Digitalização de Processos de Gestão de Espaços

Caio Cesar Hideo Nakai - 42882

Trabalho de Projeto apresentado à Escola Superior de Tecnologia e de Gestão de Bragança para obtenção do Grau de Mestre em Sistemas de Informação no âmbito da dupla diplomação com a Universidade Tecnológica Federal do Paraná.

Trabalho orientado por:

Prof. Doutora Maria João Varanda Pereira (IPB)

Prof. Doutor José Eduardo Moreira Fernandes (IPB)

Prof. Doutor Rafael Liberato Roberto (UTFPR)

Bragança

Novembro - 2020

Digitalização de Processos de Gestão de Espaços

Caio Cesar Hideo Nakai - 42882

Trabalho de Projeto apresentado à Escola Superior de Tecnologia e de Gestão de Bragança para obtenção do Grau de Mestre em Sistemas de Informação no âmbito da dupla diplomação com a Universidade Tecnológica Federal do Paraná.

Trabalho orientado por:

Prof. Doutora Maria João Varanda Pereira (IPB)

Prof. Doutor José Eduardo Moreira Fernandes (IPB)

Prof. Doutor Rafael Liberato Roberto (UTFPR)

Bragança

Novembro - 2020

Dedicatória

Dedico este trabalho aos utilizadores da ferramenta desenvolvida e espero que possa vos ajudar de alguma forma.

Agradecimentos

Agradeço a todos os envolvidos direta ou indiretamente para a realização deste trabalho. Família, amigos e professores, muito obrigado.

Resumo

A ESTiG faz adaptações em suas instalações no início de todos os anos letivos para atender as necessidades de cada ano. Manter as informações sobre a dimensão, ocupação, reservas, pessoas e equipamentos associados de cada espaço tem se tornado uma tarefa cada vez mais complicada.

Este trabalho tem como objetivo principal o desenvolvimento de uma ferramenta web baseada nas plantas da ESTiG, que permita manter atualizadas as informações sobre todos os espaços da escola e permita gerir as reservas de cada espaço. Além disso, pretende-se que as reservas para um espaço sejam feitas de forma coordenada com a aplicação dos horários/sumários.

Para desenvolver a ferramenta, foram utilizadas principalmente tecnologias como React, Node.js, MongoDB e Git. Entre as principais funcionalidades da ferramenta implementada estão: realizar pedidos de reserva, gerir espaços, gerir reservas, consultar horário de ocupação dos espaços e consultar informações dos espaços.

Após a implementação da ferramenta, foram realizadas várias reuniões a fim de testar principalmente, a usabilidade e o funcionamento correcto da ferramenta desenvolvida, com as pessoas responsáveis pela gestão das reservas e docentes. Logo após as primeiras reuniões de teste da ferramenta, foram sugeridas melhorias, que já foram implementadas.

Palavras-chave: gestão de espaços, desenvolvimento web.

Abstract

ESTiG makes adaptations to its facilities at the beginning of every school year to meet the needs of each year. Keeping information about the size, occupation, reservations, people and associated equipment of each space has become an increasingly complicated task.

This work has as main objective the development of a web tool based on the ESTiG floor plans, which allows to keep updated information on all spaces of the school and allows to manage the reservations of each space. In addition, it is intended that reservations for a space are made in a coordinated manner with the horários/sumários application.

To develop the tool, mainly technologies like React, Node.js, MongoDB and Git were used. The main functionalities of the implemented tool are: making reservation requests, managing spaces, managing reservations, consulting the space occupation time and consulting space information.

After the implementation of the tool, several meetings were held in order to test mainly, the usability and the correct functioning of the developed tool, with the people responsible for the management of reserves and teachers. Right after the first test meetings of the tool, improvements were suggested, which have already been implemented.

Keywords: space management, web development

Conteúdo

Lista de Figuras

Abreviaturas

API Application Programming Interface. 3, 9, 26, 29–31

CRUD Create, Read, Update, Delete. 8, 24

CSS Cascading Style Sheets. 5, 6

ESTiG Escola Superior de Tecnologia e Gestão. 1, 2, 38, 39

HTML HyperText Markup Language. 5, 6, 24

HTTP Hypertext Transfer Protocol. 31, 35

HTTPS Hypertext Transfer Protocol Secure. 35

IP Internet Protocol. 34, 35

JSON JavaScript Object Notation. 8

JWT Json Web Token. 29

LDAP Lightweight Directory Access Protocol. 3, 9, 13, 34, 35, 38

LESS Leaner Style Sheets. 6

SASS Syntactically Awesome Style Sheets. 5, 6

SQL Structured Query Language. 8

TLS Transport Layer Security. 31, 35

VM Virtual Machine. 35

Capítulo 1

Introdução

1.1 Enquadramento

A Escola Superior de Tecnologia e Gestão (ESTiG) dispõe de instalações no Campus de Santa Apolónia com uma área global de 14 000 m^2, o edifício da ESTiG comporta 112 gabinetes de docentes, 1 auditório, 2 anfiteatros, 25 salas de aula, 5 salas de informática, 1 biblioteca e 20 laboratórios, que ocupam uma área global superior a 3000 m^2, para além de zonas de convívio e de apoio técnico/administrativo [1].

Dada a dimensão das instalações e as adaptações que ocorrem para suprir as necessidades de cada ano letivo em termos de espaços e equipamentos, torna-se evidente a dificuldade em manter as informações de todos os espaços atualizadas.

Atualmente, para realizar a reserva de um espaço, o funcionário precisa falar com a funcionária responsável pela gestão de espaços e ela por sua vez faz uma anotação em um caderno para fazer o controle dos espaços reservados. Em relação às informações dos espaços, atualmente, apenas o nome do espaço e o número de lugares são armazenados em uma plataforma *online* para efeitos de elaboração de horário.

1.2 Objetivos

O objetivo é aprimorar o processo atual de reserva dos espaços da ESTiG através de uma ferramenta *web* que possibilite a gestão dos espaços, dos pedidos de reserva e armazene informações como recursos e pessoas associadas a cada espaço.

1.3 Trabalhos relacionados

Em relação aos trabalhos relacionados, foram encontradas duas soluções comerciais parecidas, são elas: "touchONE" e "Skedway". O touchONE é um sistema de reserva de salas, locais de trabalho, hotéis, grandes instalações corporativas, universidades, etc. O sistema utiliza painéis de reserva interativos na frente de cada espaço, que indica a ocupação do espaço no momento e possui integração com outros sistemas como o office 365 e o G-Suite. Também é possível utilizar através de aplicações *mobile* e *web*. Além disso, o sitema também oferece funcionalidades como *check-in* através de QR *codes*, que encontram-se nos painéis interativos, relatar problemas no espaço, suporte para 17 idiomas diferentes e outras. Para comprar esta solução é necessário entrar em contacto com a empresa, o *website* não possui nenhuma informação em relação ao custo da solução.

A solução "Skedway" é uma plataforma para gestão de espaços e secretárias. A plataforma também possui *hardware* dedicado para utilização da plataforma, assim como disponibiliza aplicações *mobile* e *web*. Entre as funcionalidades estão: integrações com office 365 e o G-Suite, *check-in* através de QR *codes*, relatar problemas no espaço, calendário para encontrar a data para a reserva, etc. A solução é vendida através de assinaturas mensais e o valor varia de acordo com a quantidade de espaços e secretárias, existe um plano gratuito entretanto é limitado a apenas 3 espaços.

Apesar das duas soluções comerciais apresentadas serem utilizadas por grandes empresas como por exemplo, a Microsoft e possuírem diversas funcionalidades, foi decidido desenvolver um novo sistema de raiz feito à medida da ESTiG. Entre os objetivos do

sistema que foi desenvolvido estavam a integração com a Application Programming Interface (API) dos sumários, a navegação com base nas plantas da escola e a autenticação através do Lightweight Directory Access Protocol (LDAP). Essas três funcionalidades são muito específicas e não existem nas soluções comerciais apresentadas, além disso, desenvolver a própria solução permite com que sejam adicionadas funcionalidades consoantes a necessidade.

1.4 Plano de Trabalho

A Figura 1.1 a seguir é o cronograma do projeto com as tarefas de forma geral e o tempo de duração de cada uma.

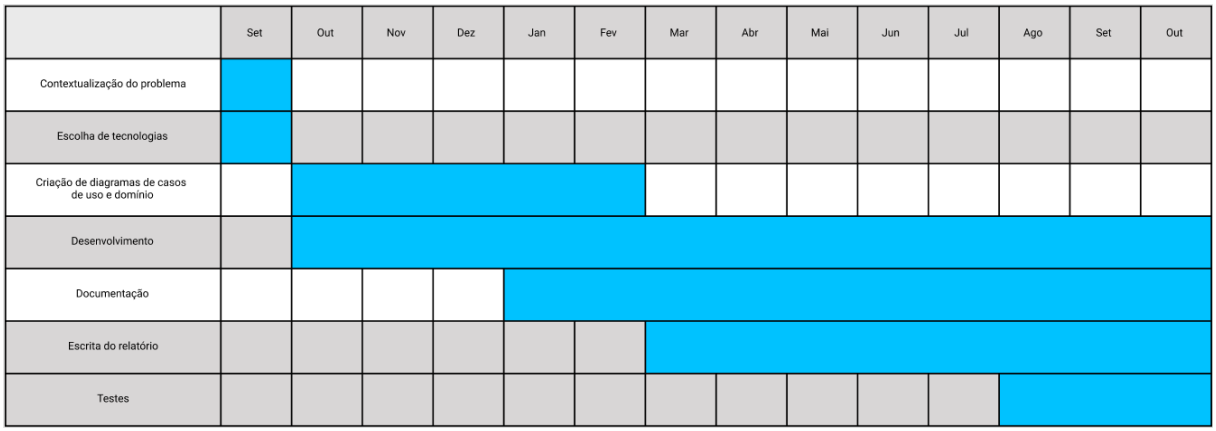

Figura 1.1: Cronograma do projeto.

1.5 Estrutura do Documento

O documento está estruturado de forma a contemplar todas as fases do desenvolvimento da solução implementada. No Capítulo 2 são abordadas as tecnologias utilizadas no desenvolvimento e as razões pelas quais foram escolhidas. No Capítulo 3 estão descritos a arquitetura do sistema, os diagramas de domínio e caso de uso e os atores do sistema. O Capítulo 4 aborda as questões relacionadas ao desenvolvimento, como os pontos mais

relevantes, algumas dificuldades encontradas e como foram resolvidas, os testes realizados e discussões. Por fim, no Capítulo 5 estão as conclusões, o conhecimento gerado e sugestões para trabalho futuro.

Capítulo 2

Contexto e Tecnologias/Ferramentas

Neste capítulo são apresentadas as ferramentas e tecnologias utilizadas para realização do trabalho. Tendo em vista que a proposta original do trabalho é desenvolver uma ferramenta *web*, as tecnologias apresentadas estão relacionadas ao desenvolvimento *web*. Em resumo, as tecnologias utilizadas para o *front-end* foram: HyperText Markup Language (HTML), JavaScript, Cascading Style Sheets (CSS), Syntactically Awesome Style Sheets (SASS) e React. Para o *back-end* foram utilizados: Node.js e JavaScript e para a base de dados foi utilizado o MongoDB. O controle de versionamento do código produzido foi feito utilizando o git com hospedagem na plataforma github.

2.1 HTML

O HTML é a principal linguagem de marcação da *World Wide Web* [2]. O papel do HTML é definir a estrutura da página, organizando os diferentes tipos de informação existentes em um documento HTML. Para que esta organização seja possível, existem as *tags*, que delimitam os diferentes tipos de informação presentes no documento. Por exemplo, no Código 2.1 temos a *tag* "p", que representa um parágrafo.

```
1    <p> paragraph example </p>
```

Código 2.1: Exemplo de *tag* HTML

2.2 JavaScript

O JavaScript é uma linguagem de programação leve, interpretada e orientada a objetos com *first-class functions* [3]. Pode ser utilizado no lado do cliente e no servidor, além disso, também pode ser utilizado para desenvolver aplicações *mobile*, isso faz com que o JavaScript seja amplamente utilizado, consequentemente faz com que a comunidade de utilizadores da linguagem seja grande, tornando a troca de informações e a solução dos problemas que possam surgir em relação a linguagem, mais fácil.

No entanto, apesar da popularidade e da ampla utilização, o JavaScript também possui seus problemas. Pelo fato dele seguir uma filosofia *"no crash"* e continuar a execução mesmo quando há comportamentos errados, isso faz com que erros possam ser facilmente ocultados e permaneçam até mesmo durante a produção [4].

2.3 CSS

O CSS é uma linguagem utilizada para adicionar estilo aos elementos de uma página *web*. Assim como o HTML, o CSS faz parte da base de criação de páginas *web* [5].

2.3.1 SASS

SASS é uma linguagem de folha de estilo que é compilada para CSS, o SASS possibilita a utilização de variáveis, regras aninhadas, funções matemáticas e outras funcionalidades, que permitem uma melhor organização para as folhas de estilo [6]. O SASS em comparação com o Leaner Style Sheets (LESS) outro pré-processador de CSS popular, não apresenta diferenças significativas, ambos apresentam funcionalidades semelhantes, a maior diferença entre eles é a sintaxe da linguagem. Independente do pré-processador escolhido, tanto o SASS quanto o LESS fazem com que código seja mais limpo, reusável e fácil de manter. Embora os pré-processadores apresentem várias vantagens, eles também possuem suas desvantagens como por exemplo, aumentar o tempo de compilação e tornar o *debugging* do código mais difícil.

2.4 React

O React é uma biblioteca JavaScript para criar interfaces de usuário, ele é declarativo, eficiente e baseado em componentes. Com ele é possível escrever códigos modulares, pois é possível separar o código em diferentes componentes, isso facilita a reutilização do código e deixa o código mais limpo. Além disso, o React possui uma comunidade grande e é utilizado por grandes empresas como Facebook, Instagram, Netflix, The New York Times e outras.

Comparando o React ao Angular e ao Vue, outros dois *frameworks* JavaScript populares, nenhum deles se destaca o suficiente para que possam ser recomendados para a maioria dos casos de uso em termos de desenvolvimento *web* [7]. A decisão para implementar a aplicação em React foi tomada devido ao fato de já existir um conhecimento prévio da biblioteca.

Entre as desvantagens do React estão: a necessidade de saber JSX, muitos exemplos da documentação utilizam classe, então também é praticamente necessário saber ECMAScript 2015 ou mais recente, o HTML fica junto ao JavaScript, o que pode prejudicar o entendimento e consequentemente ter uma curva de aprendizagem mais lenta.

2.5 Node.js

O Node.js é um ambiente que permite a execução do JavaScript fora dos navegadores de *internet*, utilizado para implementar o *back-end*. Ele é orientado a eventos e *single threaded*, utilizado para desenvolver aplicações rápidas e escaláveis. Entre as vantagens da utilização do Node.js estão: a comunidade de desenvolvedores ativa, o *node package manager* (a maior biblioteca de software do mundo) e a utilização do JavaScript tanto no *front-end* quanto no *back-end*, facilitando e acelerando o desenvolvimento.

A principal diferença entre o Node.js e outras alternativas como ASP.NET, Python, entre outras tecnologias, é o fato dele ser *single threaded*, enquanto as outras tecnologias são *multi threaded* e disponibilizam uma *thread* para cada utilizador. Portanto, não é

necessário preocupar-se com problemas de concorrência como *deadlock*, *race condition*, *starvation* e outros.

A desvantagem do Node.js está no baixo desempenho em realizar tarefas computacionais pesadas que demoram para serem executadas, como por exemplo, processamento de imagens, pois isto poderia acabar bloqueando o *event loop* (*thread* principal que permite operações não bloqueantes de entrada e saída), atrasando as outras ações que precisam ser executadas.

2.6 MongoDB

O MongoDB é uma base de dados não relacional, que armazena os dados semelhantes a um documento JavaScript Object Notation (JSON), possui escalonamento horizontal e apresenta maior flexibilidade quando comparado a base de dados Structured Query Language (SQL). Segundo [8], o desempenho do MongoDB é melhor quando comparado ao MySQL para operações Create, Read, Update, Delete (CRUD), utilizando um *schema* simples de dados.

A escolha de utilizar o MongoDB se deu principalmente pela eficiência no acesso aos dados, pelo ganho de flexibilidade para realizar alterações caso necessário, pois facilita a integração com outros sistemas e por tornar o desenvolvimento mais ágil devido a facilidade de aprendizagem.

2.7 Git

O git é um sistema de controle de versões *open source*, utilizado principalmente no desenvolvimento de *software*, com ele é possível realizar o versionamento de projetos pequenos ou grandes. Junto ao git foi utilizada a plataforma github, para fazer a hospedagem do projeto, atualmente o github é mantido pela empresa Microsoft e possui mais de 100 milhões de repositórios.

Capítulo 3

Abordagem/Análise/Modelação

Neste capítulo é apresentada a arquitetura escolhida, os diagramas utilizados, bem como uma descrição detalhada de todas as tarefas desempenhadas pelo sistema, os diferentes atores, seus papéis e permissões no sistema.

3.1 Arquitetura

A arquitetura monolítica mostrada na Figura 3.1 é a arquitetura do sistema desenvolvido. Foi escolhido esse tipo de arquitetura pois a quantidade de utilizadores não será demasiada e para aplicações com poucos utilizadores as arquiteturas monolíticas podem ser uma abordagem mais rápida para se começar [9]. Além disso, o desenvolvimento e *deploy* são mais simples, por existir apenas uma única base de código, existem menos problemas relacionados a integração e configuração [10].

Na Figura 3.1 podemos ver que o servidor se comunica com o servidor LDAP, pois é ele que realiza a autenticação dos utilizadores. Além disso, o servidor também comunica-se com a API Sumários para coletar os eventos e registrá-los no banco de dados.

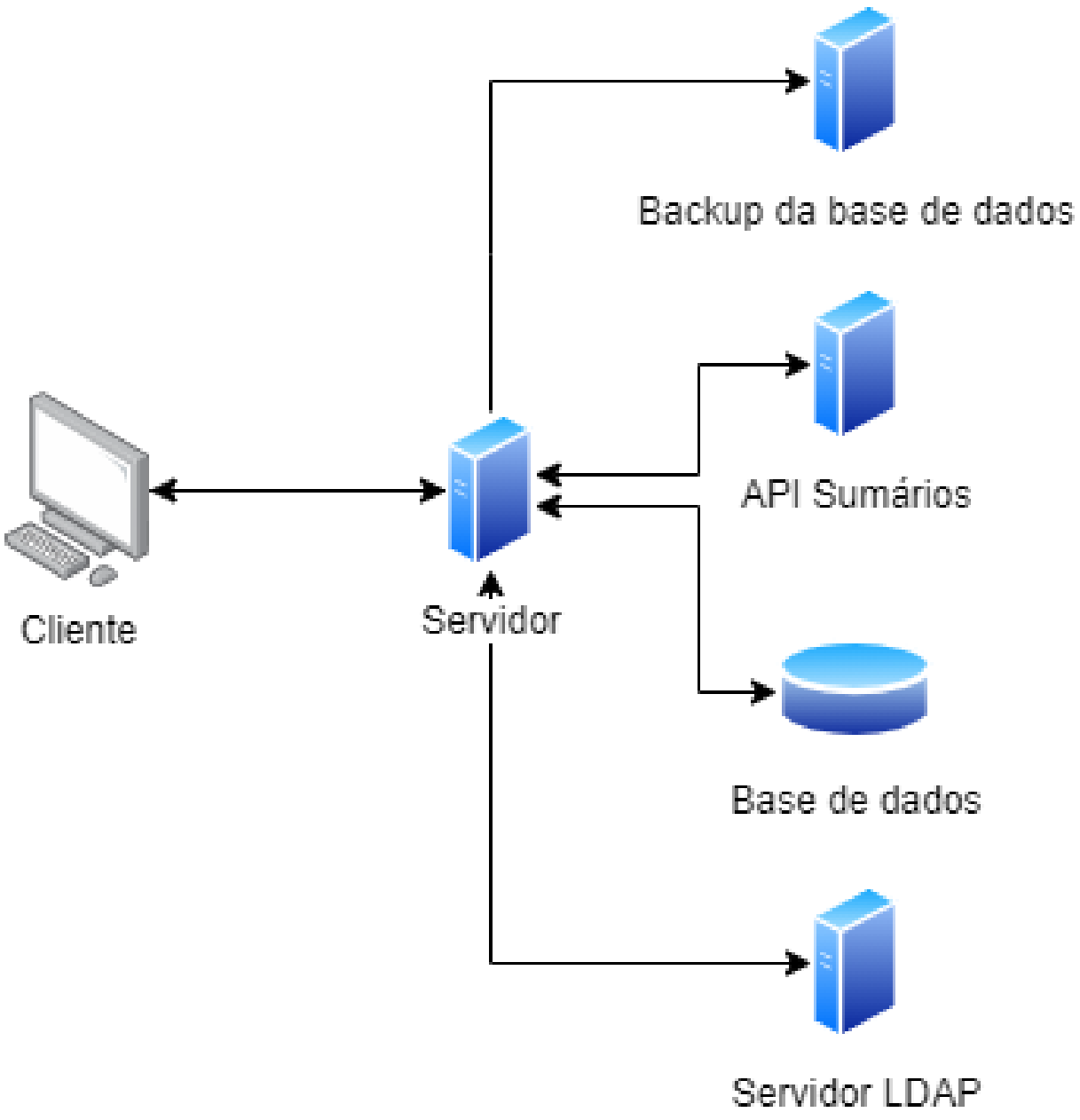

Figura 3.1: Arquitetura do Sistema.

3.2 Diagrama de Domínio

Com o intuito de fornecer uma visão geral das classes conceituais e suas relações foi criado o diagrama de domínio mostrado na Figura 3.2.

Para lidar com os diferentes pisos da escola, foi criada uma classe conceitual para o piso, possibilitando que caso haja a construção de um novo piso na escola, seja possível adicioná-lo ao sistema, tornando-o mais flexível. Também é possível remover/editar pisos. Cada piso pode possuir zero ou mais áreas. As áreas são diferentes regiões de um determinado piso e podem possuir zero ou mais espaços.

O espaço é a classe conceitual que possui a maior quantidade de relacionamentos. Cada espaço possui exatamente um tipo, por exemplo, tipo: "sala de aula", o espaço pode possuir zero ou vários recursos (recursos materiais), por exemplo, recurso: "ar condicionado". O espaço também possui relacionamentos com o funcionário, ele possui nenhum ou um funcionário responsável, pode possuir nenhum ou muitos funcionários técnicos e no mínimo um ou muitos gestores de espaço. Por fim, cada espaço pode possuir zero ou muitas reservas, cada reserva possui exatamente um motivo, por exemplo, motivo: "sala de estudo".

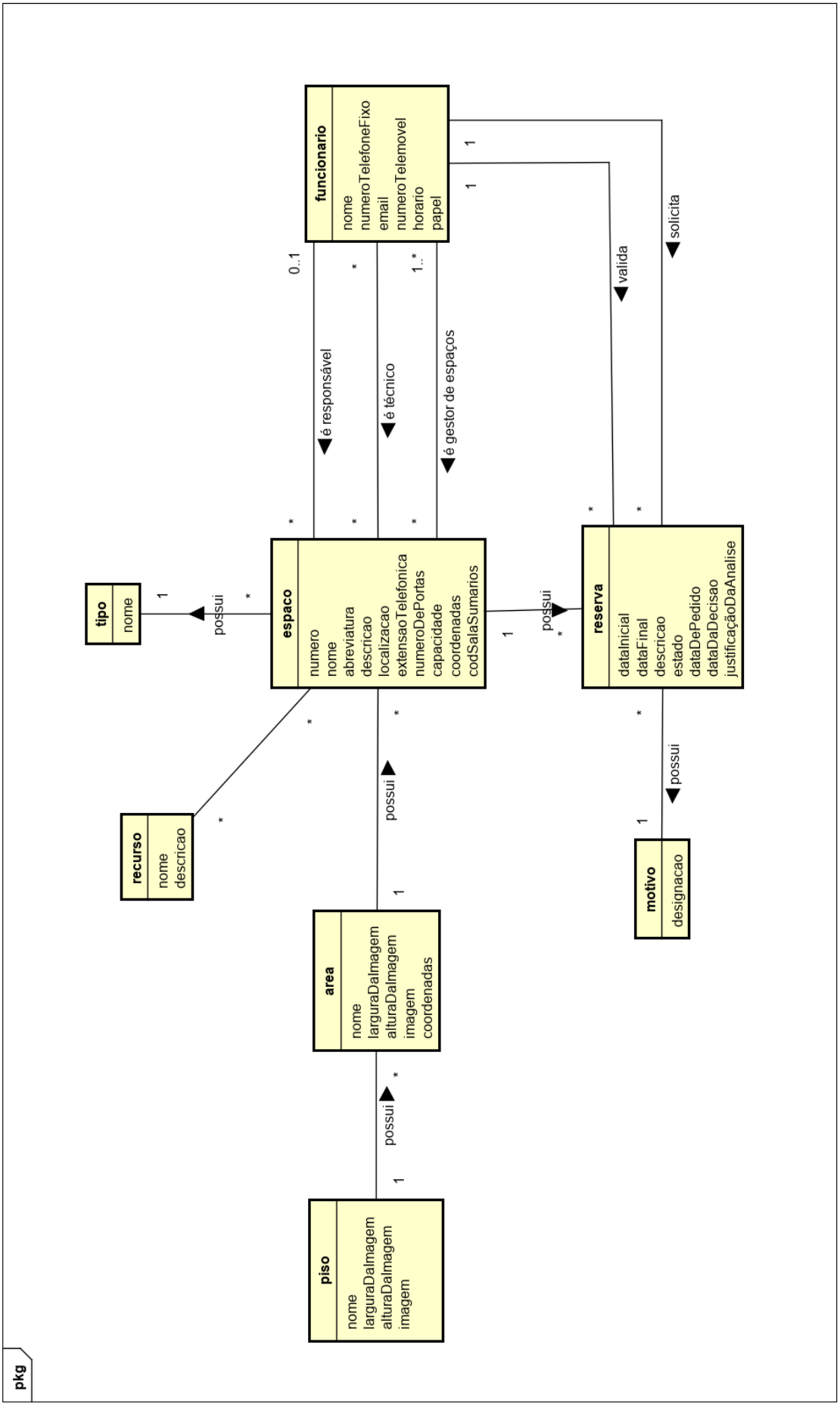

Figura 3.2: Diagrama de Domínio.

3.3 Diagrama de Casos de Uso

Nesta seção é apresentado o diagrama de casos de uso mostrado na Figura 3.3. Cada caso de uso existente na Figura 3.3 possui um texto explicatório com o nome, uma descrição curta, os atores envolvidos, os procedimentos padrões para realizar o caso de uso e procedimentos alternativos para realizá-lo caso houver.

O caso de uso UC1, "Realizar login" está associado a todos os atores. Permite que todos possam aceder ao sistema. Para realizar *login* o utilizador deve inserir o id de utilizador e senha, em seguida é verificado se os dados inseridos estão registrados na base de dados, ou no servidor LDAP. Caso os dados inseridos não estejam registrados o *login* não é realizado.

O caso de uso UC2,"Realizar logout" também está associado a todos os atores. Permite que todos possam sair do sistema. Para realizar *logout* o utilizador deve selecionar a ação para sair do sistema. Caso o utilizador não o faça, o *logout* é feito automaticamente 90 dias após o *login* ter sido realizado.

O caso de uso UC3, "Consultar informações dos espaços" também está associado a todos os atores. Permite que todos possam consultar as informações dos espaços. Entre as informações que podem ser consultadas estão: nome do espaço, número, abreviatura, extensão telefónica, localização, descrição, capacidade, número de portas, responsável, *email* do responsável, gestores e seus *emails*, técnicos e seus *emails* e recursos associados ao espaço, como por exemplo, projetores, ar condicionado, etc. Para consultar a informação de um espaço, o utilizador deve estar autenticado, isto é, ter feito o *login* e em seguida selecionar o espaço desejado navegando através das plantas da escola. Também é possível consultar a informação de um espaço através da procura dos espaços com filtros. Após fazer uma procura com filtros, se houver resultados, o utilizador deve selecionar um dos resultados para consultar as informações do espaço escolhido. A última forma de consultar as informações dos espaços é válida apenas para gestores do sistema e de espaços, a consulta é feita através da edição das informações de um espaço.

O caso de uso UC4, "Consultar estado da própria reserva" está associado ao funcionário. Permite que o funcionário possa consultar os estados das reservas que ele realizou. As reservas possuem quatro estados diferentes, sendo eles: "pendente", "aprovada", "recusada" e "cancelada". O estado de pendente significa que a reserva foi solicitada, no entanto nenhum gestor de espaço aprovou o pedido ainda. O estado de aprovada significa que um gestor já aprovou o pedido de reserva. O estado de recusada significa que um gestor recusou o pedido de reserva. O quarto estado, o estado de cancelada, significa que o funcionário que realizou a reserva a cancelou, antes ou depois do pedido ter sido aprovado ou recusado. As seguintes informações sobre a reserva podem ser consultadas: o solicitante (quem fez a reserva), o motivo da reserva, a descrição, a data, o período de início e fim e há quanto tempo a reserva foi realizada. Para as reservas que foram recusadas, existe também a informação da justificativa da análise, um texto opcional que o gestor de espaços pode escrever para justificar o motivo da reserva não ter sido aprovada. Para consultar o estado da própria reserva, o utilizador deve selecionar o estado da reserva que pretende-se consultar. Além disso, o funcionário pode consultar o estado das reservas após selecionar o espaço desejado e a data da reserva. Finalmente, a última forma de consultar o estado da própria reserva é através do *email*, sempre que o estado de uma reserva é alterado, um *email* é enviado pelo sistema para notificar o funcionário sobre a alteração.

O caso de uso UC5, "Consultar horário de ocupação dos espaços" está associado ao funcionário e ao gestor de espaços. Permite que ambos possam verificar se um espaço está ocupado para uma determinada data/horário. O utilizador (funcionário ou gestor de espaços), pode consultar o horário de 15 datas distintas simultaneamente. O intervalo de horários que pode ser consultado varia entre as 8:00 da manhã até as 24:00. Para consultar o horário de ocupação o utilizador seleciona um espaço navegando através das plantas da escola e escolhe uma ou mais data(s) em um calendário. Após escolher o espaço e as datas é exibida uma tabela de horários, contendo as reservas que estão pendentes ou já foram aprovadas de todos os utilizadores. Além do horário da reserva o utilizador também pode ver o nome da pessoa que a fez e o motivo. A segunda e última forma de

consultar o horário de ocupação de um espaço é através da procura de espaços com filtros. O utilizador pode procurar espaços através da disponibilidade pretendida, isto é, colocar as datas e horários de início e fim desejados e com isso verificar quais são os espaços que estão no resultado da procura.

O caso de uso UC6, "Solicitar reserva dos espaços" está associado apenas ao funcionário. Permite que o funcionário solicite a reserva de um espaço para uma data e horário específicos. Não é possível solicitar reservas para datas que já possuem uma reserva com o estado aprovada. Para realizar a solicitação de reserva o funcionário deve consultar o horário de ocupação dos espaços como mencionado no caso de uso acima. Em seguida o funcionário seleciona os horários e preenche um formulário, com o motivo da reserva. Os motivos são predefinidos, por exemplo, "aula", "defesas de tese", "reuniões", etc. Além do motivo, o funcionário pode preencher a descrição opcional que permite descrever informações adicionais que ele julgar necessário para o gestor de espaços. Também é possível fazer a solicitação através da procura dos espaços. Caso a disponibilidade pretendida seja especificada nos campos de procura, os resultados da procura permitem que seja feita a solicitação da reserva.

O caso de uso UC7, "Cancelar a própria solicitação de reserva dos espaços" está associado apenas ao funcionário. Permite que o funcionário cancele a própria reserva. Para cancelar, o funcionário deve selecionar o estado da reserva, pendente ou aprovada, pois não é possível cancelar reservas que foram recusadas ou reservas que já foram canceladas. Após selecionar o estado, o utilizador seleciona a reserva pretendida e realiza o cancelamento. O funcionário também pode cancelar a própria solicitação de reserva após selecionar o espaço para o qual a reserva foi realizada e selecionar a data da reserva.

O caso de uso UC8, "Gerir recursos" está associado apenas ao gestor de espaços. Permite que o gestor realize a gestão dos recursos, isto é, criar, listar, editar e remover um recurso. Cada recurso possui um nome e uma descrição e pode ser associados aos espaços. Para criar um recurso, o gestor deve preencher um formulário com o nome e descrição do recurso. Para listar todos os recursos existentes, o gestor seleciona a ação de listar recursos. Para editar um recurso, o gestor seleciona o recurso que pretende-se editar e

preenche novamente um formulário com o nome e a descrição. Para remover um recurso, o gestor novamente seleciona o recurso que pretende-se remover. Além disso, o gestor pode editar ou remover um recurso após listar todos os recursos existentes e selecionar o recurso que pretende-se editar/remover através da lista.

O caso de uso UC9, "Alterar o valor dos atributos dos espaços" está associado apenas ao gestor de espaços. Permite que o gestor realize alterações nos valores dos atributos de cada espaço, ou seja, pode-se alterar todos os atributos do espaço mostrados na Figura 3.2. Para alterar os valores dos atributos dos espaços, o gestor de espaços deve selecionar o piso do espaço e a área do espaço desejado e preencher um formulário que possui campos de todos os atributos do espaço.

O caso de uso UC10, "Editar solicitações de reservas dos espaços" está associado apenas ao gestor de espaços. Permite que o gestor altere algumas informações sobre reservas que possuem o estado pendente ou aprovada, não sendo possível editar solicitações que foram canceladas ou recusadas. Para o gestor de espaços alterar uma reserva também é necessário que esta reserva seja relativa aos espaços dos quais ele é gestor. As informações sobre a reserva que podem ser alteradas são: espaço para o qual a reserva foi solicitada, data da reserva, hora de início e hora de fim. Não é possível alterar o funcionário que solicitou a reserva, o motivo nem a descrição. Para editar as solicitações de reservas dos espaços o gestor de espaços deve selecionar a solicitação de reserva relativa aos espaços nos quais o gestor de espaços autenticado é gestor, isto é, não pode editar as solicitações de reserva de espaços que não estão associados ao gestor autenticado. Após selecionar a solicitação, o gestor seleciona a ação de editar e preenche um formulário para realizar a edição da solicitação. O gestor também pode editar uma solicitação de reserva após escolher o espaço e a data da solicitação, escolher a solicitação desejada, selecionar a ação de editar e preencher o formulário novamente.

O caso de uso UC11, "Rejeitar solicitações de reservas dos espaços" está associado apenas ao gestor de espaços. Permite que o gestor rejeite as solicitações de reserva dos espaços. Ao rejeitar uma solicitação pode-se escrever uma justificativa opcional por rejeitá-la, esta justificativa é enviada por *email* para o solicitante juntamente com a data da reserva, o

espaço solicitado, o estado e a justificativa se houver. Os procedimentos para rejeitar uma solicitação são basicamente os mesmos procedimentos do caso de uso anterior "Editar solicitações de reservas dos espaços". O gestor escolhe a solicitação que pretende-se rejeitar e seleciona ação de rejeitar, em seguida pode ou não preencher a justificativa. O outro procedimento é: o gestor escolhe o espaço, a data da solicitação e em seguida escolhe a solicitação desejada, a ação de de rejeitar e preenche ou não, a justificativa.

O caso de uso UC12, "Aceitar solicitações de reservas dos espaços" está associado apenas ao gestor de espaços. Permite que o gestor altere o estado de uma reserva pendente para aprovada. Não é possível aceitar solicitações canceladas ou que já foram rejeitadas, além disso, não é possível aceitar solicitações para datas que já possuem outra reserva com o estado aprovada. Após aprovar uma solicitação é enviado um email para o solicitante informando-o sobre a aprovação. Os procedimentos para aceitar uma solicitação, novamente são os mesmos dos casos de uso anteriores (editar e rejeitar solicitações), a única diferença é que para aceitar uma solicitação, não é necessário preencher nenhum formulário.

O caso de uso UC13, "Consultar informações de todas as solicitações pendentes de reservas" está associado apenas ao gestor de espaços. Permite que o gestor consulte as informações das solicitações para então decidir entre aprovar ou rejeitar ou editar a solicitação, além disso, para facilitar a tomada de decisão, o gestor pode filtrar as solicitações por solicitante, espaço e ordenar por solicitações mais antigas, mais recentes e por ordem alfabética dos nomes dos solicitantes. Para consultar as informações de todas as solicitações pendentes relativas aos espaços do qual ele é gestor, o gestor deve selecionar a ação para consultar as informações de todas as solicitações pendentes de reservas.

O caso de uso UC14, "Reservar espaço" está associado apenas ao gestor de espaços. Permite que o gestor faça a reserva dos espaços, isto é, o gestor não precisa solicitar a reserva e passar pelo processo de análise da solicitação. O gestor pode reservar todos os espaços até mesmo os espaços nos quais não é gestor. Os procedimentos para reservar um espaço são idênticos aos procedimentos do caso de uso "Solicitar reserva dos espaços".

O caso de uso UC15, "Exportar para excel" está associado apenas ao gestor de espaços.

Permite ao gestor fazer o *download* de ficheiros excel baseado em um intervalo de tempo. O ficheiro conterá informações sobre todas as reservas que possuem o estado "aprovada" e possuem datas entre o intervalo de tempo escolhido. As informações são: nome do espaço para o qual a reserva foi aprovada, nome do solicitante, motivo, data, horário de início e horário de fim da reserva. Para fazer o *download* do ficheiro excel, o gestor de espaços deve escolher um intervalo de tempo através da inserção de duas datas, a data de início e fim.

O caso de uso UC16, "Gerir tipos de espaço" está associado apenas ao gestor do sistema. Permite ao gestor criar, listar, editar e remover tipos de espaços. Para criar um tipo de espaço, o gestor seleciona a ação para criar e em seguida preenche o formulário com o nome do tipo de espaço. Para listar os tipos de espaço o gestor deve selecionar a ação para listar. Para editar um tipo de espaço, o gestor seleciona o tipo desejado e em seguida preenche o formulário com as alterações. Para remover o tipo de espaço, o gestor seleciona o tipo que pretende-se remover e em seguida confirma a ação para remover. Também é possível selecionar o tipo de espaço desejado após listar todos os tipos existentes.

Os procedimentos para listar, editar e remover espaços, pisos e áreas são basicamente os mesmos procedimentos explicados no caso de uso "Gerir tipos de espaço", portanto não serão explicados nos três casos de uso a seguir.

O caso de uso UC17, "Gerir espaços" está associado apenas ao gestor do sistema. Permite ao gestor criar, listar, editar e remover espaços. Para criar um espaço o gestor seleciona a ação para criar e em seguida preenche o formulário com todos os atributos do espaço mostrados na Figura 3.2 e em seguida confirma a ação.

O caso de uso UC18, "Gerir pisos" está associado apenas ao gestor do sistema. Permite ao gestor criar, listar, editar e remover pisos. A criação do piso exige um nome e a imagem do piso, devendo a imagem possuir uma das seguintes extensões: jpeg, jpg ou png. Para criar um piso, o gestor seleciona a ação para criar e em seguida preenche o formulário com o nome do piso e a imagem.

O caso de uso UC19, "Gerir áreas" está associado apenas ao gestor do sistema. Permite ao gestor criar, listar, editar e remover áreas. A criação da área exige um nome, o piso

onde a área pertencerá, a imagem da área e as coordenadas (mais sobre as coordenadas na subseção 4.1.1). Esta imagem também deve possuir uma das seguintes extensões: jpeg, jpg ou png. Para criar a área o gestor seleciona a ação para criar e em seguida preenche o formulário com o nome, piso, imagem e as coordenadas.

O caso de uso UC20, "Gerir permissões" está associado apenas ao gestor do sistema. Permite que o gestor altere o nível de permissão de um utilizador, por exemplo, fazer com que um funcionário passe a ser gestor de espaços. Não é possível alterar a permissão de um gestor de espaços caso algum dos espaços possuírem-no como gestor e não possuam nenhum outro gestor além dele, porque assim como indica o diagrama na Figura 3.2, um espaço está sempre associado a no mínimo um gestor de espaços.

3.4 Atores

Esta seção aborda os atores e cada um de seus papéis no sistema. O sistema possui três atores, sendo eles: funcionário, gestor de espaços e gestor do sistema. As ações que cada um pode realizar estão descritas nas três subseções seguintes.

3.4.1 Funcionário

O funcionário é como um utilizador comum, ele pode acessar as funcionalidades básicas do sistema, sendo elas: realizar *login* e *logout*; consultar informações dos espaços; consultar horário de ocupação dos espaços; solicitar reserva dos espaços; cancelar a própria solicitação de reserva dos espaços e verificar o estado das próprias reservas.

3.4.2 Gestor de Espaços

O gestor de espaços é o responsável por aprovar, rejeitar e editar as solicitações de reservas dos espaços; ele pode alterar o valor dos atributos dos espaços, por exemplo, alterar o nome de um espaço; pode gerir os recursos, isto é, criar, visualizar, editar e remover recursos; pode consultar o estado de todas as solicitações pendentes de reservas; consultar

o estado dos espaços; reservar espaços; e assim como o o funcionário, ele também pode realizar *login* e *logout* e consultar informação dos espaços.

3.4.3 Gestor do Sistema

O gestor do sistema, como o próprio nome já diz, é o responsável, de forma geral, por gerir o sistema, ele pode: inserir, remover, editar e visualizar os espaços, pisos, áreas e tipos de espaços; e novamente, assim como o o funcionário, ele também pode realizar *login* e *logout* e consultar informação dos espaços. Além disso, é ele que controla o perfil de gestor de espaços dos utilizadores e atribui a responsabilidade de gerir cada espaço.

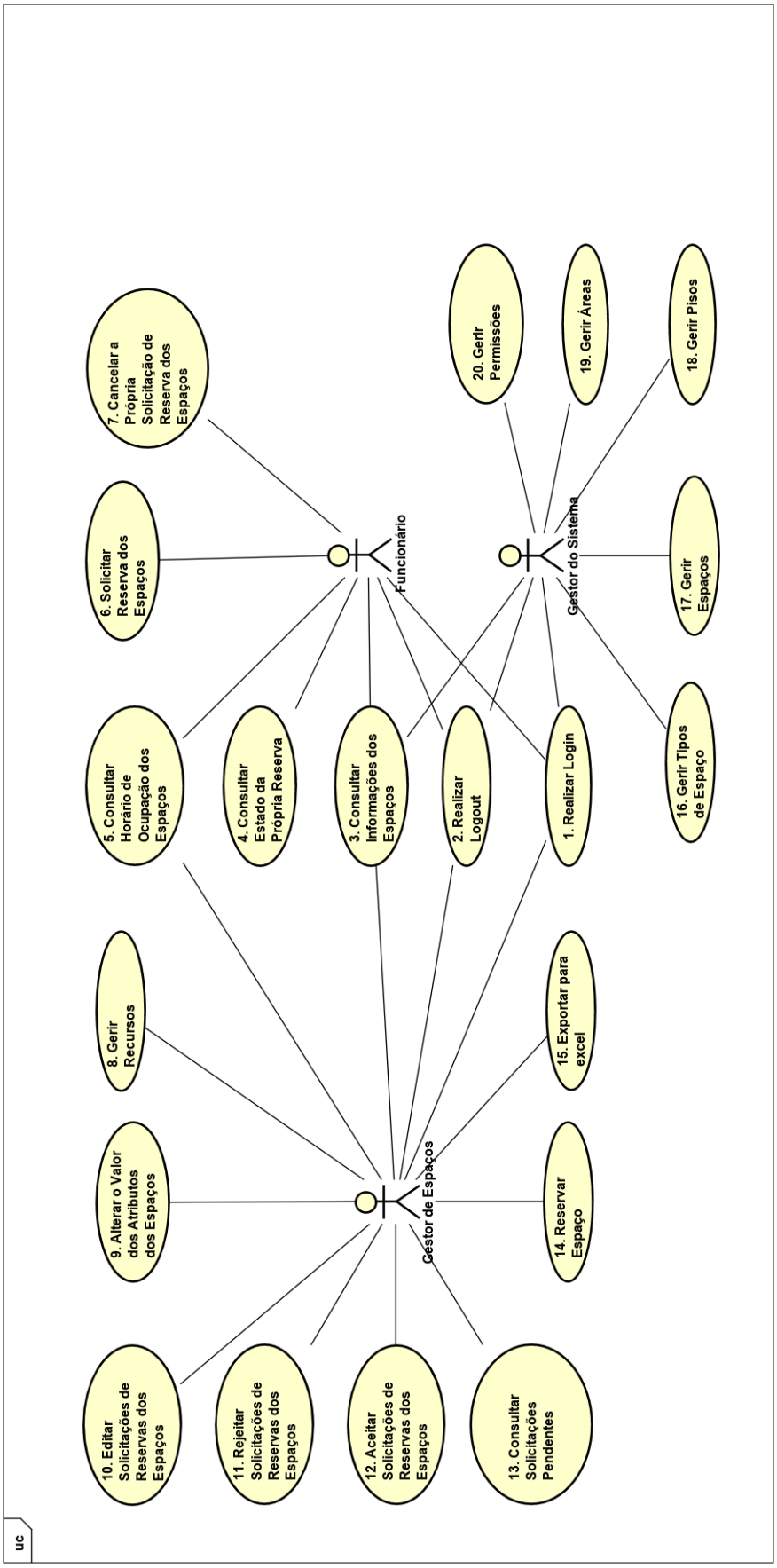

Figura 3.3: Diagrama de Casos de Uso.

Capítulo 4

Desenvolvimento/Implementação

Neste capítulo são apresentados os pontos mais relevantes, dificuldades encontradas e soluções técnicas aplicadas durante a fase de implementação do sistema que se baseia nas plantas reais da escola tendo uma interface gráfica que permite visualizar a disposição espacial dos vários espaços da escola.

4.1 Formas para realizar reservas de espaços

Existem duas formas para reservar ou solicitar a reserva de um espaço no sistema. Na primeira forma o utilizador seleciona o espaço navegando através das plantas da escola. Após selecionar o espaço é necessário escolher as datas no calendário, até quinze datas podem ser selecionadas. Depois das datas serem escolhidas o utilizador finalmente pode selecionar os horários na tabela de horários (mais detalhes sobre a tabela de horários na subseção 4.1.1) e solicitar/reservar um espaço.

A segunda forma consiste em procurar espaços utilizando filtros e incluir a disponibilidade pretendida no filtro. Caso algum espaço seja encontrado de acordo com os filtros e disponibilidade pretendida, o utilizador pode solicitar/reservar os espaços mostrados no resultado da busca.

4.1.1 Navegação por imagens selecionáveis

Entre os pontos a serem destacados está a implementação das imagens selecionáveis para navegar entre os diferentes espaços da escola. Um dos objetivos iniciais definidos na proposta para este trabalho é que a ferramenta fosse baseada nas plantas da escola. Portanto, para consultar as informações ou realizar a solicitação de reserva dos espaços o utilizador seleciona um piso e escolhe a área onde o espaço está localizado, como é mostrado na Figura 4.1. Após selecionar a área, ele seleciona o espaço desejado clicando em sua localização na planta da escola. Para auxiliar o utilizador a encontrar o espaço desejado, ao passar o rato por cima dos espaços/áreas o nome do espaço é mostrado ao lado. Além disso, a ferramenta também suporta as operações de CRUD para os pisos, áreas e espaços.

Para implementar as funcionalidades citadas acima, foi utilizada a *tag* HTML "path" para desenhar as áreas e espaços selecionáveis nas imagens, dessa forma ao definir o "path", as áreas selecionáveis na imagem, são sempre as mesmas independente do seu tamanho. No entanto, definir o *path*, ou seja, as áreas que serão selecionáveis na imagem, é um tanto quanto complexo para se fazer manualmente. Para isso são utilizados editores de imagens ou editores de *Scalable Vector Graphics*. Sendo assim, para o gestor do sistema adicionar novas áreas ou espaços é necessário que ele indique o *path*, ou seja, é necessário utilizar algum editor como dito anteriormente para obter este valor. Como isso não é uma tarefa trivial de ser feita pela primeira vez, foi adicionado um vídeo que explica como obter o *path* utilizando o editor de imagens *open source* GIMP.

A implementação da tabela de horários (mostrada na Figura 4.2) é outro ponto a ser destacado. Cada coluna da tabela representa uma data específica e as linhas representam um intervalo de horário de trinta minutos. As células da tabela podem ser selecionadas de três formas diferentes: ao clicar diretamente na célula, ao clicar no horário para selecionar toda a linha e ao clicar na data para selecionar toda a coluna. A tabela também exibe as reservas e solicitações de reservas, as quais são posicionadas nas linhas referentes aos horários que foram reservadas/solicitadas.

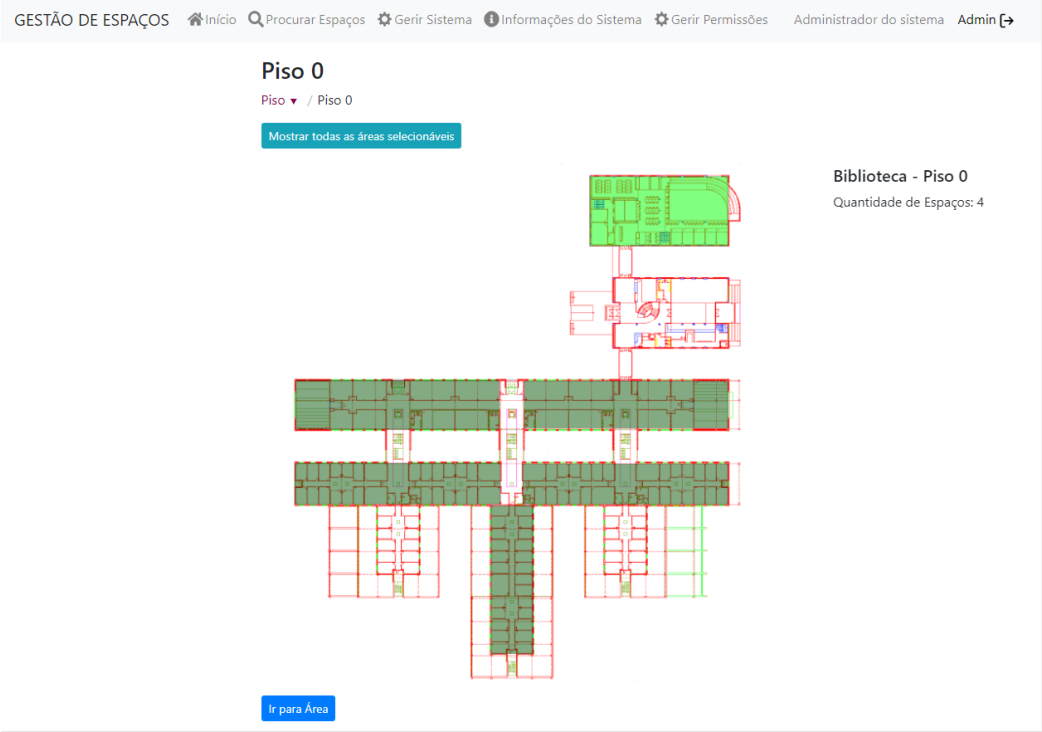

Figura 4.1: Navegação dos espaços baseada na planta da escola.

A tabela foi implementada do zero, isto é, não foram utilizadas bibliotecas de terceiros, pois nenhuma das bibliotecas pesquisadas possuíam as funcionalidades e *layout* esperados. Além disso, ao desenvolver os próprios componentes torna-se mais fácil acrescentar novas funcionalidades visto que não é necessário estudar o código desenvolvido por outra pessoa para verificar se é possível realizar as alterações desejadas.

4.1.2 Procurar espaços utilizando filtros

A funcionalidade de procurar espaços utilizando filtros como tipo de espaço, capacidade, número de portas, recursos, técnicos, responsável e disponibilidade de horários foi a funcionalidade com uma das *queries* mais complexas. A página de procurar espaços com filtros é mostrada na Figura 4.3. Apesar disso, foi possível implementar de forma muito simples, com expressões regulares e métodos JavaScript de objetos e *arrays*.

Exemplificando, caso o utilizador queira procurar um espaço com a capacidade maior

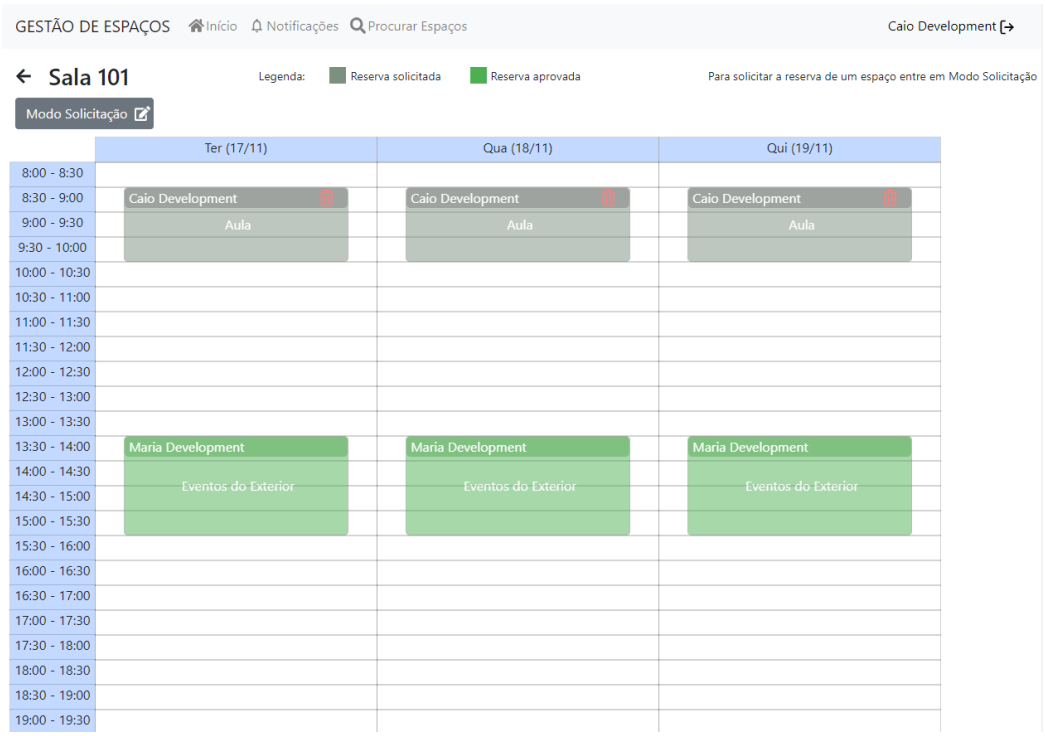

Figura 4.2: Tabela de horários.

ou igual a 40 que possua como recursos um quadro branco e um projetor e desconsiderar os outros filtros, é feita uma requisição do tipo *GET* para a API no seguinte formato: "https://api.com/spaces?capacity[gte]=40&resources=['idQuadroBranco', 'idProjetor']". A API por sua vez processará a requisição da seguinte forma: primeiro, transformará o objeto *query* em uma *string*, segundo, utilizará expressões regulares para formatar a *string* de forma que o MongoDB reconheça a query, por exemplo, o objeto query neste caso é recebido da seguinte forma: "{capacity: {gte: 40}, resources: ['idQuadroBranco', 'idProjetor'] }", no entanto, o formato esperado pelo MongoDB é "{capacity: { $gte: 40}, resources: { $all: ['idQuadroBranco', 'idProjetor']} }". Após formatar a *string* basta transformá-la em objeto novamente e utilizar o método *find* do MongoDB.

Dessa forma, a API consegue lidar com os diferentes tipos de operação como, maior, maior ou igual, menor, menor ou igual, exatamente igual de maneira simples e entendível.

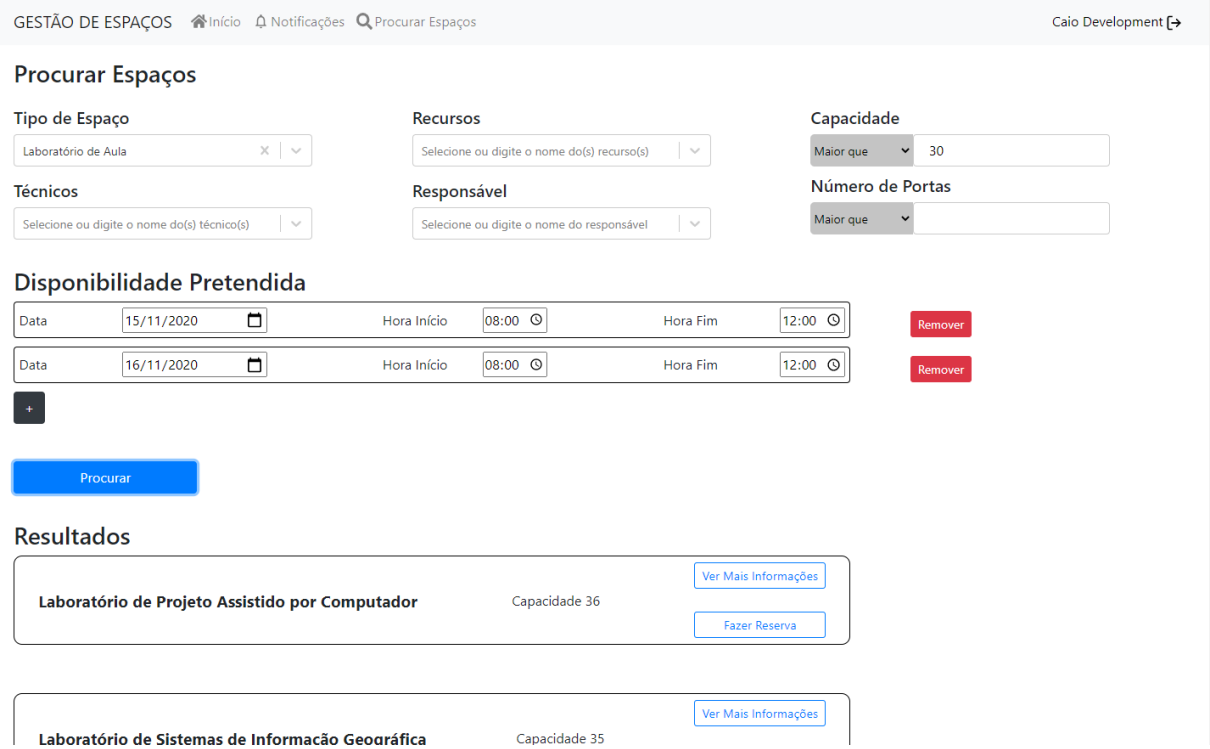

Figura 4.3: Página de procurar espaços com filtros.

4.2 Gestão de reservas

A implementação da gestão de solicitações de reservas também é um ponto interessante para destacar. A Figura 4.4 é a tela de gestão de reservas que pode ser acedida pelos gestores de espaço. Para fazer a implementação dos *"toasts"*, as "caixas" verdes mostradas na Figura 4.4, foi utilizado um componente da biblioteca React Bootstrap. Esta biblioteca também fornece vários outros componentes, como botões, *inputs*, formulários e outros, sendo que muitos deles também foram utilizados em outras partes da aplicação.

As solicitações de reserva podem possuir três cores diferentes, sendo elas: verde, significa que a notificação pode ser aceite e não possui conflito com nenhuma outra solicitação; amarela, significa que a notificação não pode ser aceite, pois há uma reserva em conflito e por último, vermelha, que significa que duas ou mais solicitações estão em conflitos, porém uma delas pode ser aceite. Além disso, também é possível filtrar as solicitações por solicitante e espaço assim como ordenar as solicitações pelas mais antigas, mais recentes

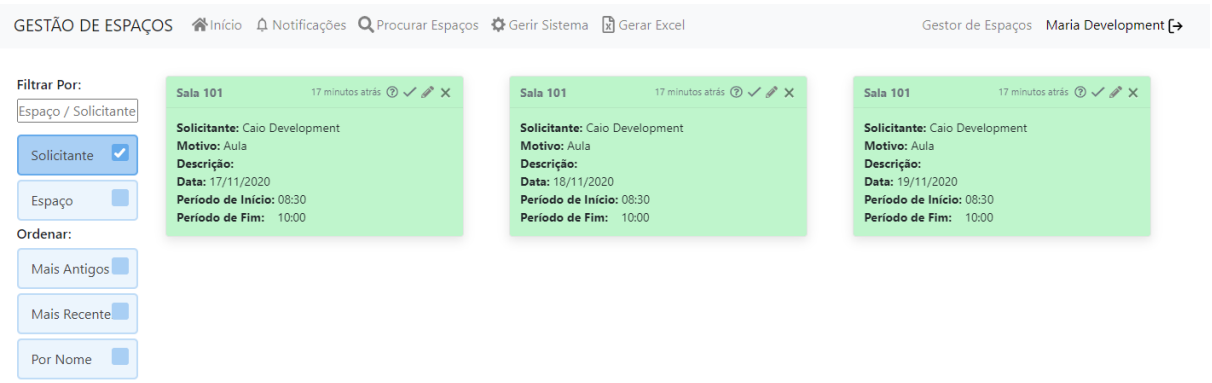

Figura 4.4: Tela de gestão de reservas.

e por nome.

Para implementar a funcionalidade das cores, primeiro foi necessário verificar quais solicitações estavam em conflito. Para fazer isso, como as datas são armazenadas como objetos JavaScript do tipo *date*, foram utilizadas as funções *get* para obter os valores das datas e uma lógica condicional para verificar se uma data está em conflito com a outra. Para realizar o filtro, como as solicitações são armazenadas em um *array* (JavaScript) de objetos, bastou utilizar o método *filter* para retornar apenas as solicitações que incluem o texto digitado e, para implementar a ordenação, também foi utilizado um método nativo do *array* chamado *sort* para ordenar as solicitações de acordo com o tipo de ordenação escolhido.

4.3 Integração com a aplicação dos Sumários

A aplicação dos Sumários é responsável por armazenar o horário das aulas e sua localização, sendo a comunicação é feita através de uma API Restful. Para importar os eventos dos Sumários primeiro foi necessário relacionar os espaços entre as diferentes aplicações, pois os sumários possuem os próprios espaços cadastrados na base de dados e o sistema desenvolvido também. Portanto para importar corretamente os eventos e associá-los ao mesmo espaço, os espaços no sistema desenvolvido possui o código do espaço que existe na aplicação dos Sumários. O utilizador ao criar um novo espaço no sistema, pode relacionar os espaços através do nome que o espaço possui na plataforma dos sumários. Dessa forma o utilizador não precisa saber o código do espaço, tornando a tarefa de criar novos espaços mais simples.

Após relacionar os espaços entre as diferentes aplicações, foi feita a importação dos eventos. Para isso, foi desenvolvida uma função para fazer requisições *GET* para cada espaço cadastrado no sistema com o intervalo de tempo de uma semana, a partir do momento em que foi feita a requisição. A função foi implementada para ser executada nos domingos as 23:00. Assim, a partir de segunda-feira de manhã, o sistema ficará pronto para serem analisados os pedidos de reserva que entretanto foram armazenados no sistema e cuja aprovação dependia da ocupação das salas para essa semana. Todos os pedidos feitos durante a restante semana podem ser analisados de imediato porque os horários apenas são alterados no fim de semana seguinte. Desta forma, consegue-se uma sincronização perfeita entre a exportação de novos horários e tratamento de novas reservas de espaços.

4.4 API

Para estabelecer a comunicação entre a base de dados e o *front-end* foi desenvolvida uma API Restful, ou seja, outras aplicações também podem aceder a API. Entretanto a maioria dos *endpoints* exigem um *token* de autenticação Json Web Token (JWT), podendo o *token*

ser obtido após fazer o *login*. Dessa forma é possível proteger o acesso das informações por pessoas não autorizadas.

Alguns *endpoints* para além da exigência do *token* de autenticação, também exigem um certo nível de permissão no sistema. Por exemplo, o *endpoint* para atualizar a informação de um utilizador, exige que o utilizador que fez a requisição além de estar autenticado seja também gestor do sistema. Esta funcionalidade foi implementada com o uso de funções *middleware*, estas funções possuem a capacidade de executar qualquer código, alterar os objetos de requisição e resposta, encerrar o ciclo de requisição/resposta e chamar a próxima função *middleware* que está na pilha.

No Código 4.1 é mostrado como é feita a definição do *endpoint* para atualizar o papel de um utilizador. No primeiro parâmetro é passado uma *string* com o caminho da rota, nos outros parâmetros passados são funções *middleware* que são executadas na ordem em que foram passadas como parâmetros. A primeira função passada é chamada "withAuth" que verifica se o utilizador possui um *token* válido. A segunda função chamada "restrictTo" atribui o nível de permissão necessário ao *endpoint*, ela pode receber várias *strings* como parâmetros. Essas *strings* representam os níveis de permissão que serão atribuídos ao *endpoint*, finalmente, a última função *middleware* é a função onde está contida a lógica para atualizar o papel do utilizador.

```
1  router.patch("/:employeeId", authController.withAuth, authController.
   restrictTo("system-manager"), employeeController.updateEmployeeRole);
```

Código 4.1: Exemplo de *endpoint* que utiliza funções *middleware*

Portanto, proteger os *endpoints* e atribuir diferentes níveis de permissão, torna-se uma tarefa muito simples, pois as funções *middleware*: "withAuth" e "restrictTo" podem ser reutilizadas para qualquer rota.

4.5 Backups

Em relação a base de dados, foram implementados *backups* automáticos que também são realizados aos domingos após a importação dos eventos da API dos sumários. Para gerar

os *backups* foi utilizado o comando "mongodump", que permite escolher a base de dados e também coleções específicas para gerar os *backups*. Os *backups* são armazenados na pasta "/backups". Esta pasta também possui *backups* automáticos para outro servidor para evitar que os dados estejam todos na mesma máquina caso ocorra algum problema.

4.6 Documentação

A documentação, apesar de muitas vezes esquecida, é uma das partes mais importantes de um projecto de *software*, pois facilita o entendimento do projecto e reduz drasticamente a quantidade de tempo necessário para as pessoas que não participaram do processo de desenvolvimento, realizem contribuições ou manutenções.

Para documentar a API foram criadas páginas *web* utilizando Sphinx e reStructuredText. O Sphinx é uma ferramenta que permite a criação rápida de páginas *web* para a escrita de documentação. O reStructuredText é a linguagem de marcação utilizada pelo Sphinx e possui uma sintaxe simples. A documentação possui todos os *endpoints* da API, o método Hypertext Transfer Protocol (HTTP) de cada um e uma pequena descrição sobre o que é esperado do *endpoint* e um exemplo de resposta. Além disso, a documentação também permite a pesquisa por *strings*. A Figura 4.5 mostra uma parte da documentação da API.

Também foram documentados os requisitos necessários para correr os projetos do *front-end* e do *back-end*. Para além dos requisitos necessários, também foi documentada a lista de todos os comandos necessários para correr ambos os projetos e outros comandos que possam ser úteis, como por exemplo, como aceder a linha de comando da base de dados, entre outros. Parte da documentação sobre o *back-end* é mostrada na Figura 4.6.

Além destas documentações, existe também foi documentado o processo de *deploy* das alterações realizadas no código e a localização dos projetos (*back-end* e *front-end*), da configuração do Nginx, da base de dados, dos certificados Transport Layer Security (TLS) e dos *backups*.

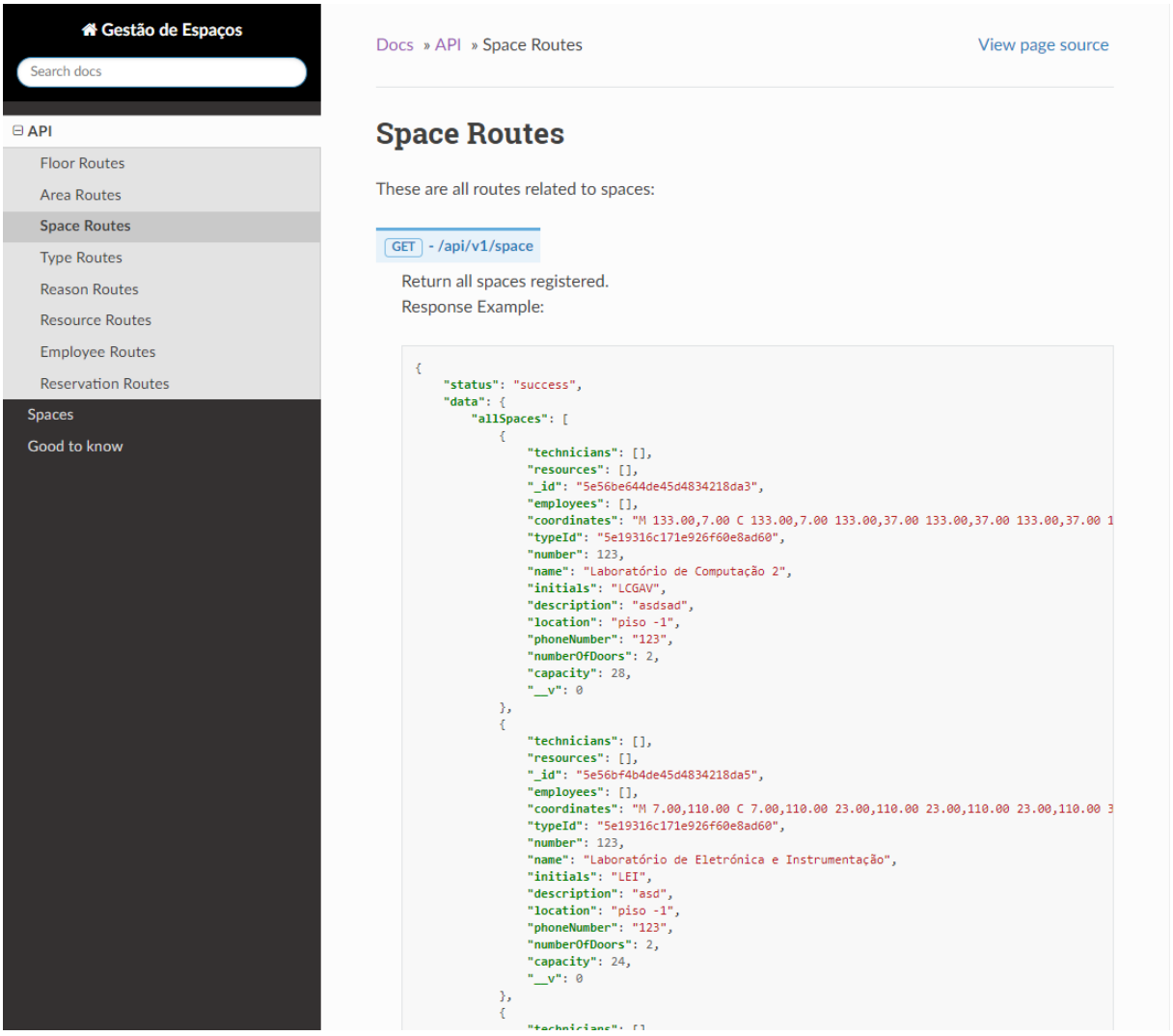

Figura 4.5: Parte da documentação da API.

Requirements to run the project

- node & npm

How to run the database

- run `sudo mongod --fork --logpath /var/log/mongodb/mongod.log --auth --port 27017 --dbpath /var/lib/mongodb`

How to run the project

- run `npm install` in the project's root folder to install all dependencies if there are any that are not installed yet
- run `node ./app.js` or `nodemon ./app.js` or `pm2 start ./app.js` in the project's root folder

How to use the mongo CLI

In case you want to use the mongo shell run the following command:

- run `mongo --authenticationDatabase "admin" -u DB_USERNAME --password DB_PASSWORD`

Useful Information

Mongo commands to DUMP and RESTORE data

`mongodump --host localhost --port 27017 --authenticationDatabase admin --username DB_USERNAME --password DB_PASSWORD --db space-management --archive=/backup/db.gz --gzip`

`mongorestore --host localhost --port 27017 --authenticationDatabase admin --username DB_USERNAME --password DB_PASSWORD --gzip --archive=/backup/db.gz --db space-management`

Figura 4.6: Parte da documentação do back-end.

4.7 Logs e erros

Como dito anteriormente no Capítulo 2, o javascript permite que *bugs* fiquem ocultos e existam até mesmo em produção. Por isso no *back-end* foi implementado *logs* de erros, isto é, ficheiros de texto que armazenam a data, horário e a mensagem do erro que ocorreu. Os *logs* são importantes pois auxiliam no momento de resolver um problema inesperado e fazer o *debug*. A implementação dos *logs* foi relativamente fácil graças a forma como foram tratados os erros. Quando é lançada uma exceção, esta exceção é tratada em um único lugar. Desta forma a função que escreve os *logs* pode ser chamada uma única vez, evitando que a função tenha que ser chamada em todos os lugares onde seja possível existir uma exceção.

Além dos *logs*, também foi implementado o envio de emails dos erros que ocorreram e a partir do momento em que ocorreram. A ideia é enviar emails para a pessoa, ou equipa responsável por prestar manutenção caso ocorra algo inesperado. Desta forma, os problemas que surgirem serão imediatamente relatados para a pessoa/equipa responsável pelo sistema, consequentemente otimizando o tempo de resolução do problema.

4.8 Autenticação

Como dito na seção 3.1 a autenticação dos utilizadores é feita através do LDAP. Para que a autenticação ocorra com sucesso, o utilizador deve inserir o id de utilizador e a palavra passe, que devem estar registrados na base de dados utilizada pelo servidor LDAP mais especificamente na *organizational unit* (ou) *staff*. O utilizador só conseguirá fazer a autenticação se estiver no grupo *staff*, isso significa que alunos não conseguem aceder ao sistema, pois não pertencem a este grupo. Além disso, a requisição para o servidor LDAP deve ser feita por um endereço Internet Protocol (IP) reconhecido, isto é, existe uma *whitelist* onde são colocados endereços IP que podem fazer requisições para o servidor.

Mesmo havendo a autenticação com o LDAP, há um único utilizador que autentica-se de forma diferente, sendo ele o utilizador com permissão de gestor do sistema. Este

utilizador foi criado para que fosse possível realizar a gestão da permissão dos utilizadores. Assim este utilizador será como um administrador geral, mesmo que hajam outros gestores ou apenas um gestor do sistema. Caso ele/eles não consigam aceder ao sistema por algum motivo, por exemplo, demissão ou férias, há sempre uma conta que poderá ser utilizada para gerir as permissões.

Essa conta que não utiliza a autenticação LDAP está cadastrada diretamente na base de dados. Para evitar que a senha estivesse exposta diretamente na base de dados, foi utilizada uma biblioteca chamada "bcrpyt" para fazer o *hashing* da senha, isto é, a partir da senha é gerada uma sequência de caracteres que não podem ser revertidos a senha original.

4.9 Deploy

Para realizar o *deploy* da ferramenta foi utilizado o Nginx, um servidor *proxy* reverso e HTTP, tanto para o *front-end* quanto para o *back-end*. A escolha se deu pelo facto do Nginx ser mais fácil de ser utilizado quando comparado ao Apache, outro servidor HTTP. Além disso, assim como o Apache, o Nginx é muito popular e é utilizado por grandes empresas como a Uber, Airbnb, Instagram, etc.

O Nginx foi configurado em uma Virtual Machine (VM) com o sistema operativo Ubuntu 20.04. Nesta VM encontram-se os projetos do *front-end* e do *back-end*. É importante que o *back-end* corra exactamente nessa VM, pois a autenticação feita pelo LDAP exige que as requisições sejam feitas por endereços IP reconhecidos como mencionado anteriormente na seção 4.8. A aplicação foi então disponibilizada no endereço "https://reservas.estig.ipb.pt/" para realização dos testes.

Para utilizar o protocolo Hypertext Transfer Protocol Secure (HTTPS) e tornar a comunicação entre o cliente e o servidor mais segura, foram utilizados certificados TLS emitidos pela autoridade de certificação gratuita "Let's encrypt". O protocolo TLS possui o objetivo principal de fornecer privacidade e integridade dos dados durante a comunicação entre duas aplicações [11]. Os certificados emitidos pela "Let's encrypt" são válidos

durante 90 dias, dessa forma chaves roubadas e certificados emitidos incorretamente são válidos por menos tempo. O facto dos certificados serem válidos durante um período de tempo curto não é um problema, pois eles são renovados automaticamente antes de serem expirados.

A aplicação React foi desenvolvida utilizando o ambiente *create react app*, sendo esta a forma suportada oficialmente e sugerida na documentação gerida pelo Facebook. Com este ambiente é possível gerar a versão minimizada dos pacotes da aplicação com um único comando e desta forma, fazer o *deploy* do *front-end* é uma tarefa simples e rápida. Após gerar a versão minimizada do projeto só é preciso reiniciar o servidor Nginx para que o *front-end* seja atualizado.

Para fazer o *deploy* da aplicação Node.js foi utilizado o gerenciador de processos PM2 para Node.js. O PM2 possui um balanceador de carga integrado, permite manter as aplicações ativas o tempo todo e recarregá-las sem *downtime*. O balanceador de carga funciona ao utilizar o modo *cluster*. Este é um modo especial ao iniciar uma aplicação Node.js que permite o início de múltiplos processos e distribui a carga entre eles. A quantidade de processos que podem ser utilizados depende do número de *cores* disponíveis.

4.10 Testes

Foram realizadas reuniões para testes após a realização do *deploy*, possibilitando que qualquer máquina com conexão a internet pudesse aceder a ferramenta. Com o sistema acessível, foram testados todos os casos de uso com as pessoas que realmente irão utilizar o sistema, sendo elas os funcionários que realizam a gestão dos pedidos de reserva atualmente e os docentes.

Após as primeiras reuniões de teste foram sugeridas algumas alterações para melhorar a experiência do utilizador e tornar a ferramenta mais *user friendly* de forma geral. Entre as alterações sugeridas estão: alterar o texto de botões, para tornar a ação esperada mais evidente, manter a consistência dos ícones utilizados para as mesmas ações, adicionar um botão para voltar a página anterior na página da tabela de horários, etc.

Após a realização das alterações sugeridas e outras reuniões para testes, pode-se conjecturar que o sistema desenvolvido possibilita a gestão dos espaços, pedidos de reserva e armazena as informações dos recursos e pessoas associadas a cada espaço. Além da gestão, o sistema também possibilita que as reservas sejam feitas de forma coordenada com a aplicação dos horários/sumários.

4.11 Discussão

Entre o que acredita-se que poderia ter sido feito de forma diferente, está a entrega das funcionalidades que foram sendo desenvolvidas ao longo do tempo, a partir do momento em que estavam prontas. Em engenharia de *software* esta prática é chamada de entrega contínua. Dessa forma, acredita-se que a fase de testes com os funcionários que utilizarão o sistema poderia ter sido mais rápida. Além disso, poderia ter sido implementado testes automatizados, para auxiliar na redução de erros e no esforço exigido para realizar todos os testes necessários.

A funcionalidade que permite o *download* de ficheiros excel das reservas aprovadas e o facto da ferramenta desenvolvida permitir o *upload* de imagens para possibilitar a navegação com base nas plantas são os pontos onde se foi para além dos objetivos iniciais.

A possibilidade de fazer o *download* dos ficheiros excel permite que possam ser feitas análises interessantes como por exemplo:

- Qual a média de duração das reservas.
- Quantas reservas reservas foram feitas durante o semestre.
- Quem são as pessoas que mais fizeram reservas.
- Quais são os espaços mais reservados.
- Quais são os horários mais reservados.
- Qual o tempo médio que um determinado espaço fica reservado por dia.

Além destas análises, podem ser feitas muitas outras. Essas informações são úteis pois podem auxiliar nas tomadas de decisões, por exemplo, verificar que um espaço é muito pouco utilizado para dar aulas e decidir utilizá-lo para outra finalidade.

O facto da ferramenta permitir o *upload* das imagens para a navegação, significa que pode ser feito o *upload* de plantas de outras construções e não apenas da ESTiG, ou seja, torna-se mais fácil a adaptação que seria necessária realizar para utilizar a ferramenta em outras escolas. Entretanto, existem dois fatores que limitam o funcionamento do sistema desenvolvido na ESTiG e que são: a autenticação via LDAP e a importação dos eventos da aplicação dos horários/sumários. Apesar disso, alterar o tipo de autenticação e remover a importação dos eventos, são tarefas muito simples que podem ser feitas rapidamente pois já está implementada a autenticação através da base de dados (MongoDB). Para remover a importação dos eventos, basta parar a execução do *cron job* que está a importar os eventos atualmente.

Capítulo 5

Conclusões

Neste capítulo são apresentadas as conclusões e conhecimentos gerados referentes ao trabalho desenvolvido e sugestões para possíveis trabalhos futuros.

O principal objetivo deste trabalho era desenvolver uma ferramenta *web*, que com base nas plantas da escola, permitisse manter as informações sobre os espaços e reservas da ESTiG atualizadas. Para além disso, também esperava-se que a ferramenta possibilitasse que as reservas fossem feitas de forma coordenada com a aplicação dos horários/sumários. Portanto, acredita-se que os objetivos foram todos cumpridos com a ferramenta desenvolvida e espera-se que a mesma possa facilitar a gestão das informações e reservas dos espaços da escola.

Este trabalho resultou em muito conhecimento que certamente poderá ser utilizado no futuro, em áreas como *front-end*, *user interface*, *user experience*, *back-end*, base de dados, segurança, *web server*, administração de sistemas. Ou seja, o trabalho gerou conhecimento suficiente para desenvolver uma aplicação a partir do zero e torná-la disponível às pessoas.

A fim de aprimorar a ferramenta desenvolvida de forma geral, são sugeridas as seguintes implementações para trabalhos futuros: automatização do *deploy*, pois apesar de que já foi implementado um *script shell* para facilitar ainda mais o processo de *deploy*, a automatização pode otimizar ainda mais o tempo necessário; automatização de testes, pois assim como dito Capítulo 4, a automatização dos testes contribui para a redução dos erros; recuperação automática dos *backups* da base de dados, pois apesar do automatismo

para gerar os *backups* e a necessidade de recuperação não ser algo recorrente, caso algum problema ocorra e seja necessário fazer a recuperação dos dados, a mesma tem que ser feita de forma manual.

Bibliografia

[1] *A ESTiG Descrição Geral*, http://estig.ipb.pt/index.php/estig/a-estig, Acessado em: 2020-03-18.

[2] *HTML 5.2*, https://www.w3.org/TR/html52/introduction.html#background, Acessado em: 2020-03-31.

[3] *What is JavaScript?* https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript, Acessado em: 2020-03-31.

[4] M. Selakovic e M. Pradel, "Performance Issues and Optimizations in JavaScript: An Empirical Study," em *Proceedings of the 38th International Conference on Software Engineering*, sér. ICSE '16, Austin, Texas: Association for Computing Machinery, 2016, pp. 61–72, ISBN: 9781450339001. DOI: 10.1145/2884781.2884829. URL: https://doi.org/10.1145/2884781.2884829.

[5] *HTML CSS*, https://www.w3.org/standards/webdesign/htmlcss, Acessado em: 2020-03-31.

[6] *SASS Documentation*, https://sass-lang.com/documentation, Acessado em: 2020-03-31.

[7] E. Wohlgethan, *SupportingWeb Development Decisions by Comparing Three Major JavaScript Frameworks: Angular, React and Vue.js*, eng, 2018.

[8] R. Deari, X. Zenuni, J. Ajdari, F. Ismaili e B. Raufi, "Analysis And Comparision of Document-Based Databases with Relational Databases: MongoDB vs MySQL,"

em *2018 International Conference on Information Technologies (InfoTech)*, 2018, pp. 1–4.

[9] M. Villamizar, O. Garcés, H. Castro, M. Verano, L. Salamanca, R. Casallas e S. Gil, "Evaluating the monolithic and the microservice architecture pattern to deploy web applications in the cloud," em *2015 10th Computing Colombian Conference (10CCC)*, 2015, pp. 583–590.

[10] K. Gos e W. Zabierowski, "The Comparison of Microservice and Monolithic Architecture," em *2020 IEEE XVIth International Conference on the Perspective Technologies and Methods in MEMS Design (MEMSTECH)*, 2020, pp. 150–153.

[11] E. Rescorla e T. Dierks, *The Transport Layer Security (TLS) Protocol Version 1.2*, RFC 5246, ago. de 2008. DOI: 10.17487/RFC5246. URL: https://rfc-editor.org/rfc/rfc5246.txt.