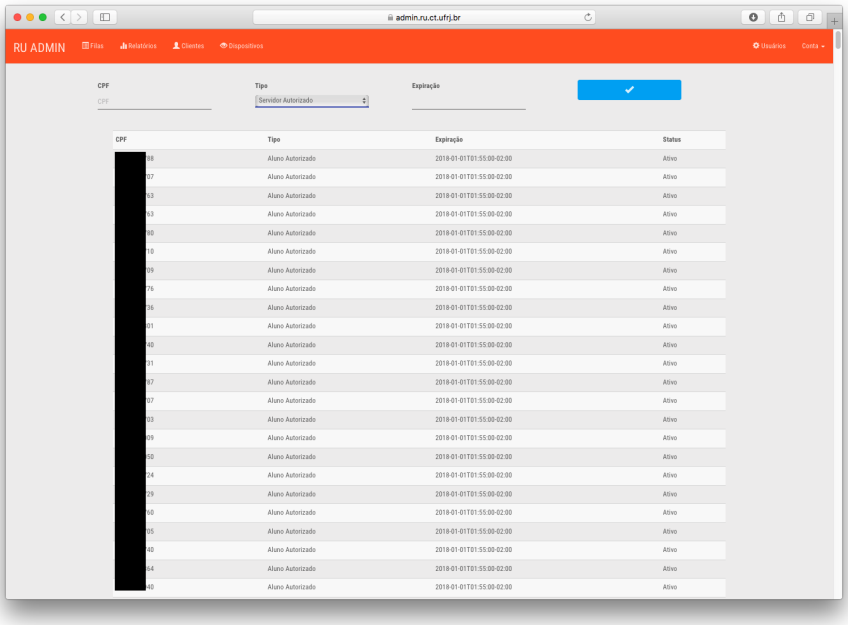

Figura 4.29: Página da interface-administrativa do sistema listando clientes cadastrados. Grifo realizado pelo autor para preservar os CPFs dos clientes cadastrados.

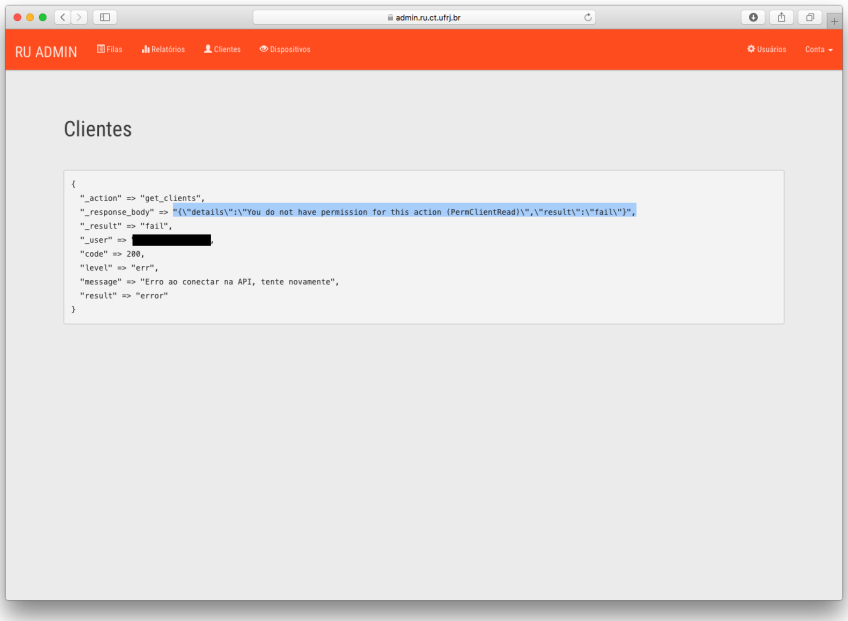

Figura 4.30: Página da interface-administrativa do sistema exibindo erro de permissão.

rápidas mudanças em virtude da necessidade de atender as demandas mais latentes do Sistema de Alimentação da Universidade.

Outro ponto desafiador, foi manter a documentação separada do código-fonte

do projeto, o que inviabilizou a utilização de uma série de ferramentas que extraem e geram a documentação automaticamente. Ainda assim, foi possível produzir uma documentação robusta e utilizável, conforme demonstra o trabalho de Felipe Oliveira [4] no desenvolvimento de aplicativos para dispositivos móveis. Trabalho este que se baseou apenas no uso da documentação de interoperabilidade existente (figura 4.31) disponível no endereço https://github.com/decania-ct/ct-docs-tickets/wiki.

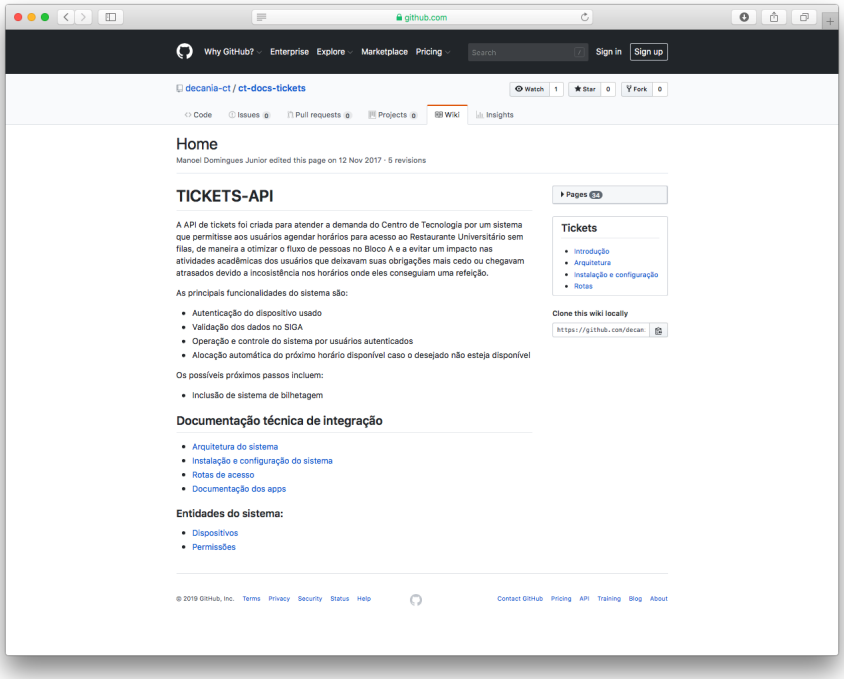

Figura 4.31: Página inicial da documentação de interoperabilidade do sistema.

Dessa forma, para que a documentação pudesse ser efetiva, além de ser de fácil acesso, a mesma deve possuir exemplos claros e reproduzíveis (conforme ilustra a figura 4.33). Para o propósito desse trabalho, toda a documentação de interoperabilidade foi construída e mantida em uma Wiki, com exemplos de funcionamento usando a ferramenta cURL [43].

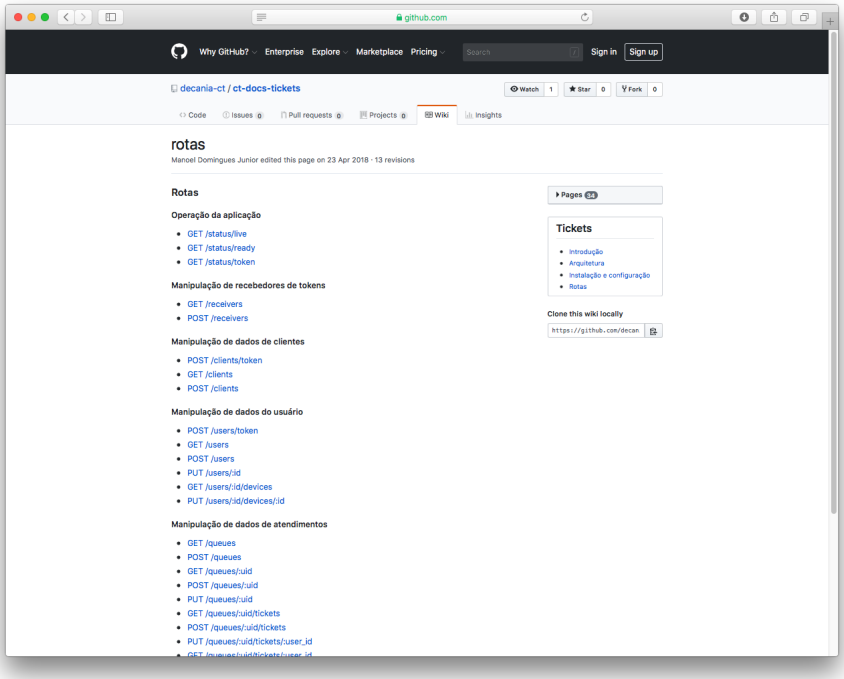

Figura 4.32: Página da documentação de interoperabilidade mostrando as rotas HTTP disponíveis no sistema.

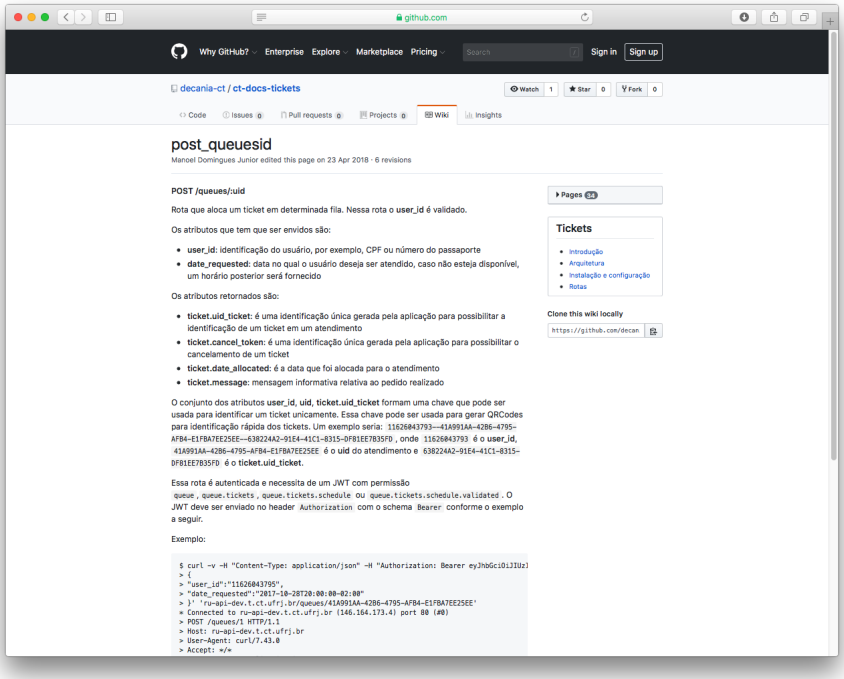

Figura 4.33: Página da documentação de interoperabilidade referente a rota POST /queues/:uid.

Capítulo 5

Impactos, conclusões e trabalhos futuros

5.1 Impacto na utilização do sistema

A universidade é um ambiente propício à troca de ideia e à formação de pessoas. Com o lançamento do sistema em 8 de setembro de 2016 [44], o impacto nas filas foi muito evidente. Com menos de uma semana de uso, a página da Internet da Seção Sindical dos Docentes da UFRJ fez uma matéria [41] com fotos mostrando a utilização do sistema. Nesse período, o sistema ainda estava em fase de testes e, mesmo assim, a diferença do tempo de espera na unidade do CT era perceptível, pois o restaurante passou a não ter mais filas.

A figura 5.1 ilustra um cenário de utilização cotidiana da unidade do Centro de Tecnologia antes da implementação do sistema. Pode-se observar uma longa fila, com início no centro-esquerdo da imagem e prosseguindo lateralmente até o canto inferior direito. Na figura 5.2, demonstra-se o impacto da utilização do sistema, onde somente é possível observar uma pequena aglomeração próximo do telão informativo de horário aos clientes, localizado próximo do centro da imagem.

5.2 Conclusão e trabalhos futuros

Neste trabalho foi desenvolvido um sistema composto por um conjunto de softwares na arquitetura de cliente-servidor com a finalidade de reduzir as filas de

Figura 5.1: Foto da fila do restaurante antes implantação do sistema de agendamento. Fonte: Rafael Cordeiro [45].

Figura 5.2: Foto da fila do restaurante após implantação do sistema de agendamento. Fonte: Isadora Vilardo [41].

acesso ao Restaurante Universitário do CT.

O que poderia ser apenas mais um trabalho ou estágio dentro da Universidade

se transformou em uma excelente oportunidade de resolver um problema real e que afetava a vida de milhares de estudantes da Universidade. Logo, quando a Decania do Centro de Tecnologia me convidou para ajudar a resolver "o problema do RU" não tive como recursar tamanho desafio.

Todos os principais pontos referentes ao sistema utilizado para agendamento e controle de acesso foram apresentados, assim como seus desafios técnicos, buscando alinhar a facilidade de uso com requisitos robustos de segurança de sistemas. As funcionalidades demandadas pelo Sistema de Alimentação da Universidade foram atendidas assim como demonstradas no presente trabalho.

A maior dificuldade enfrentada no projeto foi a instabilidade do ambiente de rede da Universidade. Este problema, somente foi contornado com a utilização de uma infraestrutura em nuvem paralela atuando apenas quando existe indisponibilidade dos recursos principais oferecidos pela Universidade para utilização do sistema. No cenário de indisponibilidade, os recursos da nuvem em paralelo fornecem capacidade computacional para o funcionamento do sistema.

Algumas funcionalidades úteis a serem incluídas no sistema, incluem a possibilidade de uma autenticação forte por parte do cliente, integração com a Intranet da UFRJ e o envio de alertas de horário via notificações e *chat-bots*.

Em um cenário distante, pode-se projetar a utilização do sistema como mecanismo de agendamento não somente para refeições, mas para qualquer outra atividade de atendimento na Universidade. Dessa forma contribuindo para uma maior organização e eficiência dos serviços administrativos da instituição.

Referências Bibliográficas

[1] ANDRADE, L. P. D., "Diretrizes do Sistema de Alimentação", http://ru.ufrj.br/index.php/2014-07-24-00-51-12/diretriz, Accessed: 2018-6-18.

[2] Universidade Federal do Rio de Janeiro, *Plano de Desenvolvimento Institucional 2012 a 2023: informações institucionais*, Jun. 2018.

[3] SOUZA, F., "tsuru: the open source PaaS", https://pt.slideshare.net/franciscosouza/tsuru-the-open-source-paas, 20 Jul. 2014.

[4] OLIVEIRA, F. P., *Desenvolvimento de Aplicativos Nativos Android e IOS para o Restaurante Universitário da UFRJ*. M.Sc. dissertation, Universidade Federal do Rio de Janeiro, Sep. 2018.

[5] DA UFRJ, G. D. R., DE COMUNICAÇÃO SOCIAL, E. D. P. D. S. S. G., DE PRODU, V. E., *et al.*, *Plano Diretor UFRJ 2020*, Report, Universidade Federal do Rio de Janeiro, Apr. 2011.

[6] SINTUFRJ, "Roleta livre da fome", *Jornal do SINTUFRJ*, v. 582, pp. 8, 2003.

[7] SOL, V., "Restaurante Universitário é inaugurado na Letras", https://ufrj.br/noticia/2015/10/22/restaurante-universit-rio-inaugurado-na-letras, 8 Nov. 2008, Accessed: 2018-9-9.

[8] SOL, V., "Restaurante Universitário Central é inaugurado", https://ufrj.br/noticia/2015/10/22/restaurante-universit-rio-central-inaugurado-0, 31 Mar. 2009, Accessed: 2018-9-9.

[9] CORONEL, M., "Comida com qualidade e horário noturno no campus na Ilha do Fundão", http://www.poli.ufrj.br/noticias/n805.html, 31 Jan. 2012, Accessed: 2018-9-9.

[10] Carolina Barreto Edição, "Campi do Centro e Praia Vermelha terão serviço de quentinha", https://ufrj.br/noticia/2016/07/08/campi-do-centro-e-praia-vermelha-terao-servico-de-quentinha, 7 Aug. 2016, Accessed: 2018-9-9.

[11] NOGUEIRA, D. M. S., "Campus Xerém - UFRJ - Restaurante Universitário do Campus UFRJ Duque de Caxias", http://caxias.ufrj.br/index.php/783-restaurante-universitario-do-campus-ufrj-duque-de-caxias, 17 Aug. 2018, Accessed: 2018-9-9.

[12] ALVAREZ, L., "Nutrição - Sistema de Alimentação - Passo a passo para acessar o sistema de agendamento do CT", http://ru.ufrj.br/index.php/102-passo-a-passo-para-acessar-o-sistema-de-agendamento-do-ct, Accessed: 2019-1-22.

[13] "Atuação — Decania do CT", https://www.ct.ufrj.br/decania/atuacao, 10 Jan. 2014, Accessed: 2019-1-20.

[14] PEREIRA, P. G., "Corpo e Rebeldia: pesquisa cênica em processo". In: *Diálogos e Dinâmicas*, v. 2, p. 329, 2009.

[15] AZEVEDO, S., "Manifesto do Juntos-UFRJ: construir o movimento estudantil e continuar a Primavera Carioca!", https://juntos.org.br/2013/05/manifesto-do-juntos-ufrj-construir-a-oposicao-de-esquerda-da-une-e-continuar-a 5 May 2013, Accessed: 2018-9-9.

[16] MELLO, A., ROPPA, A., DE OLIVEIRA, C., *et al.*, "Quem é você na fila do RU?", https://www.youtube.com/watch?v=hOHZfxBSrI4, 2017.

[17] PFLEEGER, S. L., *Engenharia de software: teoria e prática*. Prentice Hall, 2004.

[18] RICHARDSON, C., "Pattern: Microservice Architecture", http://microservices.io/patterns/microservices, 2017, Accessed: 2017-2-12.

[19] BARTH, A., *HTTP state management mechanism*, Report, 2011.

[20] VAN TILBORG, H. C. A., *Encyclopedia of Cryptography and Security*, v. 1. Berlin, Heidelberg, Springer-Verlag, 2005.

[21] DOMINGUES JUNIOR, M., "Segurança em APIs externas de uso mobile", Grupo de Trabalho em Segurança, 26 May 2017.

[22] "Protecting Applications", https://duo.com/docs/protecting-applications, Accessed: 2018-8-15.

[23] JONES, M., BRADLEY, J., SAKIMURA, N., "RFC7519: JSON Web Token (JWT)", , 2015.

[24] SAKIMURA, N., BRADLEY, J., JONES, M., *et al.*, "Final: OpenID Connect Core 1.0 incorporating errata set 1", 8 Nov. 2014.

[25] PERCIVAL, C., JOSEFSSON, S., *The scrypt password-based key derivation function*, Report, 2016.

[26] "Keepalived for Linux", http://www.keepalived.org/, Accessed: 2019-1-30.

[27] NADAS, S., "Virtual Router Redundancy Protocol (VRRP) Version 3 for IPv4 and IPv6", , Mar. 2010.

[28] FIELDING, R., GETTYS, J., MOGUL, J., *et al.*, *Hypertext transfer protocol–HTTP/1.1*, Report, 1999.

[29] RENTEA, V., "The Art of Clean Code", JPrime, 30 May 2017.

[30] Globo.com, "Tsuru", https://tsuru.io/, Accessed: 2019-10-20.

[31] WIGGINS, A., "The Twelve-Factor App (traduzido)", https://12factor.net/pt_br/, Accessed: 2018-3-26.

[32] "dep · Dependency management for Go", https://golang.github.io/dep/, Accessed: 2019-1-20.

[33] MIYAGAWA, T., MIYAGAWA, "Carton", https://metacpan.org/pod/Carton, Accessed: 2019-1-20.

[34] FIELDING, R. T., *Architectural Styles and the Design of Network-based Software Architectures*. Ph.D. dissertation, University of California, 2000.

[35] RIENDEL, S., "Mojolicious - Perl real-time web framework", https://mojolicious.org/, Accessed: 2018-10-20.

[36] OTTO, M., THORNTON, J., Bootstrap contributors, "Bootstrap", https://getbootstrap.com/, Accessed: 2019-1-20.

[37] "MySQL :: Benchmarks", https://www.mysql.com/why-mysql/benchmarks/, Accessed: 2019-1-20.

[38] "CWE - CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition') (3.2)", https://cwe.mitre.org/data/definitions/362.html, Accessed: 2019-1-20.

[39] DOMINGUES JUNIOR, M., "Documentação do sistema do Restaurante Universitário da UFRJ", Wiki, 28 Oct. 2017.

[40] ISO/IEC, *Information technology - Automatic identification and data capture techniques - QR Code bar code symbology specification*, Report 18004:2015, Feb. 2015.

[41] VILARDO, I., "Fila virtual no bandejão do CT", https://adufrj.org.br/noticia/fila-virtual-no-bandejao-do-ct/, 20 Mar. 2017, Accessed: 2018-4-11.

[42] RYDELL, J., PEI, M., MACHANI, S., *TOTP: Time-Based One-Time Password Algorithm*, Report, May 2011.

[43] "curl", https://curl.haxx.se/, Accessed: 2019-1-21.

[44] XIMENES, N., "Restaurante Universitário do CT: agendamento online — Decania do CT", https://www.ct.ufrj.br/comunicacao/news/restaurante-universitario-do-ct-agendamento-online, 4 Apr. 2017, Accessed: 2019-1-10.

[45] CORDEIRO, R., "Restaurante Universitário CT", https://pt.foursquare.com/v/restaurante-universit%C3%A1rio-ct/4f4e5226e4b027c8742ba6c9?openPhotoId=4f8596a6e4b03a290f1d5ec0, 4 Nov. 2012.