



ISEL

ADEETC

Área Departamental de
Engenharia Electrónica e
Telecomunicações e
de Computadores

Instituto Superior de Engenharia de Lisboa

Licenciatura em Engenharia Informática e de Computadores

Projeto e Seminário - Semestre de verão 2019/2020
Relatório de projeto

SiGeMaDi - Sistema de Gestão de Material Didático

44775 Carolina Santos A44775@alunos.isel.pt 913 901 230
44783 Tiago Duarte A44783@alunos.isel.pt 911 866 262
44813 Francisco Fialho A44813@alunos.isel.pt 925 594 608

Orientador

Pedro Miguens (ADEETC) pedro.miguens@isel.pt

21 de Julho de 2020

Agradecimentos

A realização deste projeto não seria possível sem a ajuda, direção, e acompanhamento do orientador de projeto, Engenheiro Pedro Miguens, o qual agradecemos todo o seu esforço e dedicação.

Por fim, gostaríamos de agradecer aos nossos professores de licenciatura, pela dedicação e por todos os ensinamentos e conhecimentos transmitidos que contribuíram para o nosso enriquecimento pessoal e profissional.

Julho 2020, Grupo 12

Queria agradecer a todos os meus colegas, que me acompanharam ao longo da licenciatura, que termina com este projeto, em especial aos meus colegas de trabalho Carolina Santos e Tiago Duarte que foram incansáveis e dedicados durante todo o percurso do projeto, bem como no decorrer de todo o curso, no qual tive o privilégio de pertencer em grupos de trabalho desde o primeiro ano.

Não podendo esquecer todo o apoio, consideração e interesse, por parte dos meus pais, Sandra e Fernando, da minha irmã Silvana, e toda a minha família, que consequentemente e consecutivamente me incentivaram a trabalhar melhor e com mais confiança.

Julho 2020, Francisco Fialho

Gostaria de agradecer a todos os meus amigos que estiveram ao meu lado ao longo da minha licenciatura, especialmente aos meus colegas Francisco Fialho e Tiago Duarte que sempre demonstraram ser pessoas esforçadas, dedicadas e dispostas a ajudar em tudo o que precisei. Não me imaginaria a realizar este projeto com mais ninguém, dado que este é o grupo que me acompanha desde o primeiro ano.

Por fim, gostaria de agradecer aos meus pais, Sandra e Miguel, ao meu irmão João e ao meu namorado Rui, que sempre me apoiaram em todas as minhas decisões e estiveram ao meu lado possibilitando um término de licenciatura com sucesso.

Julho 2020, Carolina Santos

Gostaria de agradecer a todos os meus amigos que estiveram ao meu lado ao longo da minha licenciatura, especialmente aos meus colegas Francisco Fialho e Tiago Duarte que sempre demonstraram ser pessoas esforçadas, dedicadas e dispostas a ajudar em tudo o que precisei. Não me imaginaria a realizar este projeto com mais ninguém, dado que este é o grupo que me acompanha desde o primeiro ano.

Por fim, gostaria de agradecer aos meus pais, Sandra e Miguel, ao meu irmão João e ao meu namorado Rui, que sempre me apoiaram em todas as minhas decisões e estiveram ao meu lado possibilitando um término de licenciatura com sucesso.

Julho 2020, Carolina Santos

Gostaria de agradecer a todos os meus colegas de trabalho e amigos que me apoiaram durante o decurso da licenciatura e que de alguma forma contribuíram para a minha aprendizagem e desenvolvimento enquanto pessoa, mais concretamente aos meus colegas Carolina Santos e Francisco Fialho pelo apoio e compreensão durante estes três anos. Gostaria ainda de agradecer à minha família, por me ter dado todo o apoio necessário, tanto nos piores, como nos melhores momentos sendo que sem essa força teria sido, sem sobre de dúvida, um caminho mais difícil.

Julho 2020, Tiago Duarte

Resumo

No paradigma atual os sistemas de informação revelam ser uma ferramenta fundamental para o bom funcionamento de qualquer organismo. Pensando na área de ensino, sistemas com estas características possuem um conjunto de vantagens que levam a muitas unidades do setor a implementá-los na sua infraestrutura, uma vez que permitem o mapeamento de grandes quantidades de informação, extração de informação para análise estatística para posteriores tomadas de decisão, melhor controlo sobre a informação, entre outras. Contextualizando para a organização alvo (ISEL - Instituto Superior de Engenharia de Lisboa) a mesma carece de suporte a ferramentas deste género, mais concretamente viradas para a gestão de materiais didáticos usados em laboratório pelos alunos e docentes.

O presente documento visa abordar esta questão expondo o problema em causa explorando as atuais ferramentas existentes no mercado e analisando os seus pontos fortes e fracos, extrair ideias e em combinação com outras previamente concebidas implementar um sistema de gestão de material didático que permite dar suporte às técnicas previamente usadas pela organização alvo.

Ainda, é nele que poderá encontrar informação relativa à forma como foi solucionado o sistema, na medida em que é exposta uma arquitetura possível do mesmo bem como um modelo de dados para dar suporte ao mapeamento da informação existente na organização e ainda detalhes técnicos de implementação.

Posteriormente, poderá encontrar dados estatísticos realizados de forma a comprovar a eficiência da nossa solução e ainda um conjunto de ideias a ter em conta em soluções posteriores à que é apresentada neste documento.

A solução proposta neste documento tem por base a implementação de dois grandes módulos, *backend* e *frontend*, sendo que cada um é composto pelos seus respetivos componentes, nomeadamente o lado *backend* é composto por uma base de dados - implementada em *PostgreSQL* - responsável por armazenar a informação e por uma componente servidora - implementada na linguagem *Kotlin* com *framework Spring MVC*, e o lado *frontend* é composto por uma aplicação Web - implementada com a *framework React* - e uma aplicação móvel - implementada em *Xamarin*.

Abstract

In the current paradigm, information systems prove to be a fundamental tool for the proper functioning of any organism. On the educational sector, systems with these characteristics have a set of advantages that lead many units of the sector to implement them in their infrastructure, since they allow the mapping of large amounts of information, extraction of information for statistical analysis for later decision-making, better control over information, among others. Contextualizing the target organization (ISEL - Instituto Superior de Engenharia de Lisboa), it lacks support for tools of this kind, more specifically aimed at the management of teaching materials used in the laboratory by students and teachers.

This document aims to address this issue by exposing the problem in question by exploring the current tools on the market and analyzing their strengths and weaknesses, extracting ideas and in combination with others previously designed to implement a didactic material management system that allows supporting the techniques previously used by the target organization.

Here you can find information related to how the system was solved, as it exposes a possible architecture of the system as well as a data model to support the mapping of existing information in the organization and technical implementation details.

Later, you will be able to find statistical data carried out in order to prove the efficiency of our solution and also a set of ideas to be taken into account in solutions after the one presented in this document.

The solution proposed in this document is based on the implementation of two large modules, *backend* and *frontend*, each of which is composed of its respective components, namely the *backend* side is composed of a database - implemented in *PostgreSQL* - responsible for storing the information and for a server component - implemented in the *Kotlin* language with *Spring MVC* framework, and the *frontend* side is composed by a web application - implemented with the *framework React* - and a mobile application - implemented in *Xamarin*.

Índice

1	Introdução	1
1.1	Objetivos	1
1.2	Organização do documento	1
2	Estado da arte	2
3	Solução Proposta	3
3.1	Aplicação Web	3
3.2	Aplicação Móvel	4
3.3	Base de Dados e <i>Web API</i>	4
3.4	Escolha das tecnologias e fundamentação	4
4	Modelo de dados	7
4.1	Análise e escolha do modelo de dados	7
4.2	Modelo Entidade-Associação	8
5	Detalhes de implementação	12
5.1	Aplicação Móvel	12
5.1.1	Detalhes de implementação da aplicação móvel	15
5.2	Aplicação Web	18
5.2.1	Detalhes de implementação da aplicação Web	40
5.3	Componente Servidora	42
5.4	Detalhes de implementação da componente servidora	42
6	Resultados Experimentais	44
6.1	Metodologia	44
6.2	Resultados experimentais	44
6.2.1	GET /sigemadi/api/materials	45
6.2.2	POST /sigemadi/api/materials	46
7	Conclusões e Trabalho Futuro	49
7.1	Conclusões	49
7.2	Trabalho Futuro	49
A	Documentação da API	54

Lista de Figuras

3.1	Estrutura da solução proposta	3
4.1	Escalamento Vertical [1]	7
4.2	Escalamento Horizontal [1]	8
4.3	Modelo Entidade Associação	11
5.1	Diagrama de navegação da aplicação móvel	12
5.2	<i>Mocks</i> da aplicação móvel - aluno	13
5.3	<i>Mocks</i> da aplicação móvel - aluno	14
5.4	<i>Mocks</i> da aplicação móvel - funcionário	14
5.5	<i>Mocks</i> da aplicação móvel - funcionário	16
5.6	<i>Mocks</i> da aplicação móvel - funcionário	17
5.7	Diagrama de Navegação da Aplicação <i>Web</i>	19
5.8	Página Principal	20
5.9	Início de Sessão	20
5.10	Seleção de Cargo	21
5.11	Página Principal do Administrador	21
5.12	Utilizadores	22
5.13	Definir Cargos	22
5.14	Áreas Científicas	23
5.15	Unidades Curriculares de uma Área Científica	23
5.16	Estatísticas	24
5.17	Página Principal do Responsável de Laboratório	24
5.18	Gestão de Materiais	25
5.19	Detalhes do Material	25
5.20	Eliminar Material Avulso	26
5.21	Adicionar Materiais	26
5.22	Tipos de Materiais	27
5.23	Adicionar Tipos de Materiais	27
5.24	Página Principal do Funcionário	28
5.25	Ações rápidas	28
5.26	Material	29
5.27	Detalhes do Material	29
5.28	Reportar Avaria	30
5.29	Requisições	30
5.30	Detalhes da Requisição	31
5.31	Remover Materiais da Requisição	31
5.32	Terminar Requisição	32
5.33	Leitura do <i>QR Code</i>	32
5.34	Nova Requisição	33
5.35	Finalizar a Nova Requisição	33
5.36	Página Principal do Técnico	34
5.37	Avarias	35
5.38	Detalhes da Avaria	35
5.39	Histórico de Avarias	36
5.40	Detalhe de Avaria Passada	36
5.41	Página Principal do Docente	37
5.42	Material	37
5.43	Detalhes do Material	38
5.44	Reservar Material	38
5.45	Finalizar Reserva	39
5.46	Escolha da Data para a Reserva	39
5.47	Reservas	40
5.48	Detalhes da Reserva	40
6.1	Tempos de execução para o <i>endpoint</i> GET /sigemadi/api/materials	45
6.2	Corpo do pedido <i>HTTP</i> usado para criar as unidades	46
6.3	Dados recolhidos para o <i>endpoint</i> POST /sigemadi/api/materials	46

1 Introdução

Na conjectura atual, os Sistemas de Informação (SI) revelam ser uma peça fundamental para o desenvolvimento de qualquer organismo, uma vez que proporcionam suporte aos mecanismos existentes, bem como elevam o rendimento do mesmo. Ainda, proporcionam uma otimização do fluxo de informação, permitindo maior agilidade e organização, redução de custos operacionais e administrativos e ganho de produtividade, maior integridade e veracidade da informação, maior estabilidade e finalmente, maior segurança no acesso à informação.

Centralizando o problema para os estabelecimentos de ensino, estes por vezes requerem de um tipo específico de SI focado para a logística de materiais didáticos dentro da organização, na medida em que certos materiais são valiosos e indispensáveis para a boa aprendizagem e funcionamento da mesma. Ainda, a requisição desses materiais necessita de ser registada para um maior controlo sobre o inventário, podendo ser vantajoso a utilização destes sistemas de forma a facilitar os registo e possibilitar a análise de dados.

1.1 Objetivos

O objetivo do presente projeto consiste no desenvolvimento de um sistema de informação focado para a gestão de material didático de forma a desmaterializar o processo já existente na organização alvo. A requisição de material constitui ser o principal objetivo do sistema tendo duas perspetivas principais, nomeadamente a do aluno e a do funcionário. Na ótica do aluno, é possível consultar material disponível no sistema, bem como a visualização do material que este tem em sua posse. Do ponto de vista do funcionário, este será capaz de analisar o conjunto de material detido por um aluno, recorrendo apenas à sua identificação, e associar o material a requisitar por este com recurso à leitura de um *QR Code*, presente em cada item. Ainda, existe a possibilidade de reportar avarias dos materiais, e consultar dados estatísticos para tomadas de decisão.

1.2 Organização do documento

O restante documento encontra-se organizado por quatro capítulos. O Capítulo 2 (Estado da arte) onde são descritas as soluções já existentes no mercado, as suas vantagens e desvantagens, bem como é feita uma comparação com a solução que se pretende implementar. No Capítulo 3 (solução Proposta) é apresentada, detalhadamente a solução que se pretende implementar, nomeadamente a arquitetura do sistema, o modelo de dados e a fundamentação para a escolha das tecnologias.

No Capítulo 4 (Modelo de dados) é feita uma descrição do modelo de dados do sistema, no Capítulo 5 (Detalhes de implementação) são descritas, as soluções de implementação. No Capítulo 6 (Resultados Experimentais) são apresentados testes desenvolvidos para averiguar a eficiência do serviço implementado bem como concluir o que se pode melhorar no mesmo para obter melhor *performance*.

Ainda, no capítulo 7 (Conclusões e Trabalho Futuro) é realizada uma reflexão de tudo aquilo que foi implementado nesta fase do projeto, averiguar coisas que se poderiam ter feito melhor e propor opções que no futuro se poderiam acrescentar à solução.

2 Estado da arte

Para poder compreender as diferentes aplicações existentes no mercado que visam solucionar o nosso problema, foi realizada uma pesquisa na qual foram identificadas três entidades, *Parago Software* [2], *EZOfficeInventory* [3] e *Sortly* [4].

A *Parago Software* é, tal como o nome indica, uma empresa de *software* focada em desenvolvimento para o ensino, proporcionando vários sistemas de informação que visam suportar uma vasta gama de áreas dentro do ensino, como por exemplo gestão de inventário, gestão de capital, gestão de risco, gestão de incidentes, gestão de instalações, etc... O sistema com características similares ao projeto aqui proposto é o *Asset, IT & Inventory*.

Este sistema é composto por uma componente móvel e uma componente *desktop* tendo como principal objetivo realizar o *tracking*, a gestão, o aluguer, a auditagem, a localização de inventário pertencente a uma determinada organização, fornecer dados estatísticos para tomadas de decisão mais fundamentadas, nomeadamente, o inventário mais e menos usado, permitir o registo detalhado de todos os ativos presentes num inventário, monitorizar o uso de ativos e restringir permissões de acesso.

Ainda, através de dispositivos móveis, é possível realizar o *tracking*, o *loaning*, a edição, localização e *device deployment* (entrega de ativos aos estudantes e professores) sobre o inventário.

O *EZOfficeInventory* é um sistema *Web-based* de *tracking* fundado pela Universidade do Illinois (Estados Unidos) como uma solução *Software as a Service*(SaaS) que depois foi comprada pela *EZ Web Enterprises*.

O sistema suporta a gestão de inventário, a integração com dispositivos móveis, a reserva de ativos e notificação de compra dos mesmos, a gestão de materiais situados noutras localizações, a análise de dados estatísticos tais como custos, aumento de produtividade, entre outras funcionalidades. Tem ainda como particularidade a leitura de ativos por *QR Code*.

Através da aplicação móvel é possível consultar os ativos que se encontram disponíveis, localizar o material, bem como fazer o seu levantamento.

O sistema *Sortly* tem essencialmente o mesmo propósito que os outros sistemas aqui enumerados também composto por duas aplicações, móvel e *Web*. No geral o sistema fornece funcionalidades como o *tracking* de ativos, recorrendo à leitura por *QR Code*, a notificação ao cliente aquando de escassez de inventário, interoperabilidade com outros dispositivos (*iOS, Android, tablet, computadores*), gerir o controlo de acessos dos utilizadores, realizar o *tracking* da atividade de cada utilizador e reportação de relatórios de atividade, e do estado atual do inventário.

Com a análise das soluções encontradas durante a pesquisa, foi possível concluir que todos eles possuem características semelhantes de acordo com o propósito que os mesmos possuem, contudo existem algumas diferenças.

A *Parago Software* foca-se mais no contexto global de uma organização de ensino, em comparação com os outros dois sistemas, na medida em que a gestão de inventário é realizada entre organismos que compõem a mesma e dá a possibilidade de realizar aluguer de ativos com outros organismos. Contextualizando para o nosso problema, este não é tão globalizado na medida em que não trata questões entre partilha de ativos entre organismos mas sim sobre uma área em concreto dentro de um organismos, nomeadamente dentro de um departamento.

O *EZOfficeInventory* possui uma característica, considerada importante, como a leitura de ativos através de *QR Code*.

No entanto, a nossa proposta permite a reserva de material para uma aula prática e posterior preparação da mesma recorrendo aos mecanismos fornecidos pelo sistema de modo a que o processo seja mais eficiente e rápido possível. O sistema proposto permite reportar avarias nos ativos e acompanhamento subsequente, permitindo estabelecer e garantir histórico de avarias, essenciais em gestão preventiva.

3 Solução Proposta

Nesta secção é apresentada e discutida a solução proposta e dos componentes constituintes desta, bem como são fundamentadas as tecnologias usadas, para a respetiva implementação.

O sistema proposto considera seis (6) papéis fundamentais: administrador; responsável de laboratório; técnico; docente; funcionário e estudante. Estes terão acesso ao sistema através de duas aplicações, uma móvel e uma Web. A primeira com 2 interfaces, nomeadamente a de funcionário e de estudante, enquanto que a segunda terá 5 interfaces: *i*) de funcionário; *ii*) de docente; *iii*) de técnico; *iv*) de responsável de laboratório; e *v*) de administrador. Além das aplicações cliente, o sistema será constituído por uma base de dados e uma *Web API*.

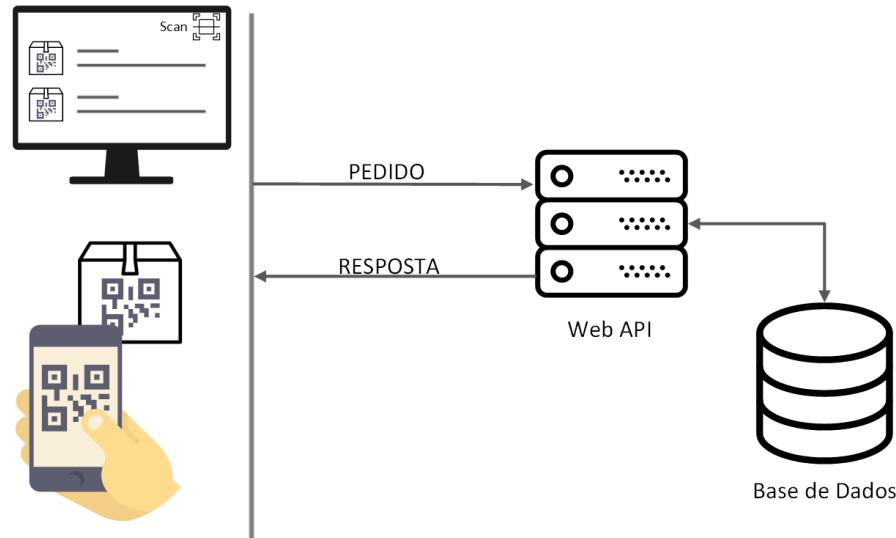


Figura 3.1: Estrutura da solução proposta

3.1 Aplicação Web

A aplicação *Web* fornece controlo sobre a disponibilidade de material didático, bem como permitirá registar o material associado ao estudante, a consulta de material danificado e a marcação pontual de aulas práticas. Além disso, permite a análise de dados estatísticos para tomadas de decisão, por exemplo, verificar se existe material suficiente para o volume de requisições feitas durante de um período específico. Contudo, todas as funcionalidades enumeradas não serão todas realizadas pela mesma entidade, mas sim por subconjuntos afetos a cada um dos seis papéis anteriormente enumerados, consequentemente são necessárias interfaces distintas para cada papel.

A interface de administrador, *v*), terá como principal objetivo a gestão de cargos no sistema, nomeadamente gerir as permissões e cargos dos utilizadores do sistema, e visualizar os dados estatísticos.

No futuro, a aplicação poderá permitir que certos dados de consulta, restrita ao administrador, se tornem visíveis para outras entidades do sistema escolhidas pelo mesmo.

A interface de responsável de laboratório, *iv*), disponibiliza a gestão (adição, remoção e atualização) de material no inventário, bem como permite obter dados estatísticos para tomadas de decisão.

A interface do técnico, *iii*), estará focada na consulta de material danificado, podendo alterar o estado do mesmo, isto é, colocar o material em processo de reparação fazendo com que esse material em concreto não esteja disponível para requisição.

A interface de docente, *ii*), focar-se-á na marcação de aulas práticas não previstas no horário pré-definido podendo especificar qual o material necessário à realização desta.

A interface de funcionário, será a mais complexa das interfaces, visto que esta é a que implementa mais funcionali-

dades, nomeadamente, poder obter a informação de um estudante recorrendo à leitura do cartão do mesmo, associar material ao aluno, recorrendo a leitura de *QR Code* presente em cada item, consultar o material por aluno e por disponibilidade e reportar avarias de material.

3.2 Aplicação Móvel

A aplicação móvel permitirá garantir a mobilidade e a rapidez no registo de material, sendo que esta requer duas interfaces distintas para duas entidades diferentes.

A interface de funcionário estará focada na consulta dos materiais requisitados e de quem os detém (estudantes ou docentes), bem como na preparação de aulas práticas, isto é, com o auxílio da câmara do dispositivo registar quais os materiais que vão ser requisitados e associar os mesmos ao docente que realizará a aula. Também, será possível o registo de avaria de materiais que não estejam requisitados.

No futuro, a interface poderá ter a funcionalidade acrescida de notificar os alunos com material requisitado para o entregarem, dado que pode existir uma reserva feita por um docente e a entrega desse material permita a realização da aula.

A interface do estudante focar-se-á na consulta de material didático disponível para requisição bem como, visualizar o material requisitado e que ainda se encontra na sua posse. Ainda, será possível o registo de avarias dos materiais/equipamentos requisitados.

No futuro, o sistema poderá implementar as seguintes funcionalidades extra:

- Na eventualidade de o aluno não possuir identificação, ser gerado um *QR Code* temporário como meio de autenticação;
- Ter um mecanismo de reserva de material, para um determinado horário.

3.3 Base de Dados e *Web API*

A base de dados servirá como um meio de suporte à infraestrutura com a responsabilidade de armazenar os dados necessários e fornecer dados para estatística. Definem-se os requisitos funcionais deste componente como todas as ações estabelecidas anteriormente nas interfaces que requeiram obtenção de dados e modificação dos mesmos.

A *Web API* terá o papel de interveniente entre as aplicações, *Web* e móvel, e a base de dados, sendo que fornece representações dos dados e permite a alteração dos mesmos.

3.4 Escolha das tecnologias e fundamentação

A implementação da aplicação móvel poderia ser realizada recorrendo a duas abordagens distintas, de forma nativa ou multi-plataforma. Uma das principais vantagens de implementar uma aplicação nativa, devem de utilizar recursos específicos da plataforma alvo, tornando a aplicação mais eficiente.

Por outro lado, pelo facto de se pretender implementar a aplicação móvel para os sistemas operativos *iOS* e *Android*, ao abordarmos o problema de forma nativa iria ser necessário despender mais tempo e recursos para o realizar, uma vez que seria necessário aprender linguagens de programação dedicadas, por exemplo, *Swift* (*iOS*) e aprender conceitos e técnicas sobre cada plataforma.

Comparativamente com aplicações multi-plataforma, estas requerem de um menor custo de desenvolvimento, na medida em que o código é criado apenas uma vez e pode ser usado por um conjunto de sistemas operativos diferentes. Contudo, uma aplicação implementada com esta solução não revela ser tão eficiente como outra que seja implementada de forma nativa.

Desta forma, optou-se por desenvolver a aplicação recorrendo a ferramentas multi-plataforma. Após uma análise sobre o conjunto de ferramentas existentes no mercado para executar esta tarefa, surgiram-nos três, nomeadamente, *Xamarin* [5], *React Native* [6] e *Flutter* [7].

Xamarin é uma ferramenta que apresenta bastantes vantagens quando comparada com as outras mencionadas, nomeadamente, o código ser compilado de forma nativa, permitindo criar aplicações com desempenho elevado, com aparência e comportamento nativos, fazendo com que a aplicação tenha a aparência e comportamento expectável por parte do utilizador, e o facto de as aplicações terem acesso a todas as funcionalidades expostas pelo dispositivo e pela plataforma, incluindo recursos específicos da mesma. Dado que *Xamarin* trabalha com o mesmo código de maneira a acelerar o processo de desenvolvimento, permite que cerca de 90% do código criado seja o mesmo [8]

independentemente da plataforma destino. Além disso, permite evocar o código da plataforma existente, como por exemplo *Swift* para *iOS*, caso seja preciso reutilizar alguns módulos ou realizar algumas funções específicas da plataforma. No entanto, esta ferramenta também apresenta algumas desvantagens, dado que é necessário comprar uma licença para o *Microsoft's Visual Studio* [9] e não é recomendado a sua utilização para aplicações graficamente exigentes, pelo facto de cada plataforma ter uma diferente maneira de realizar a representação visual.

React Native tem a vantagem de usar *Hot Reloading* [10], que tem como objetivo continuar a execução da aplicação e injetar novas versões dos ficheiros editados em tempo de execução, permitindo ao programador visualizar as mudanças feitas no código em segundos, ao contrário de ferramentas que usam componentes nativos, que podem demorar minutos. Para além disso, um dos maiores focos desta ferramenta é ter boa renderização gráfica.

Apesar destas vantagens, esta ferramenta apresenta alguns problemas. *React Native* não é completamente multi-plataforma, ou seja, para ser possível usar alguns componentes nativos como por exemplo a câmara, é necessário desenvolver código específico para a plataforma alvo. Também, comparado a aplicações nativas, por vezes tem pior *performance*.

Assim como as outras ferramentas já mencionadas, *Flutter* [7] apresenta também vantagens e desvantagens. Assim como *React Native*, *Flutter* usa *Hot Reloading*, as aplicações são desenvolvidas utilizando a linguagem *Dart*, que é uma linguagem de programação orientada a objetos de fácil compreensão e contém uma vasta biblioteca de *widgets*, permitindo também a criação de novos e edição dos já existentes.

Esta ferramenta também apresenta algumas desvantagens. Comparado a outras ferramentas multi-plataforma, ainda apresenta algumas falhas no que se diz respeito ao desenvolvimento nativo. Para além disso, as aplicações desenvolvidas acabam por requerer uma maior capacidade de armazenamento, devido ao facto de usarem *built-in widgets* e não específicos de cada plataforma.

Optou-se assim por usar a ferramenta *Xamarin* para o desenvolvimento da componente móvel, uma vez que se destacou em relação às restantes pelo facto de ter uma percentagem maior de código partilhável. Também, providencia bibliotecas e *frameworks* que permitem aceder às componentes nativas, não sendo necessário desenvolver código específico para cada plataforma.

Na fase de escolha da tecnologia a usar na implementação da aplicação *Web* consideraram-se diferentes tipos relacionados com a linguagem de programação em *JavaScript*, uma vez que esta é uma linguagem frequentemente utilizada no desenvolvimento de aplicações *Web*. Após realizada a análise de mercado sobre as diferentes bibliotecas disponíveis, surgiram-nos duas mais usadas, nomeadamente *ReactJS* [11] e *Angular* [12].

Ambas cumprem o mesmo propósito sendo que a primeira se destaca na percentagem de utilização (cerca de 78,1% - com base num estudo realizado [13]) comparativamente como a segunda (cerca de 21,0% [13]).

Além disso, *ReactJS* faz uso do conceito de *virtual DOM* (*Document Object Model*) [11] que atualiza apenas as partes da página que sofreram alterações, em vez de realizar uma atualização total da mesma, o código de implementação de componentes é facilmente reutilizado e tem como principal objetivo ser uma biblioteca com rapidez de resposta, escalável e simples, podendo ser combinada com outras bibliotecas de *JavaScript* como inclusivamente o *Angular*.

Finalmente, *ReactJS* [11] é a única biblioteca que usa *JSX* (*JavaScript XML*) [13] que combina os *templates* da interface e a lógica *JavaScript* podendo ser uma grande vantagem para o desenvolvimento de aplicações.

Por outro lado, *Angular* [12] permite a criação de *user-interfaces* mais apelativas assim como a criação personalizada de *DOM's* e ainda permite uma implementação simples de rotas e ligação de dados *Angular* [14]. Uma das desvantagens dessa *framework* tem a ver com o facto de ter uma integração difícil com terceiros e ainda ter problemas de desempenho com elementos *DOM*.

Deste modo, com base no que foi descrito em cima, e evidenciado o facto de ser considerada uma das melhores bibliotecas *JavaScript* [14] de momento, tomou-se a decisão de usar *React* como ferramenta para o desenvolvimento da componente *Web*.

Para decidir qual a *framework* a utilizar para o desenvolvimento da componente servidora, foi realizada uma análise do conjunto de ferramentas existentes no mercado, sendo que nos surgiram duas com grande expressividade, nomeadamente *Django* [15] e *Spring Boot* [16].

Ambas as *frameworks* possuem uma boa reputação no mercado, porém *Django* detém um maior *ranking* como melhor *back-end web framework* (segundo lugar) comparativamente com *Spring* (décimo quinto) [17], sendo que quem prevalece em primeiro lugar é *Phoenix* [18].

Contudo, *Spring Boot* é conhecido por ter um excelente desempenho, requerer de uma preparação relativamente simples, estar baseado em *Java*, considerado o melhor no uso do modelo *Java MVC - Model View Controller* e de

rápido execução.

Decidiu-se assim, optar por *Spring Boot* para a implementação da componente servidora, uma vez que é considerado a melhor framework para a realização de uma *REST API*, comparativamente com *Django*.

É importante referir que a linguagem de programação selecionada para implementação da componente servidora foi *Kotlin* na medida em que possui interoperabilidade com Java executa sobre a *JVM - Java Virtual Machine* [19] e como tal recorre às bibliotecas da *JVM*.

Relativamente à *API* a usar para o acesso à base de dados, consideram-se três hipóteses, nomeadamente *JDBC*, *JDBI* ou *JPA*.

JDBC Java Database Connectivity [20] é flexível e revela ser opção equilibrada para aplicações não muito complexas, porém, é mais *verbose* no sentido em que a quantidade de código resultante para a execução de uma *query* é muito superior comparativamente com as outras duas opções. Além disso, tem um baixo desempenho, por necessitar de conversões das *queries SQL* para *ODBC Open Database Connectivity* [21].

Por outro lado *JPA Java Persistence API* [22], proporciona uma maior abstração da camada de acesso a dados, tendo como finalidade tirar alguma responsabilidade por parte do programador permitindo a este último concentrar-se apenas e só na lógica do servidor sem necessitar de lidar com problemas mais específicos relativos à base de dados. Contudo, esta *API* não revela ter um bom comportamento quando lhe são pedidas *queries* mais complexas e possui um menor desempenho comparativamente com a *JDBC* [23].

A *JDBI* é a solução mais equilibrada por ser uma solução intermédia entre as duas tecnologias apresentadas anteriormente permitindo ter a flexibilidade do *JDBC*, bem como alguma abstração da camada de acesso a dados do *JPA*, assim opta-se por esta ferramenta para a implementação do acesso aos dados da componente servidora.

4 Modelo de dados

Neste capítulo apresenta-se o modelo de dados do sistema proposto sendo que, numa primeira fase discute-se os diferentes modelos de concepção de uma base de dados, as suas vantagens e desvantagens, bem como o modelo optado e a fundamentação da opção tomada.

4.1 Análise e escolha do modelo de dados

Para que o sistema se possa comportar da maneira proposta, precisa, como peça fundamental na sua arquitetura, de uma base de dados. Atualmente existem dois modelos de desenho muito conhecidos, relacional e não relacional, e de seguida são expostas as definições dos modelos, vantagens e desvantagens e, finalmente, o modelo selecionado para a implementação do problema.

Um modelo relacional representa e armazena a informação na forma de tabelas e tuplos baseando-se na álgebra relacional (um dos ramos da teoria dos conjuntos algébricos) para o fazer. Ainda, recorre a SQL (*Structured Querying Language*) para poder controlar os dados (realizar *queries*, inserções e remoções de dados) o que, por um lado, se torna um aspecto bastante importante na medida em que é uma das opções mais versáteis e usadas no mercado, tornando a sua escolha mais segura. O facto de usar SQL torna a pesquisa de informação muito mais rápida e eficiente na medida em que a informação se encontra bem estruturada e organizada.

Sistemas *transaction-oriented* tais como plataformas *e-commerce* e *softwares* de contabilidade requerem muito deste modelo por essa razão.

Cada tuplo de uma tabela num modelo relacional representa uma entidade distinta sendo que cada coluna representa um atributo dessa entidade, tornando-se também uma boa escolha para quando é necessário respeitar a norma ACID. Esta norma é designada pelas iniciais **A**tomicidade, **C**onsistência, **I**solamento e **D**urabilidade sendo que cada uma representa as propriedades que se pretendem ver cumpridas. Atomicidade define que cada transação é executada completamente ou sofre de *rollback*; Consistência estabelece que a execução de uma transação deve levar o banco de dados de um estado consistente a um outro estado consistente; Isolamento determina que a execução de transações concorrentemente, são vistas como sendo sequências; Finalmente a Durabilidade estipula que ao realizar a operação de *commit* de uma transação, as alterações na base de dados são consideradas permanentes.

Por outro lado, um modelo relacional pode ser restritivo na medida em que é necessária uma preparação considerável na estruturação da base de dados, recorrendo a um *schema* para o fazer. Uma alteração na estrutura implica um enorme esforço para que todo o modelo de dados se torne novamente operacional. Ainda, o facto de recorrer a SQL abre a porta para a ocorrência de ataques do tipo *SQL Injection* sobre a base de dados.

Em termos de escalabilidade, um modelo relacional é escalável verticalmente (Figura 4.1), no sentido em que para suportar uma maior quantidade de dados, podem ser feitas melhorias no servidor como o aumento da capacidade de processamento (CPU), mais memória (RAM) e mais armazenamento (SSD). Esta é a única forma de escalar na medida em que as bases de dados relacionais foram desenhadas para correr num único servidor de forma a manter a integridade dos mapeamentos de tabelas e evitar a computação distribuída. Por consequência, uma base de dados relacional quando escalada não possui a flexibilidade para retroceder (*scale back down*) pelo facto os dados já estarem alocados no espaço adicionado e serem quase impossíveis de distribuir.

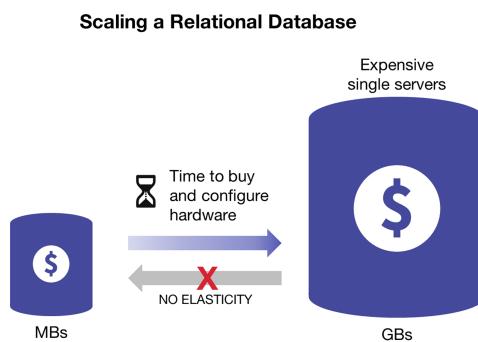


Figura 4.1: Escalamento Vertical [1]

Um modelo não-relacional é mais flexível na medida em que possui *schemas* dinâmicos para informação não estruturada podendo armazenar os dados sobre diferentes formas, *column-oriented*, *document-oriented*, baseado em grafo, entre outras. Desta forma, é possível criar documentos sem a necessidade da preparação estrutural que um modelo relacional implica e cada documento tem a sua própria estrutura tornando fácil a adaptação conforme o processo se torne mais complexo. No entanto, apenas armazena informação não recorrendo a mecanismos de ligação de dados a tabelas distintas como o modelo relacional. Em termos de escalabilidade, um modelo não relacional é escalável na horizontal (Figura 4.2) na medida em que conseguimos suportar mais tráfego com o aumento de servidores na base de dados, fazendo com que esta se possa tornar muito grande, tornando-se a escolha favorita para armazenamento de enormes quantidades de informação que mudam constantemente.

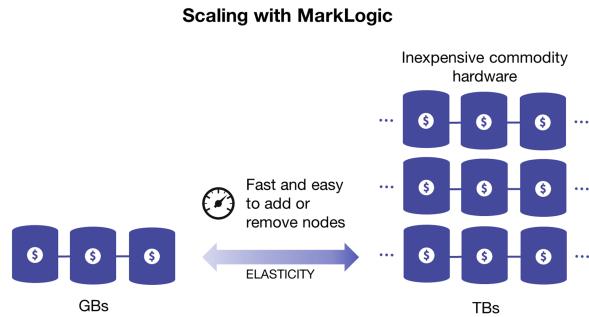


Figura 4.2: Escalamento Horizontal [1]

Considerando que o problema aqui apresentado necessita de um modelo de dados estruturado, por serem fortemente relacionados entre si, tomou-se a decisão de implementar a base de dados recorrendo ao modelo relacional. Para implementar o modelo de dados recorreu-se à variante da linguagem SQL *PostgreSQL* devido ao facto de ser uma linguagem amplamente usada no mercado, ser *open-source* permitindo a implementação da base de dados em outros ambientes sem ser *Microsoft* proporcionado-lhe maior flexibilidade. Ainda, o mecanismo da base de dados é conhecido por garantir o maior nível de atomicidade, consistência, isolamento e durabilidade (política ACID) e por ter um suporte considerado da comunidade.

4.2 Modelo Entidade-Associação

Para a concretização da base de dados foi necessária a concepção da sua arquitetura e das componentes que este iria conter.

O sistema só deve ser manipulado por intervenientes registados, assim implementa-se um sistema de utilizadores, com senha de acesso (*password*). Assim implementa-se a entidade utilizador *User*. Esta entidade tem como principal responsabilidade assegurar toda a informação relevante para a identificação do interveniente, nomeadamente um identificador, um nome, uma *password* (cifrada - usamos como encriptação uma função *hash SHA2.256*), uma fotografia, um contacto telefónico obrigatório e um contacto de email.

O identificador constitui ser o elemento mais importante desta entidade pelo facto de ser aquele que distingue cada utilizador. Um identificador é caracterizado por:

- 1 carácter - identifica o papel principal do utilizador dentro da organização
- 5 dígitos - representam o número do utilizador dentro da organização

Exemplo

Um identificador possível é o seguinte, A00001. Com esta representação estamos a dizer que o utilizador é um estudante e o seu número é o 00001. Inicialmente, cada utilizador apenas pode ter o conjunto de 3 caracteres, A (aluno), D (docente) e F (funcionário).

Ainda, a fotografia revela ser um atributo também ele importante para caracterizar o utilizador, uma vez que distingue cada utilizador através dos seus traços biométricos e acaba por contribuir para a rápida identificação do utilizador. É importante referir que, pelo facto do sistema possuir uma hierarquização de cargos, esta identificação não é de todo uniforme na medida em que um utilizador, independentemente de ter como identificador, por exemplo, D00001, este pode ter mais cargos dentro do sistema (responsável de laboratório, administrador, técnico, etc...).

A entidade papel *Role* representa um cargo, que um utilizador pode ter no sistema. Este cargo é um de um conjunto pré-definido de valores devidamente estabelecidos, nomeadamente: *i*) um administrador (cargo máximo); *ii*) um responsável de laboratório; *iii*) um técnico; *iv*) um docente; *v*) um funcionário; e finalmente *vi*) um estudante. Um utilizador que não detenha o cargo de estudante ou funcionário previamente estabelecido poderá ter atribuído mais do que um cargo, ou seja, um docente poderá ter mais do que uma função tal como responsável de laboratório e/ou administrador e/ou técnico.

Uma área representa um domínio onde as unidades curriculares (*UC's*) estão afetas, onde tipos de materiais estão afetos e onde um utilizador detentor de privilégios de responsável de laboratório pode inserir, remover e atualizar material. Quando o sistema é corrido pela primeira vez existe, por omissão, a área *geral* na medida em que inclui todos os materiais considerados avulso respetivamente.

Posteriormente, existe um conjunto de 5 entidades representativas de uma área, nomeadamente *arquitetura de computadores, sinais, redes e eletrónica*, estando de acordo com as já existentes na organização alvo onde se podem incluir unidades curriculares bem como tipos de material que sejam característicos dessa área.

A entidade *Subject* representa uma unidade curricular que é leccionada por um ou vários docentes, e onde estão afetados materiais. Cada unidade curricular precisa de estar associada a uma área sendo que é permitido que várias UC's possam ter nomes iguais, no entanto podem estar incluídas na mesma área.

Restringe-se a possibilidade de poder ter uma unidade curricular associada à área *geral* pelas razões fundamentadas na secção anterior.

Uma das entidades mais importantes do sistema, visto que representa o tipo de material. Cada entidade representativa de um tipo (*Type*) precisa dos seguintes atributos:

- identificador - peça fundamental para posterior definição do identificador de um material
- descrição - contém um resumo das funcionalidades deste tipo de material

Um aspeto relevante sobre esta entidade tem a ver com o identificador da mesma, na medida em que este é implementado por um modelo criado para o feito mencionado na secção seguinte. De forma geral é possível descrever o identificador de um tipo como a concatenação entre o identificador da área e o seu próprio identificador.

Será importante mencionar que um tipo precisa de estar afiliado a uma área e que o mesmo tipo pode estar incluído em várias unidades curriculares. De forma a exemplificar, na organização para o qual o problema é proposto, as unidades curriculares de LSD (Lógica de Sistemas Digitais) e AC (Arquitetura de Computadores), ambas incluídas na área de *arquitetura_computadores*, usam o material ATB também ele incluído nessa área.

Ainda outro exemplo, considerando as mesmas unidades curriculares, é o facto de ambas poderem recorrer a material avulso estando este último associado a uma área (*geral*) que não a mesma onde as UC's se incluem.

Outra das entidades fundamentais para a concretização do sistema é a entidade *Material*, na medida em que esta representa o recurso levantado pelos utilizadores. Um material, no contexto do problema, necessita de ser caracterizado por um identificador, um nome e um estado, podendo este último ter os valores *disponível* (o material pode ser levantado por um utilizador) ou *indisponível* (o material está a ser usado por um utilizador).

Um material unitário, para além de requerer estes atributos, também necessita de ter associados dois outros estados, nomeadamente o de "em reparação" para representar o material que se encontra danificado e que está em processo de reparação por parte do técnico e ainda o de "danificado" para representar materiais já verificados pelo técnico e que não possuem arranjo possível. De modo a garantir consistência do estado, só é possível ter um único estado para cada material.

Um material avulso, pelo facto de não possuir o carácter de unidade, precisa, além dos atributos referidos em cima, de um valor total e um valor disponível que representam a quantidade de material existente no sistema e, pelo que o nome indica, o material disponível para ser levantado e usado, respetivamente.

Um dos aspetos importantes desta entidade está sobretudo relacionado com a forma como um material é identificado no sistema a desenvolver. Pelo facto de se pretender que os materiais possam ser lidos com o auxílio de um *QR*

Code e, de forma a que o valor lido do mesmo possa ser o mais simples e organizado possível, desenvolveu-se um tipo específico de identificador, um pouco inspirado na análise ABC [24] sobre gestão de inventários. Desta forma, um identificador de um material é composto pelas três componentes, identificador de área, identificador de tipo e número do material tendo a seguinte formatação: *área - tipo - número*

Exemplo

Considerando um material do tipo ATB e, por consequência a área *arquitetura_computadores*, o identificador do primeiro material será 001-001-0001, o segundo 001-001-0002 e assim sucessivamente.

Relativamente aos materiais avulso, o identificador não é diferente daquele que é usado para o material unitário, apenas não se dá importância à ultima secção do mesmo e considera-se sempre esse valor como 0000, ou seja, um material cujo tipo seja, por exemplo, *chip* 001 e supondo que esteja identificado pelo tipo 000-001, o identificador final deste material será 000-001-0000 e tal secção irrelevante é estática.

A entidade *Request* constitui uma das peças essenciais para o sistema proposto, na medida em que é através dela que se associa um levantamento de material, a um utilizador. Uma requisição tem um identificador, uma data de início de levantamento, uma data de término do levantamento e ainda o tempo decorrido (em minutos) calculado após a entrega do material. Ainda, sobre o mesmo pedido, pode estar associado mais do que um material.

Uma avaria (*damage*) representa como a própria designação uma avaria de um material, num dado momento, e estas avarias são reportadas pelos utilizadores no sistema. Cada avaria precisa de ter um identificador, uma descrição da avaria, a data de submissão da avaria, a data de resolução, um relatório (breve descrição sobre o domínio do problema e a reparação aplicada) e finalmente, um estado que pode assumir o valor *resolvido*, *por resolver* e *sem solução*. Ainda, é possível realizar intenções de avaria, no sentido em que um estudante, aquando da requisição de um material, pode detetar anomalias e submeter uma descrição, criada pelo mesmo, que apresenta os possíveis erros que o aluno pensa que o material pode ter, sendo que mais tarde, na tentativa de entrega dos materiais por parte do aluno, a mesma intenção pode ser confirmada pelo funcionário, e transformada numa avaria, ou cancelada.

A entidade aula (*Class*) tem como principal responsabilidade representar uma reserva para uma aula laboratorial submetida por um docente num dado momento. Esta entidade é caracterizada por um identificador, uma data / hora de realização da aula laboratorial, o número de grupos que compõem a aula e ainda os materiais necessários para a realização da aula.

Aquando da submissão da intenção de uma aula laboratorial por parte de um docente esta é convertida em requisição caso se o próprio confirme realização. Também é importante referir que não é possível para o mesmo docente, realizar uma aula à mesma hora seja para a mesma unidade curricular, quer seja para unidades curriculares diferentes.

Na Figura 4.3, apresenta-se o modelo de entidade e associação (EA) para o sistema de informação aqui descrito. Ainda, é importante referir que na figura existem duas associações e uma entidade representadas em cores distintas do resto do modelo. Tal facto é justificado por se tratar da solução para um opcional relativo ao suporte para reservas de materiais no contexto do estudante.

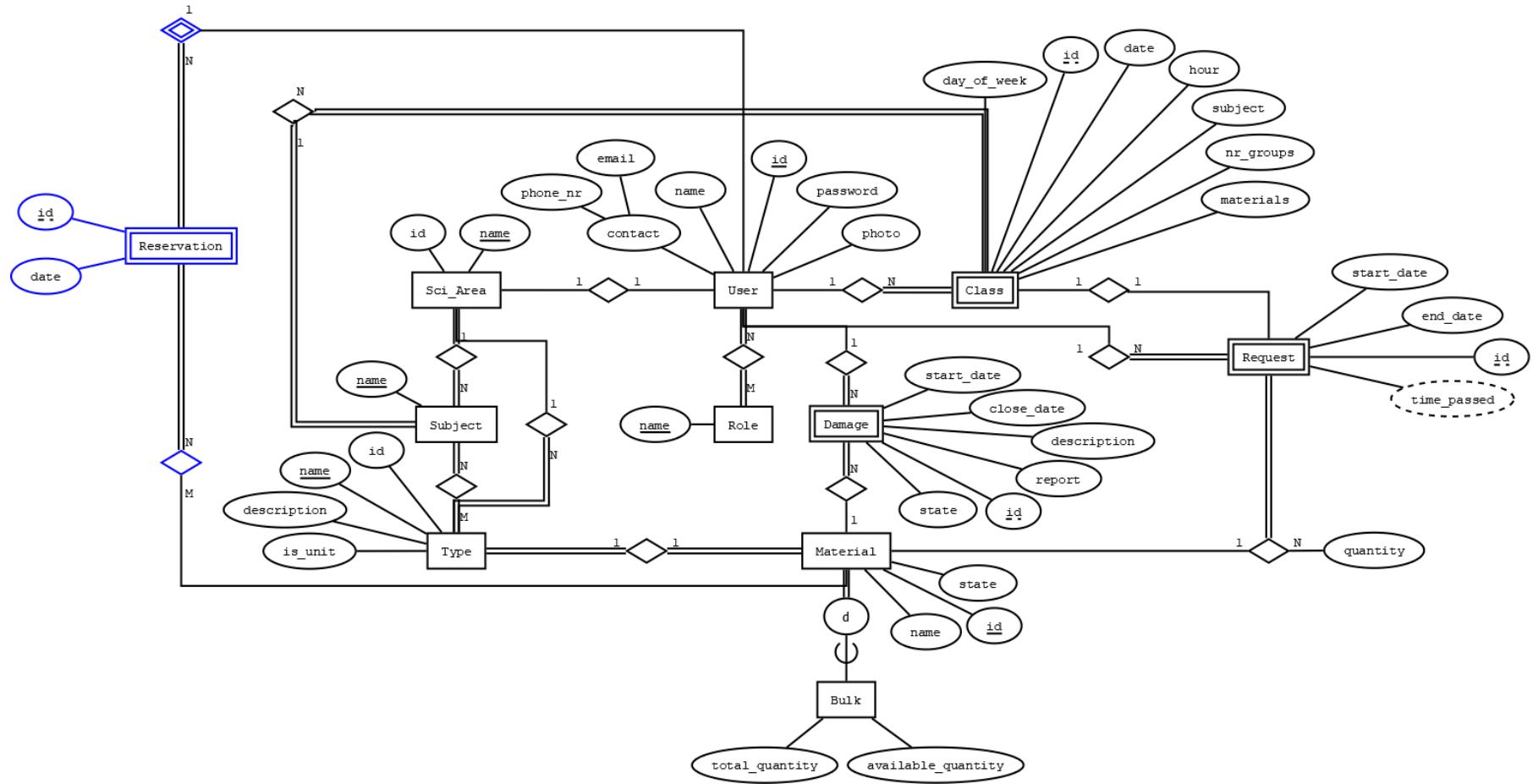


Figura 4.3: Modelo Entidade Associação

5 Detalhes de implementação

Numa primeira fase, foram concebidos os *mocks*, para cada uma das interfaces, de maneira a tornar a experiência do utilizador a melhor possível e facilitar o desenvolvimento da parte visual da aplicação. Estes *mocks* são apresentados nas secções 5.1 e 5.2.

5.1 Aplicação Móvel

Decidiu-se desenvolver uma aplicação móvel com o objetivo de garantir a mobilidade e a rapidez de utilização do sistema proposto.

A aplicação desenvolvida assegura duas interfaces, nomeadamente uma para o funcionário e uma para o aluno, dado que estes são os principais cargos que demonstram necessidade de características tais como mobilidade e rapidez de consulta e registo de material. Foram então definidas as funcionalidades que cada interface iria permitir, ilustradas no diagrama da Figura 5.1.

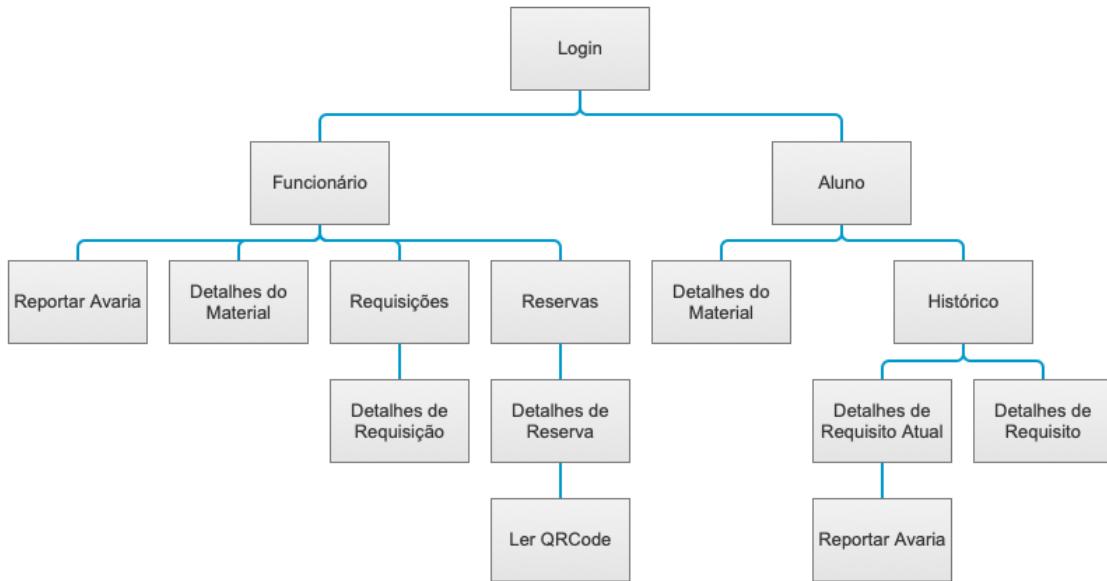


Figura 5.1: Diagrama de navegação da aplicação móvel

A primeira interface a ser desenhada, foi a do aluno, começando pela sua página inicial. A página inicial do aluno tem como objetivo permitir ao mesmo pesquisar material pelo seu tipo. Desta forma, o aluno poderá visualizar o material que se encontra disponível, por forma a observar se pode fazer uma requisição do material que deseja utilizar. Ao serem apresentados os materiais, se um for selecionado, será possível visualizar os respetivos detalhes. Na página principal, é também possível pressionar o botão "Read QR Code" que permite ler o *QR Code* de um material, como é demonstrado na Figura 5.2b. Se o mesmo se encontrar na sua posse, o aluno tem a possibilidade de escolher reportar uma avaria do mesmo ou ver os seus detalhes, caso contrário, o aluno é reencaminhado para os detalhes do material lido.

Na página que apresenta os detalhes de um material, é possível ver informações sobre o mesmo, tais como o seu nome, tipo, descrição, a área científica a que pertence, as disciplinas em que é utilizado, o seu estado, e, em casos de material não unitário, a quantidade que ainda se encontra disponível para requisitar.

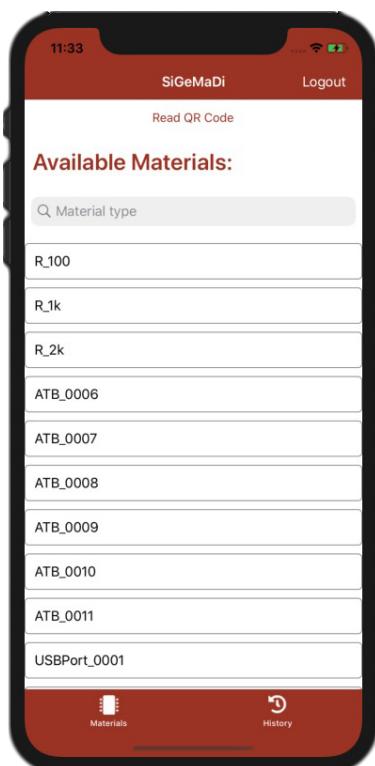
Caso o utilizador, neste caso o aluno, deseje consultar as requisições anteriores com o objetivo de ver material que já esteve na sua posse, tem a possibilidade de selecionar a opção "History", como é possível observar na Figura 5.3a. Na página de detalhes da requisição, é possível observar os materiais registados, assim como o dia e a hora em que a requisição foi realizada, ilustrada na Figura 5.3b.

Ao contrário da página de detalhes de uma requisição antiga, a página de detalhes da requisição atual permite reportar avarias sobre um determinado material, dado que um aluno pode verificar a existência de uma anomalia enquanto o material está na sua posse. Assim, é possível reportar essa avaria, através de um texto descrevendo-a.

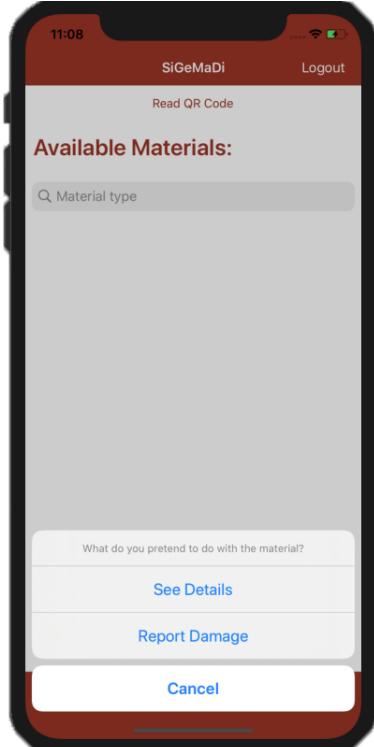
Na página principal do funcionário, é proporcionada a possibilidade de pesquisar material pelo seu tipo. É também possível filtrar o material pela sua disciplina, pela sua área e pelo seu tipo, como é demonstrado na Figura 5.4a.



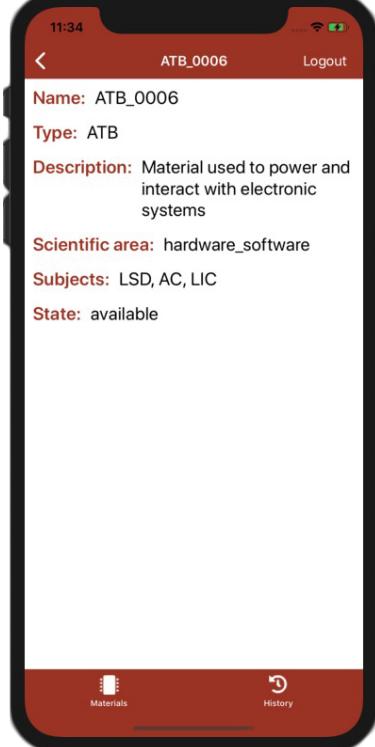
(a) Página de Login



(b) Página principal do aluno

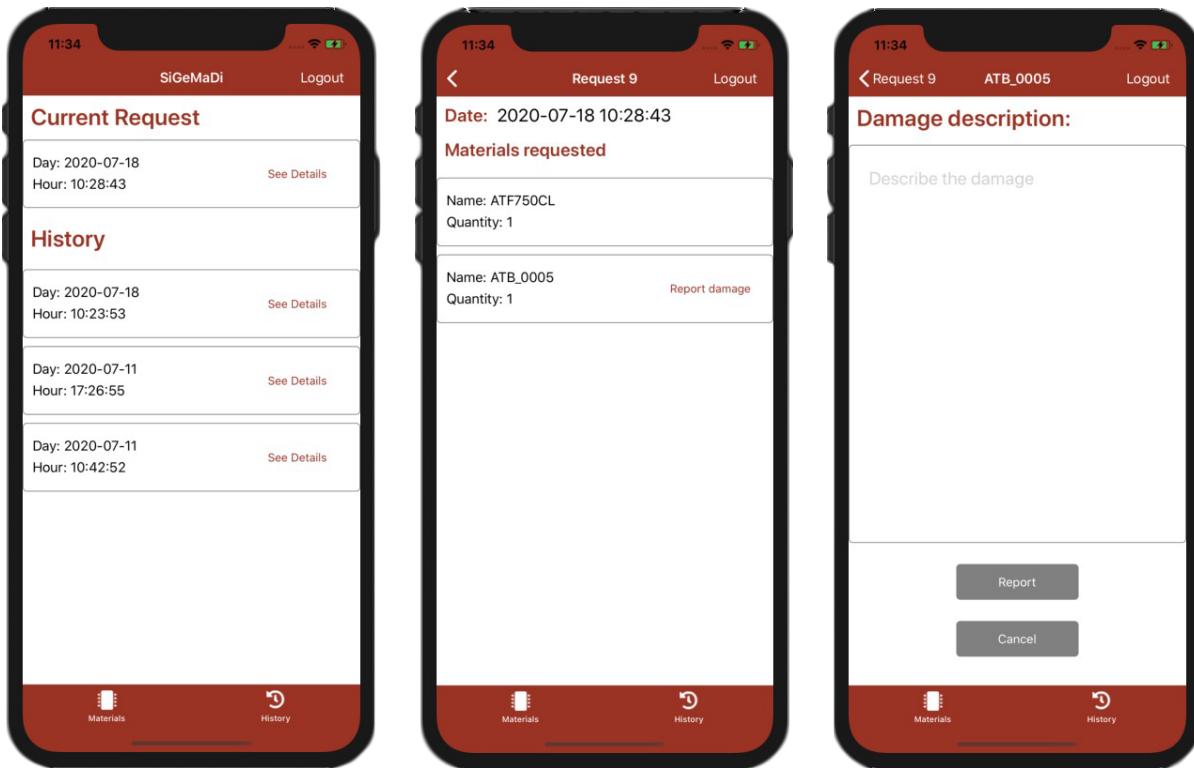


(c) Página principal do aluno



(d) Detalhes de um material

Figura 5.2: *Mocks* da aplicação móvel - aluno

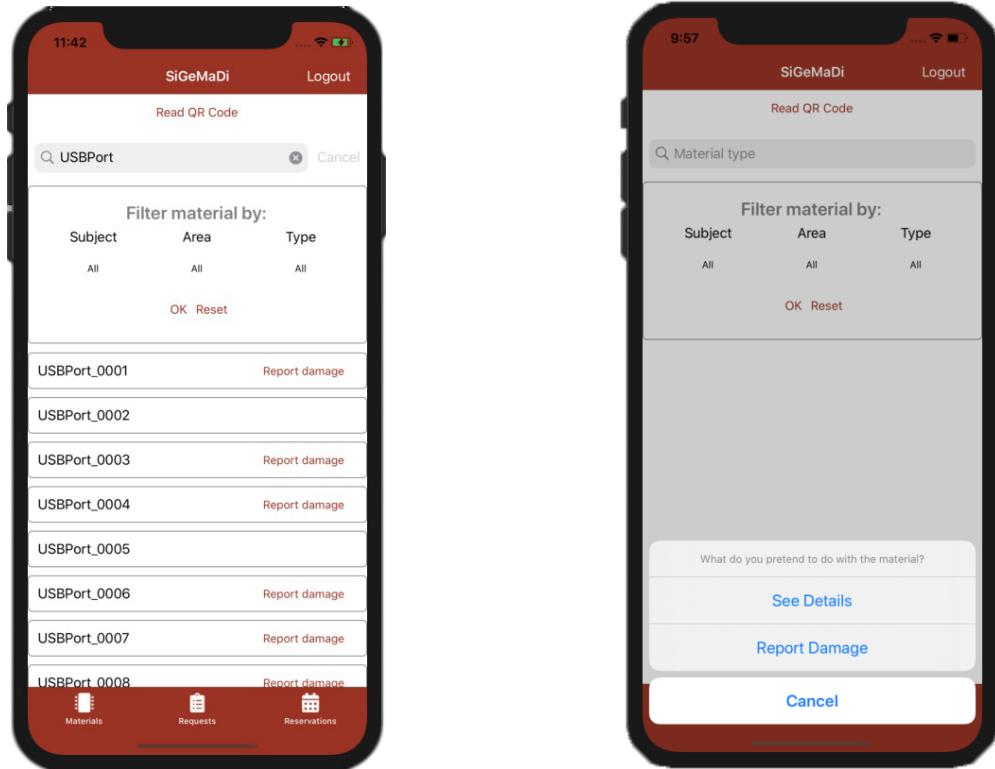


(a) Histórico de um aluno

(b) Detalhes de uma requisição

(c) Reportar Avaria

Figura 5.3: *Mocks* da aplicação móvel - aluno



(a) Página principal do funcionário

(b) Página principal do funcionário

Figura 5.4: *Mocks* da aplicação móvel - funcionário

Quando a lista dos materiais é apresentada, é possível selecioná-lo, indo assim para uma página que contém os detalhes sobre o mesmo, ou reportar uma avaria sobre um determinado material caso este não se encontre requisitado. As páginas descritas têm a mesma interface que as demonstradas para a interface do aluno, ver Figuras 5.2d e 5.3c.

Nesta página é também possível visualizar um botão com o texto "Read QR Code" que permite a leitura do *QR Code* de materiais. Caso seja possível reportar uma avaria sobre o material lido, é apresentada uma página igual à demonstrada na Figura 5.2c na interface de aluno, caso contrário, o funcionário é reencaminhado para os detalhes do material.

Caso o funcionário selecione "Requests", este é reencaminhado para uma nova página, representada pela Figura 5.5a, tendo esta como principal objetivo consultar as requisições atuais, mostrando o dia e a hora em que estas começaram, assim como o utilizador que tem material na sua posse. É também possível observar uma *ComboBox*, a qual permite filtrar as requisições de forma a ver apenas as ativas, as terminadas ou todas como é possível visualizar na Figura 5.5b. Há também um botão que indica o dia em que estas requisições ocorreram, e se este for pressionado é apresentado um calendário para escolher um outro dia representado pela Figura 5.5c. Ao selecionar uma das requisições, é possível ver detalhes sobre a mesma.

Na Figura 5.5d é apresentada a página que lista os detalhes de uma requisição na interface do funcionário. Nesta, é possível ver informações sobre a requisição selecionada na página representada pela Figura 5.5a, tal como o nome, o número, e a foto do utilizador que fez a requisição, o dia e a hora em que este foi registado, e o material requisitado. Uma das funcionalidades que a aplicação móvel fornece também ao funcionário, é a possibilidade do mesmo visualizar as reservas, ou seja, poder visualizar material que um docente deseja requisitar, para uma dada hora num determinado dia, com o objetivo de o utilizar numa aula prática. É apresentada então uma lista de todas as reservas, caso o utilizador selecione "Reservations", onde é possível observar o docente que realizou a mesma, a data em que o material é necessário e a disciplina em que vai ser usado.

Caso o funcionário selecione uma das reservas apresentadas na Figura 5.6a, este tem a possibilidade de observar os seus detalhes, sendo possível visualizar as informações da unidade curricular da aula a ser lecionada, o nome e o identificador do docente que irá realizar a aula, o número de grupos presentes na mesma, o dia e a hora em que a esta se irá realizar, e o material necessário por cada grupo.

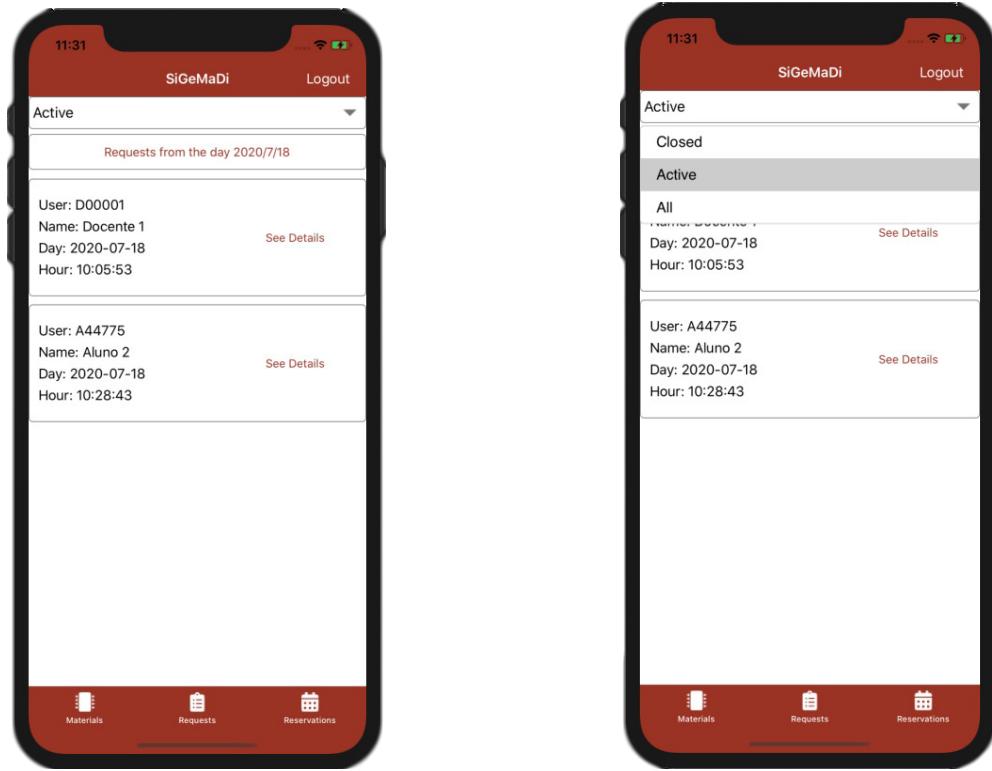
Na Figura 5.6b, é possível visualizar quatro botões com diferentes funções, nomeadamente, "Notify students", que tem como objetivo notificar alunos que tenham material na sua posse necessário para a realização de uma aula, caso não haja material suficiente, através do envio de um SMS. Quando este botão é pressionado, o funcionário tem de especificar o tipo de material que necessita, assim como a quantidade em falta do mesmo. "Request material", que, ao ser pressionado, o utilizador é reencaminhado para uma página que permite ler o *QR Code* do material que deseja requisitar para a aula, sendo este depois acrescentado à lista de materiais requisitados. Caso haja algum lapso, é possível eliminar o material requisitado desta lista. "Notified students", que tem como função fornecer ao funcionário a possibilidade de visualizar os alunos que receberam a notificação(SMS ou e-mail), como representado na Figura 5.6c. "Save request", quando selecionado, termina a reserva, comutando-a para uma requisição. Assim, todos os materiais que se encontram na lista de materiais requisitados, são registados, ficando a requisição em nome do docente que irá realizar a aula prática.

5.1.1 Detalhes de implementação da aplicação móvel

Para realizar-se a leitura de *QR Code*, *Xamarin* [5] oferece inúmeras bibliotecas que providenciam recursos para que a leitura seja possível, destacando-se assim duas bibliotecas, nomeadamente, *ZXing* [25], *ZBar* [26], e uma *API*, sendo esta *Mobile Vision API* [27].

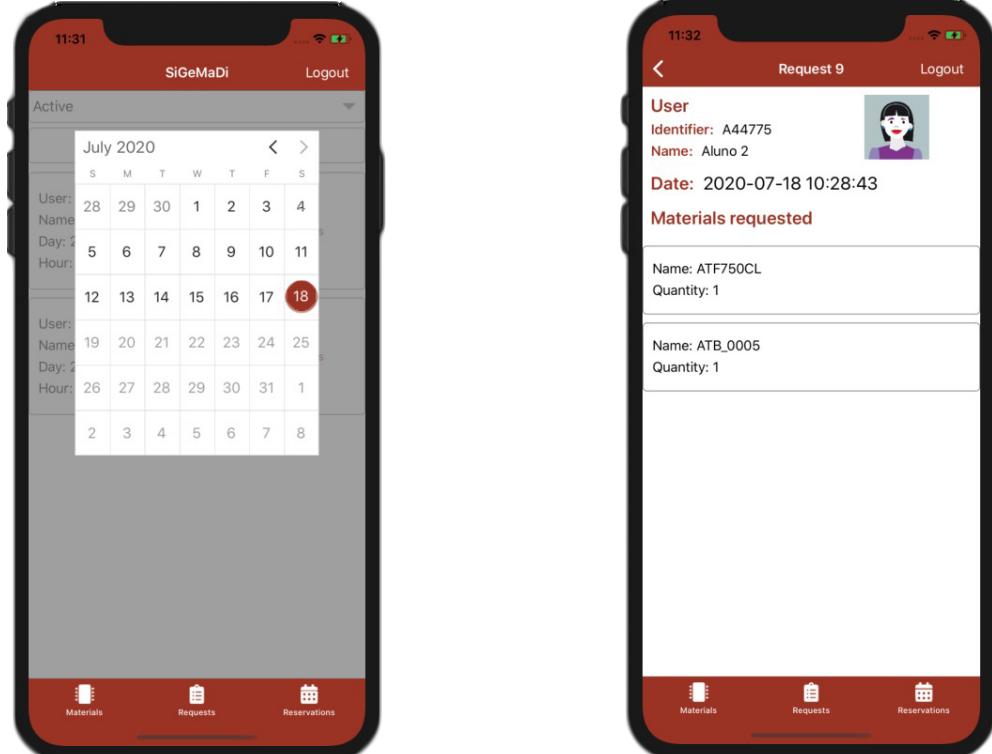
ZXing ("zebra crossing") é uma biblioteca *open-source* implementada em *Java*, podendo esta ser interligada a outras linguagens, que permite processar imagens com código de barras. Esta biblioteca é a mais popular das três listadas, e apresenta a vantagem de devido à sua popularidade, ser possível encontrar bastante documentação sobre a mesma. Para além desta vantagem, o *scan* do *QR Code* pode ser feito em qualquer posição do dispositivo, a leitura é feita numa *thread* em *background*, permitindo que a sua inicialização seja bastante rápida, e a sua interface é parametrizável. Esta biblioteca também apresenta a desvantagem de, por vezes, apresentar dificuldades a realizar a leitura de maiores *QR Codes*. Dado que os nossos *QR Codes* são pequenos, contendo apenas doze caracteres, esta biblioteca não irá apresentar este problema.

ZBar é uma biblioteca *open-source* que permite a leitura de códigos de barras vindos de várias fontes, tais como vídeos, imagens e sensores. Esta biblioteca tem a principal vantagem de ser bastante rápida, sendo, por vezes, possível ler o *QR Code* antes do utilizador estabilizar o dispositivo e focar a câmara, e é bastante fácil de implementar. Apesar do facto da leitura ser rápida parecer ser algo bastante vantajoso, pode gerar problemas. Por vezes esta leitura é



(a) Requisições

(b) Requisições



(c) Requisições

(d) Detalhes de uma requisição

Figura 5.5: *Mocks* da aplicação móvel - funcionário

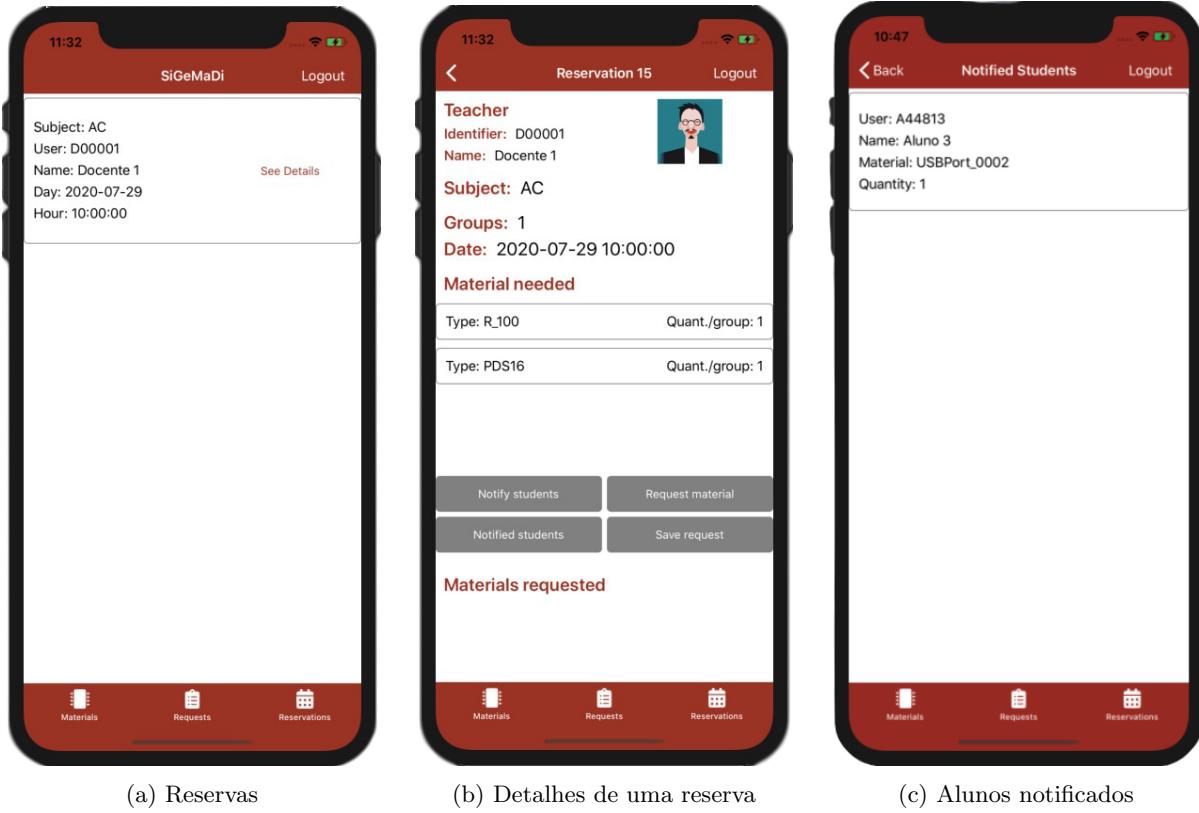


Figura 5.6: *Mocks* da aplicação móvel - funcionário

demasiado rápida, o que pode levar à deteção de códigos errados. Também, não existe muita documentação sobre esta biblioteca.

Mobile Vision API providencia uma *framework* que inclui detetores que permitem localizar e descrever objetos visuais em imagens e vídeos, como caras, códigos de barras e texto. Esta *API* tem duas grandes vantagens, ser possível realizar a leitura do código estando o dispositivo em qualquer posição, e a leitura ser bastante rápida. No entanto, para ser possível realizar a leitura do *QR Code* é necessário uma biblioteca nativa, a qual é preciso fazer o *download* para o dispositivo. Isto acaba por ser uma desvantagem dado que não é possível integrar esta biblioteca na aplicação. É, também, difícil ler o *QR Code* e focar a câmara em alguns dispositivos.

Depois da análise das bibliotecas e *APIs* existentes, optou-se pela biblioteca *ZXing* para se realizar a leitura do *QR Code*, pela sua vasta documentação, a interface ser parametrizável e não ser necessário fazer-se *download* de bibliotecas complementares para ser possível a leitura, fazendo desta a mais adequada para fornecer uma melhor experiência ao utilizador.

Para ser possível apresentar a *ComboBox* apresentada na Figura 5.5b, e o calendário da Figura 5.5c, foram utilizados componentes UI providenciados pela empresa *Syncfusion* [28]. Para ter acesso a estes componentes, foi necessário obter uma licença [29] que nos permite a utilização destes componentes de forma gratuita.

Em algumas páginas da aplicação móvel, é apresentada uma lista de itens, tais como os materiais (Figura 5.4a), as requisições (Figura 5.5a) ou reservas (Figura 5.6a). Dado o facto de, por vezes, a quantidade de itens ser bastante grande, é usada paginação para os obter.

Para ser possível apresentar todos estes itens, a lista tem de ser carregada à medida que o utilizador a visualiza, fazendo-se assim vários pedidos à *Web API*. De maneira a isto ser possível, foi implementada uma lista que usa *Infinite Scroll* nas páginas que requeriam esta possibilidade, dando assim a oportunidade ao utilizador de visualizar todos os itens desejados. A *framework Xamarin.Forms* [30], providenciada pela ferramenta *Xamarin* [5], contém um *plugin* que possibilita a implementação de *Infinite Scroll*, nomeadamente *Xamarin.Forms.Extended.InfiniteScrolling* [31].

Quando um utilizador realiza alterações enquanto mexe na aplicação, estas devem ficar armazenadas mesmo quando

a aplicação é fechada. Para resolver este problema, é necessário armazenar os dados resultantes destas alterações na aplicação. *SQLite* [32] é um mecanismo de base de dados que permite aplicações feitas com recurso a *Xamarin.Forms* [30] carregar e guardar dados em código partilhado. Este mecanismo foi utilizado sempre que foi necessário armazenar dados na aplicação. Um exemplo da utilização de *SQLite* encontra-se na página representada na Figura 5.6b, onde é possível visualizar os detalhes de uma reserva. Se o funcionário ler o *QR Code* de vários materiais com o objetivo de os requisitar para a aula, e de seguida fechar a aplicação, é usado *SQLite* para armazenar na base de dados local os materiais já lidos, de maneira a que o funcionário não tenha de ler os códigos novamente.

Para além deste mecanismo, foi também utilizada uma classe chamada *Secure Storage* [33], que é fornecida pela API multi-plataforma *Xamarin.Essentials* [34] que tem como função ajudar a armazenar pares chave/valor na aplicação. Esta classe foi utilizada sempre que foi necessário armazenar dados de forma segura, como por exemplo *tokens*, visto serem pequenos pedaços de texto, para o qual esta classe foi desenvolvida.

5.2 Aplicação Web

Decidiu-se realizar uma aplicação *Web* para englobar as diferentes ações dos diferentes tipos de utilizadores, alojar a lógica e a manutenção do sistema, como é habitual em sistemas de gestão. A aplicação assegura o acesso aos seus utilizadores através dos cargos que os mesmos desempenham, tendo diferentes ações para cada um, possibilitando que um utilizador possua vários cargos, podendo ser alterados a qualquer momento pelo administrador. Assim, a manutenção do sistema está a cargo do administrador e a lógica distribuída hierarquicamente pelos restantes cargos.

Os cargos considerados neste sistema, e anteriormente enumerados na Secção 4.2. são: Administrador, Responsável de Laboratório, Funcionário, Técnico, Docente e Aluno. Nesta aplicação o único cargo que não tem acesso é o de Aluno, uma vez que a interface está mais direcionada para a aplicação móvel.

Na implementação dos *mocks* da aplicação *Web* começou-se por especificar as diferentes páginas que esta constitui através do diagrama da Figura 5.7.

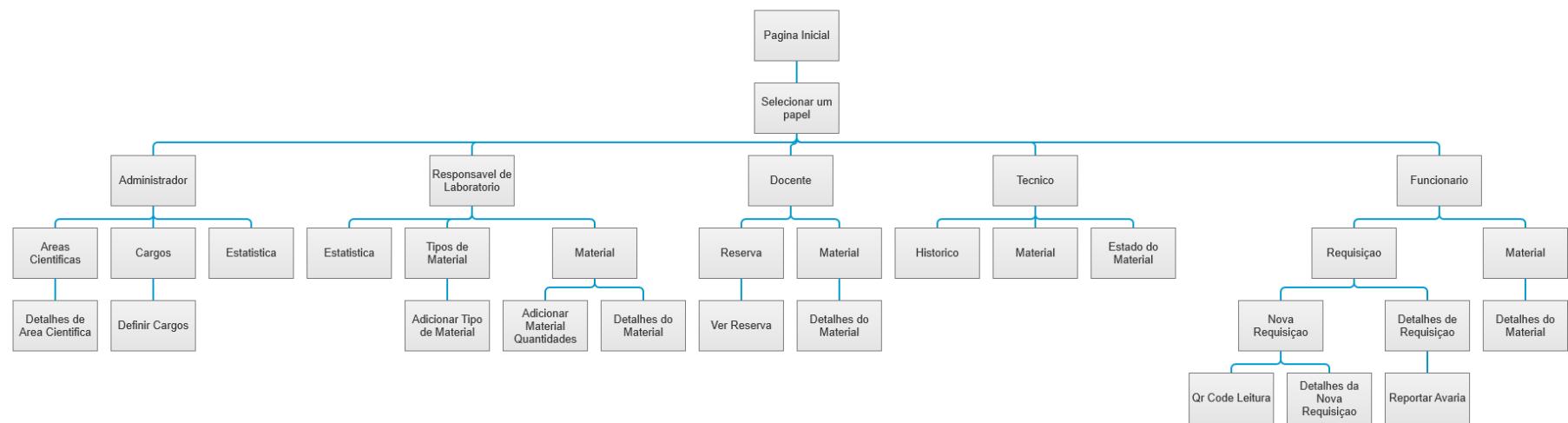


Figura 5.7: Diagrama de Navegação da Aplicação Web

A primeira página que o utilizador encontra é a página inicial da aplicação *Web* que apresenta o logótipo do sistema, bem como o logótipo da instituição, e as suas instalações.



Figura 5.8: Página Principal

Ao selecionar o botão "Login" é apresentada a página de login, que quando realizado com sucesso é redirecionado para a página de escolha de cargo a desempenhar na sessão, como ilustrado nas Figuras 5.9 e 5.10 respetivamente. Os cargos apresentados serão os mesmos que cada utilizador tem associado.

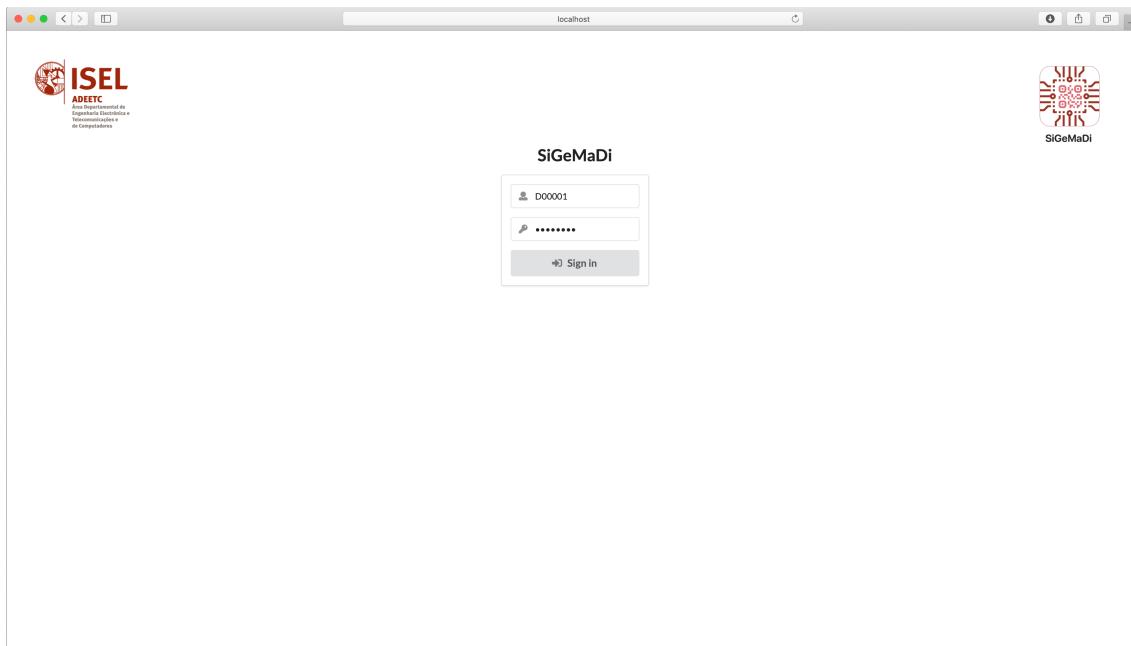


Figura 5.9: Início de Sessão

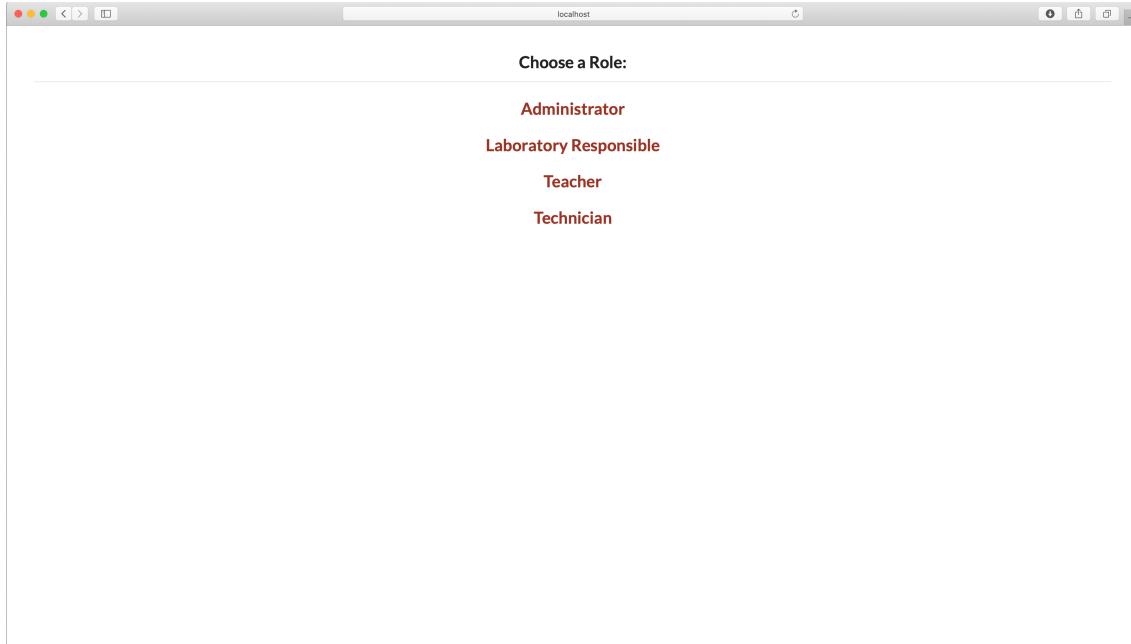


Figura 5.10: Seleção de Cargo

Dependendo do cargo que o utilizador selecionar na sessão, este é redirecionado para a página inicial do mesmo. No caso de ser selecionado Administrador a página apresentada contém uma barra de navegação com as diferentes ações possíveis para este cargo. A página principal do Administrador está representada pela Figura 5.11.

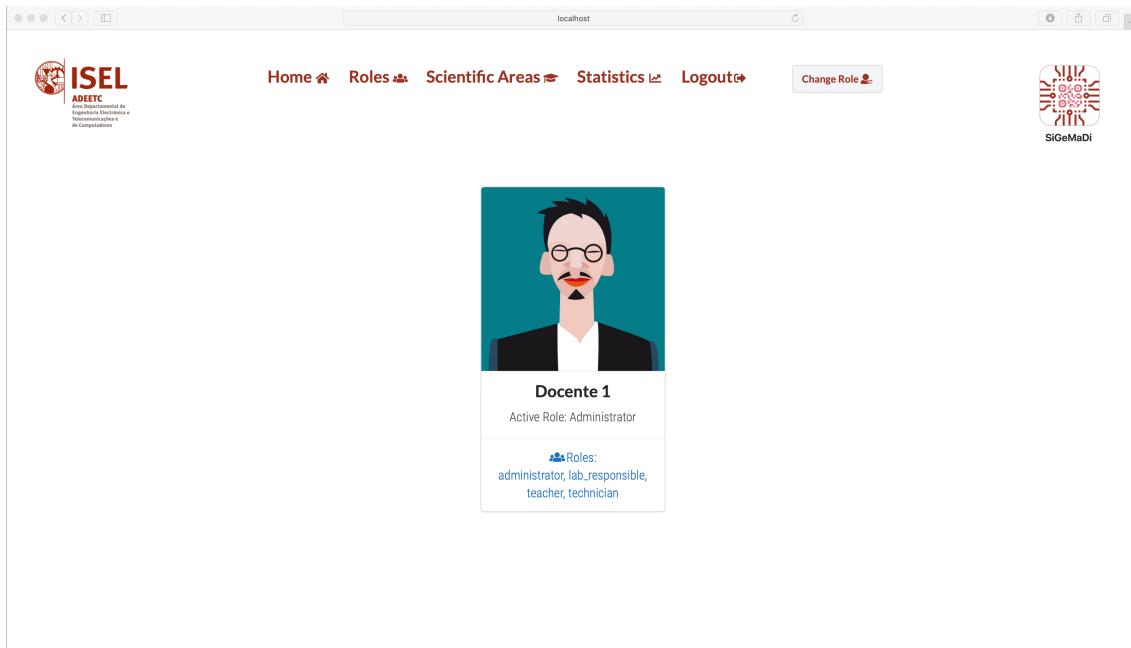


Figura 5.11: Página Principal do Administrador

Ao selecionar “Roles” a página representada na Figura 5.12 é apresentada, e é possível visualizar todos os utilizadores do sistema, filtrando-os por cargo ou ainda por número de utilizador. Cada item tem um botão que quando selecionado é feito o redireccionamento para a página que representa uma listagem de cargos para o determinado utilizador, visível na Figura 5.13, podendo selecionar os pretendidos. Os cargos que este já detém estão pré-selecionados. Assim, conclui-se a atribuição de cargos a utilizadores do sistema.

The screenshot shows a web application interface for managing users. At the top, there is a header with the ISEL logo, navigation links for Home, Roles, Scientific Areas, Statistics, and Logout, and a 'Change Role' button. On the right side of the header is the SigeMaDi logo. Below the header, the page title is 'Users:' and there is a search bar with a placeholder 'introduce user number...'. There is also a filter dropdown set to 'All'. Two user profiles are displayed: 'Docente 1' (D00001) and 'Docente 2' (D00002). Each profile includes a small portrait, the name, ID, and a 'Change Roles' button.

Figura 5.12: Utilizadores

The screenshot shows a modal dialog for assigning roles to 'Docente 1' (D00001). The left side of the dialog lists available roles: administrator, lab_responsible, teacher, and technician, each with a checkbox. The 'lab_responsible' and 'hardware_software' checkboxes are checked. On the right side, there is a preview area showing the user's portrait and the assigned role 'Docente 1'. A green '✓ - Confirm' button is located at the top right of the modal.

Figura 5.13: Definir Cargos

É ainda possível ao Administrador do sistema adicionar ou eliminar áreas científicas, adicionando a cada uma, unidades curriculares, representado nas Figuras 5.14 e 5.15 respetivamente, e ainda visualizar várias estatísticas relativas ao material, interface representada na Figura 5.16. Desta forma, estas ações são representadas pelas seguintes páginas ao selecionar os últimos botões da barra de navegação, finalizando todas as ações possíveis para o cargo de Administrador.

The screenshot shows a web browser window with the URL 'localhost'. At the top, there is a navigation bar with links: 'Home', 'Roles', 'Scientific Areas', 'Statistics', and 'Logout'. On the left, the ISEL ADEETC logo is displayed. On the right, there is a 'Change Role' button and the SIGEMADI logo. The main content area is titled 'Scientific Areas:' and contains a search bar labeled 'Scientific Area...' and a 'Add' button. Below the search bar is a table listing four scientific areas: 'hardware_software', 'redes', 'sinais', and 'eletronica', each with a red 'X' icon to its right.

Figura 5.14: Áreas Científicas

The screenshot shows a web browser window with the URL 'localhost'. At the top, there is a navigation bar with links: 'Home', 'Roles', 'Scientific Areas', 'Statistics', and 'Logout'. On the left, the ISEL ADEETC logo is displayed. On the right, there is a 'Change Role' button and the SIGEMADI logo. The main content area is titled 'Scientific Area: hardware_software' and contains a search bar labeled 'Subjects...' and a 'Add' button. Below the search bar is a table listing three subjects: 'LSD', 'AC', and 'LIC', each with a red 'X' icon to its right.

Figura 5.15: Unidades Curriculares de uma Área Científica

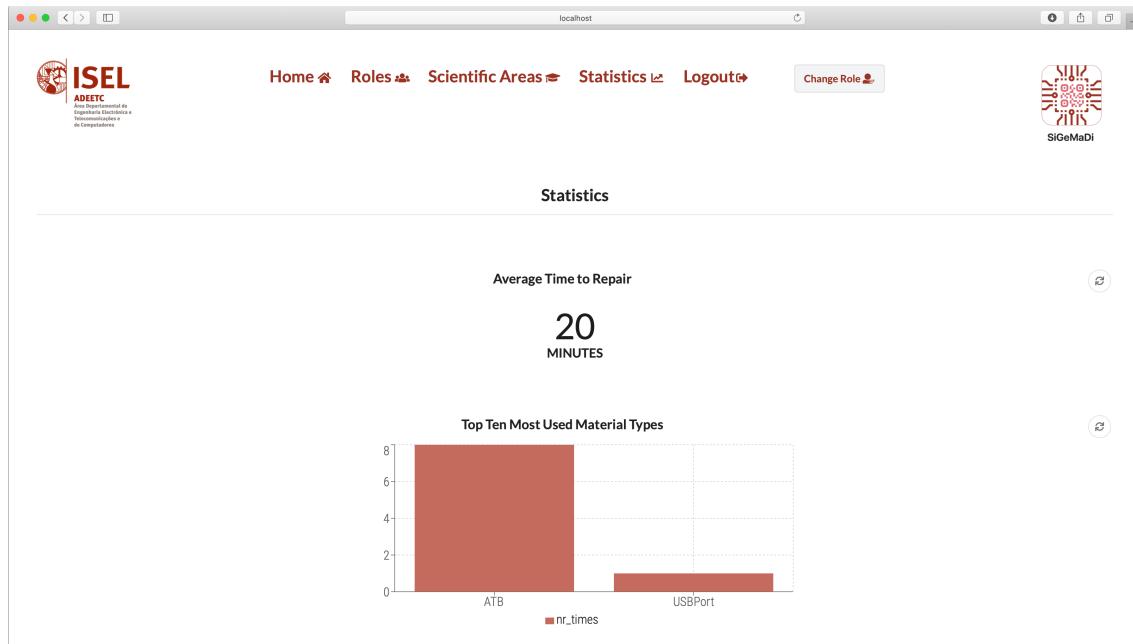


Figura 5.16: Estatísticas

O Responsável de Laboratório tem, à semelhança do cargo anterior, uma página principal representada na Figura 5.17. É possível, partilhando a opção com o Administrador, visualizar estatísticas ao selecionar "Statistics", representada na Figura 5.16, e ainda gerir o material didático presente no sistema. O Responsável de Laboratório ao selecionar "Material" na barra de navegação, conseguirá visualizar os diferentes materiais, filtrando pelas várias características associadas a cada um, nomeadamente a área científica, como mostra a Figura 5.18.

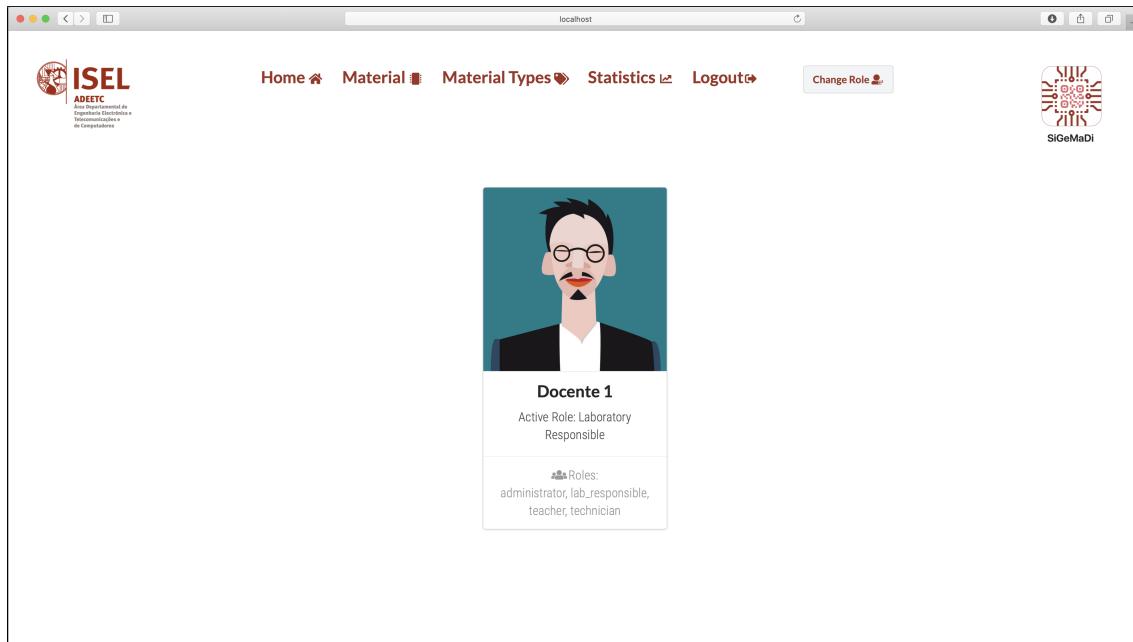


Figura 5.17: Página Principal do Responsável de Laboratório

The screenshot shows a web-based application for managing materials. At the top, there are navigation links for Home, Material, Material Types, Statistics, and Logout, along with a Change Role button. On the right side, there is a logo for SIGEMADI. The main area is titled "Material:" and contains a grid of material items. Each item has a small thumbnail, a unique identifier, and a creation date. Below each item are two buttons: "Details" (grey) and "Delete Material" (red). A "Filter by Scientific Areas" section is at the top left, and a "Filter by Subjects" section is at the top right.

Name	Identifier	Creation Date	Action Buttons
R_100	000-002-0000	01-001-0001	[Details] [Delete Material]
R_1k	000-003-0000	01-001-0002	[Details] [Delete Material]
R_2k	000-004-0000	01-001-0003	[Details] [Delete Material]
ATB_0003	001-001-0003	01-001-0004	[Details] [Delete Material]
ATB_0005	001-001-0005	01-001-0005	[Details] [Delete Material]
ATB_0006	001-001-0006	01-001-0006	[Details] [Delete Material]
ATB_0007	001-001-0007	01-001-0007	[Details] [Delete Material]
ATB_0008	001-001-0008	01-001-0008	[Details] [Delete Material]
ATB_0009	001-001-0009	01-001-0009	[Details] [Delete Material]
ATB_0010	001-001-0010	01-001-0010	[Details] [Delete Material]
ATB_0011	001-001-0011	01-001-0011	[Details] [Delete Material]
USBPort_0001	001-002-0001	01-002-0001	[Details] [Delete Material]

Figura 5.18: Gestão de Materiais

Ao selecionar o botão de detalhe de um item da lista, é apresentada uma página com a informação detalhada de um material como ilustrado na Figura 5.19, o botão vermelho representado em cada material, quando selecionado permite eliminá-lo do sistema. Caso o material seja avulso, é apresentado uma mensagem com um campo para introduzir a quantidade pretendida a eliminar, como representado na Figura 5.20.

The screenshot shows a detailed view of a material. At the top, there are navigation links for Home, Material, Material Types, Statistics, and Logout, along with a Change Role button. On the right side, there is a logo for SIGEMADI. The main area is titled "Material Details" and contains a form with the following fields:

- Name:** ATB_0010
- Type:** ATB
- Scientific Areas:** hardware_software
- Subjects:** LSD, AC, LIC
- Description:** Material used to power and interact with electronic systems
- More Information:** (Empty)

A green button labeled "State: available" is visible above the form. The status bar at the bottom indicates the URL as localhost.

Figura 5.19: Detalhes do Material

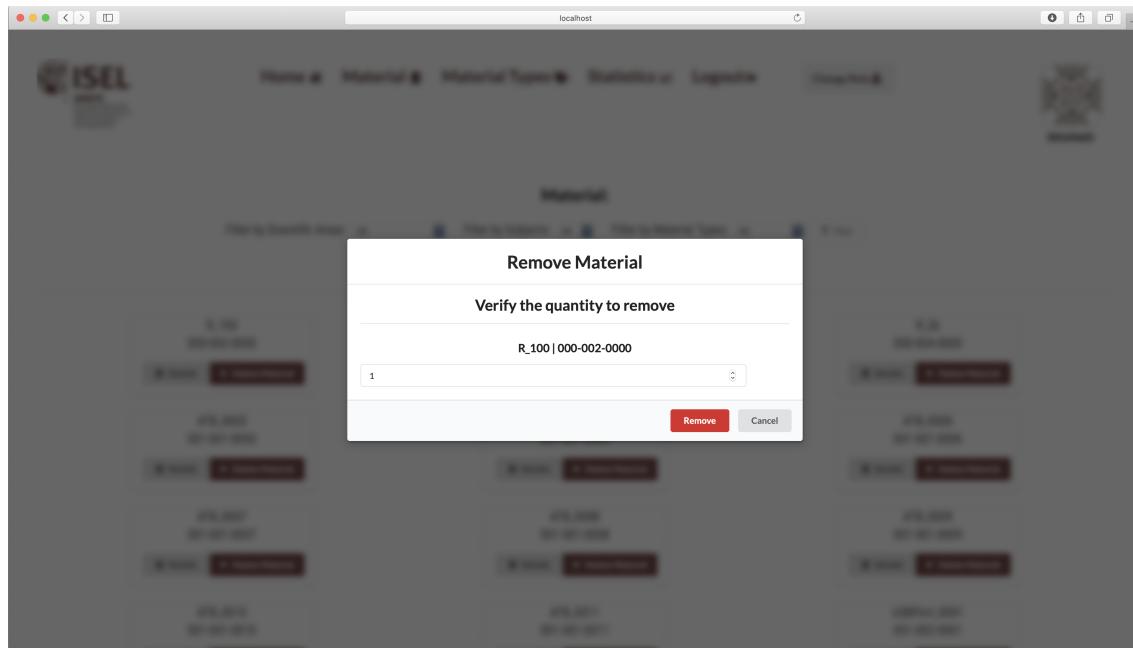


Figura 5.20: Eliminar Material Avulso

Após selecionar ”Add Material” na página de listagem de material, é possível adicionar um material, associando-o à área científica do responsável em questão, um tipo, e ainda quantidade, podendo ainda escolher a área geral, que engloba os materiais avulso. No caso de ser um material avulso, a sua representação na lista de materiais não será extensa, ou seja é apenas representado por um item, ao contrário dos materiais unitários que são representados por um item na lista, por cada material do mesmo tipo.

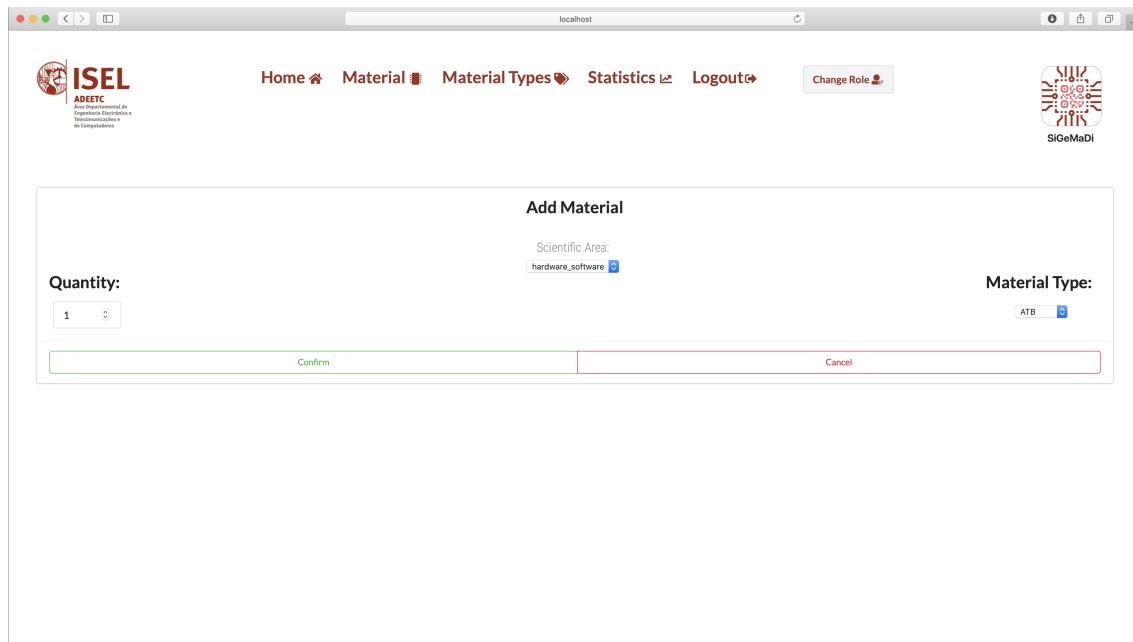


Figura 5.21: Adicionar Materiais

Na barra de navegação é ainda apresentada a opção ”Material Types”, que representa os diferentes tipos de material para a área científica que o responsável de laboratório se insere, assim como a área geral.

The screenshot shows a web application interface for managing material types. At the top, there is a navigation bar with links for Home, Material, Material Types, Statistics, Logout, and Change Role. On the right side of the header, there is a logo for 'SIGEMADI' with a QR code.

The main content area is titled 'Material Type:' and contains a table with the following data:

Material Type	Action
ATB	
USBPort	
PDS16	
ATF750CL	
R_100	
R_1k	
R_2k	

Figura 5.22: Tipos de Materiais

É de notar que é apresentado um botão "Add Type", que redireciona para a página de adição de um novo tipo, representada pela Figura 5.23, onde é especificado a área científica, a unidade curricular, e ainda uma breve descrição.

The screenshot shows a form for adding a new material type. The top navigation bar is identical to Figure 5.22. The main form is titled 'Add Material Type'.

Fields in the form include:

- Scientific Area:** A dropdown menu currently set to 'hardware_software'.
- Subjects:** Three checkboxes labeled 'LSD', 'AC', and 'LIC'.
- New Material Type:** A section containing two text input fields: 'Name:' and 'Description:'.

At the bottom of the form, there are 'Confirm' and 'Cancel' buttons.

Figura 5.23: Adicionar Tipos de Materiais

A interface de Funcionário permite as funcionalidades de requisição de material, reporte de avarias, e consulta de material. Na página principal do Funcionário, representada pelas Figuras 5.24 e 5.25, é apresentada informação sobre o mesmo, e ainda ações rápidas, nomeadamente, a leitura de *QR Code*, que após feita a sua leitura é redirecionado para a página de detalhe do material. Na primeira ação da barra de navegação, é possível realizar a consulta de material no sistema representada sob a forma de uma lista, podendo filtrar por área científica, unidade curricular ou tipo de material. Quando selecionado o botão de detalhe de um item presente na lista, é apresentada a informação detalhada do material, nomeadamente, o nome, o tipo, a área científica e a unidade a que pertence, a sua disponibilidade, uma breve descrição e ainda informação adicional, representado pela Figura 5.26. Ao selecionar o botão de reportar

é apresentado um formulário, com a descrição da avaria para preencher como ilustrado na Figura 5.27.

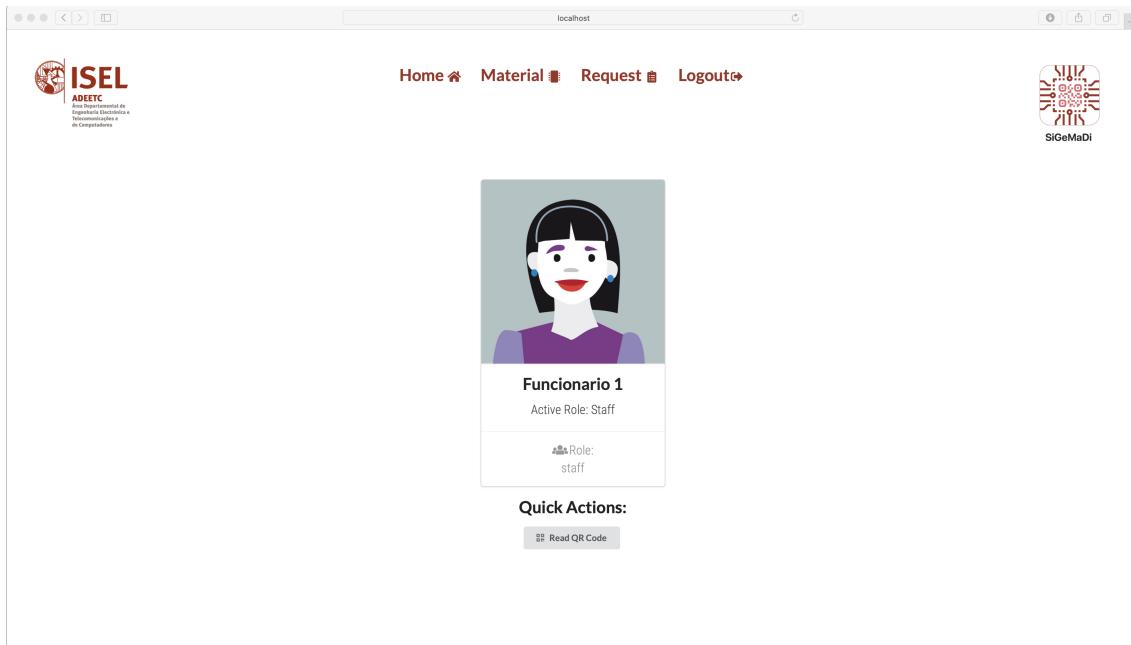


Figura 5.24: Página Principal do Funcionário

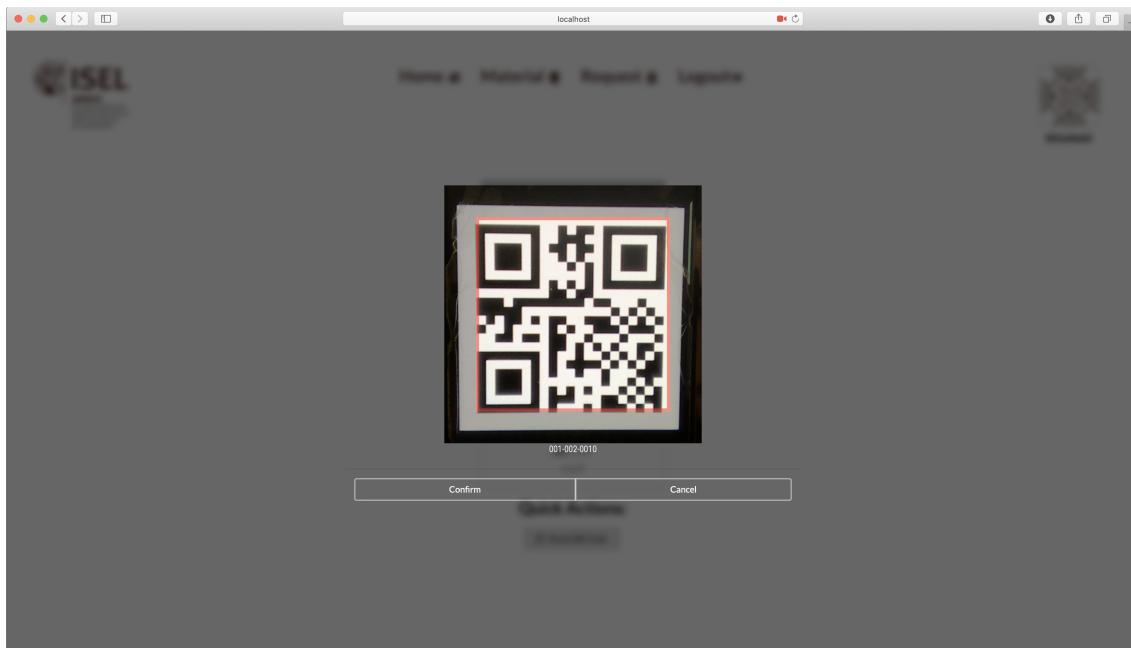


Figura 5.25: Ações rápidas

The screenshot shows the Material management system's homepage. At the top, there are navigation links: Home, Material, Request, and Logout. On the left, the ISEL ADEETC logo is displayed. On the right, the SiGeMaDi logo is shown. Below the navigation, the word "Material:" is centered. There are three rows of boxes, each containing a material item with its name, ID, and two buttons: "Details" and "Report".

Material Name	ID	Action Buttons
ATF750CL	000-001-0000	[Details] [Report]
R_100	000-002-0000	[Details]
R_1k	000-003-0000	[Details]
R_2k	000-004-0000	[Details]
ATB_0001	001-001-0001	[Details]
ATB_0002	001-001-0002	[Details]
ATB_0003	001-001-0003	[Details] [Report]
ATB_0004	001-001-0004	[Details]
ATB_0005	001-001-0005	[Details] [Report]
ATB_0006	001-001-0006	[Details] [Report]
ATB_0007	001-001-0007	[Details] [Report]
ATB_0008	001-001-0008	[Details] [Report]

Figura 5.26: Material

The screenshot shows the "Material Details" page. At the top, there are navigation links: Home, Material, Request, and Logout. On the left, the ISEL ADEETC logo is displayed. On the right, the SiGeMaDi logo is shown. The main section is titled "Material Details". It contains a form with fields: Name (filled with "USBPort"), Type (filled with "USBPort"), Scientific Areas (filled with "hardware_software"), Subjects (filled with "LIC"), and a "State" button set to "available". Below the form, there are two sections: "Description" (containing the text "Material used as bridge between software and hardware") and "More Information".

Figura 5.27: Detalhes do Material

The screenshot shows a web browser window with the ISEL ADEETC logo at the top left. The navigation bar includes links for Home, Material, Request, and Logout. On the right side, there is a QR code and the text "SIGeMaDi". The main content area is titled "Report Damage:" and contains a form with the identifier "001-002-0010". Below the identifier is a red placeholder text "Report Description" followed by a text input field containing "Fios estragados". At the bottom of the form are two buttons: "Confirm" (green) and "Cancel" (red).

Figura 5.28: Reportar Avaria

Nesta interface é possível realizar o reporte de uma avaria de um material, sendo o Funcionário o único com esta função, no entanto esta funcionalidade tem restrições, nomeadamente o facto de que o material tem que estar disponível, entenda-se disponível como o estado onde o material não se encontra requisitado, ou com avaria. Caso o material esteja requisitado, o aluno é o único com a possibilidade de criar uma submissão de reporte de avaria, ou seja, o aluno tenciona reportar uma avaria, e o funcionário confirma ou cancela a sua submissão. Assim é garantido um maior controlo sob as avarias.

Por fim, a última opção presente na barra de navegação é precisamente "Request". Aqui o funcionário consegue visualizar todas as requisições dos diferentes utilizadores, podendo ainda filtrar pelo seu identificador, data de criação ou pelo estado da requisição, e ainda criar uma nova requisição através do botão "New Request".

The screenshot shows a web browser window with the ISEL ADEETC logo at the top left. The navigation bar includes links for Home, Material, Request, and Logout. On the right side, there is a QR code and the text "SIGeMaDi". The main content area is titled "Requests:" and features a grid of four user profiles. Each profile includes a small portrait, a number (1, 2, 3, or 4), the user's name and ID (e.g., "Aluno 1 | A44783"), the creation date (e.g., "2020-07-11"), and a status indicator (e.g., "Ended"). Below each profile is a "See Request" button. There are also filters for "Filter by Type: All" and "Filter By Day: Data", and a search bar.

User ID	User Name	ID	Date	Status
1	Aluno 1	A44783	2020-07-11	See Request
2	Aluno 3	A44813	2020-07-11	See Request
3	Aluno 2	A44775	2020-07-11	Ended
4	Aluno 2	A44775	2020-07-11	Ended

Figura 5.29: Requisições

Ao selecionar um item da lista são apresentados os detalhes da requisição, onde é possível, caso exista, confirmar ou

cancelar reportes de avarias de cada material, como apresentado na Figura 5.30.

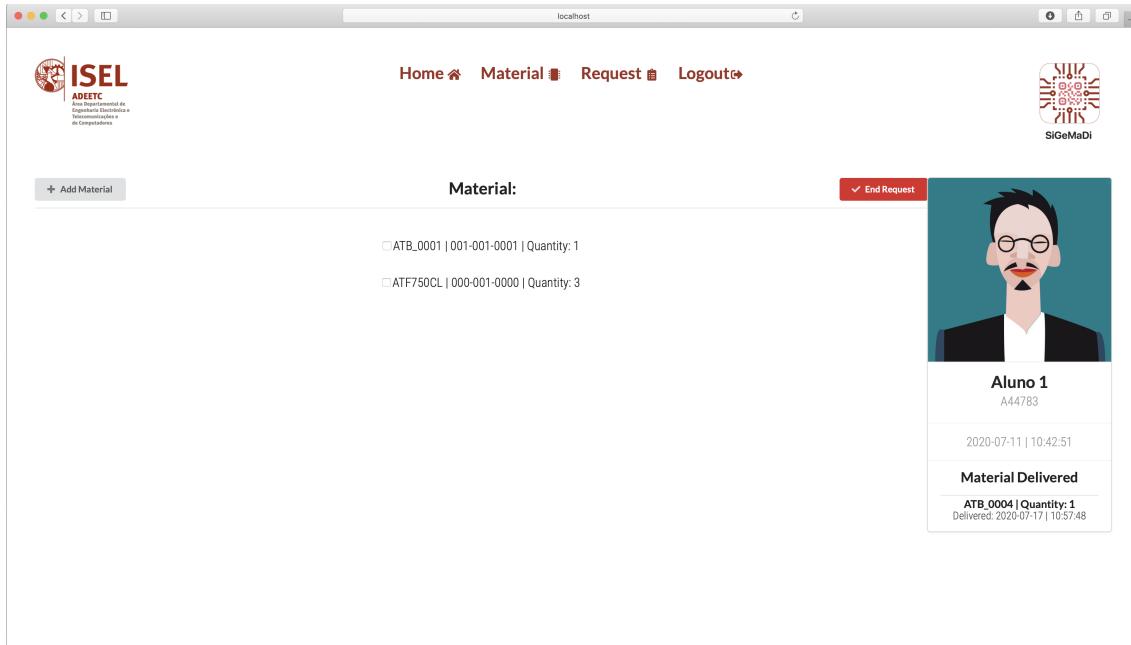


Figura 5.30: Detalhes da Requisição

A página representada na Figura 5.30, apresenta tanto os materiais que ainda permanecem na requisição, como também os já entregues. Assim que forem entregues todos os materiais a requisição é fechada, e é apenas apresentada a lista de materiais que estavam presentes na requisição, bem como as datas de inicio e de fim. Como é possível visualizar, nesta interface ao serem selecionados materiais e de seguida o botão "Remove Material", é apresentada uma mensagem de confirmação com um formulário para indicar a quantidade, no caso dos materiais avulso, de seguida estes são retirados da requisição, como representada na Figura 5.31. Caso se pretenda terminar a requisição por completo, não se seleciona materiais, mas o botão "End Request", que faz aparecer uma mensagem de confirmação, como ilustrado na Figura 5.32. É ainda possível ao selecionar o botão "Add Material", adicionar material à requisição, caso esta esteja ativa.

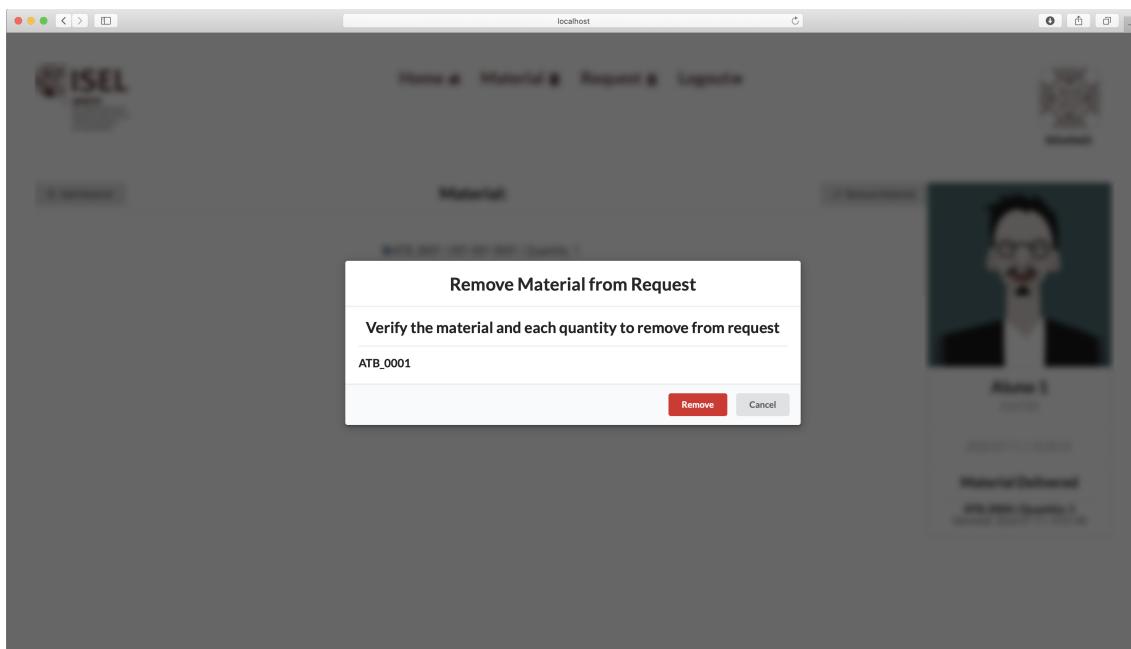


Figura 5.31: Remover Materiais da Requisição

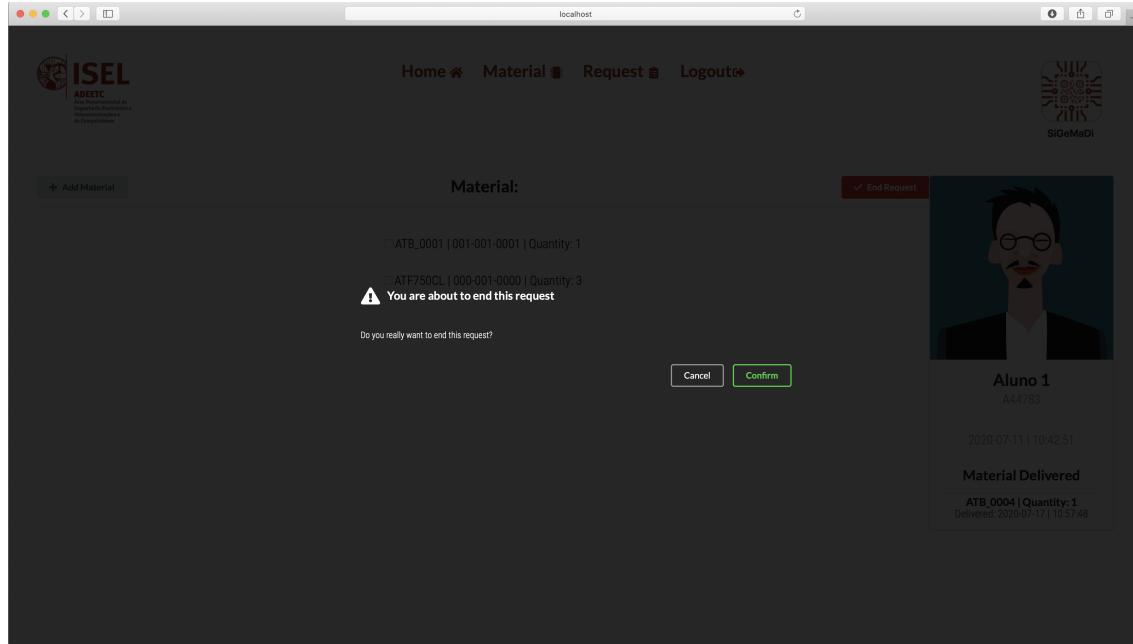


Figura 5.32: Terminar Requisição

Quando o funcionário tenciona criar uma nova requisição, será apresentada uma página com a listagem dos diferentes materiais disponíveis, representada na Figura 5.34. O funcionário seleciona os materiais pretendidos, podendo fazer uma pesquisa através da filtragem pelos atributos associados a cada material, ou ainda, através da leitura do *QR Code* do material, como é possível observar na Figura 5.33.



Figura 5.33: Leitura do *QR Code*

Material	Description	Status
R_100 000-002-0000	Select	
R_1k 000-003-0000	Select	
R_2k 000-004-0000	Select	Selected!
ATB_0003 001-001-0003	Select	
ATB_0004 001-001-0004	Select	
ATB_0005 001-001-0005	Select	
ATB_0006 001-001-0006	Select	
ATB_0007 001-001-0007	Select	
ATB_0008 001-001-0008	Select	
ATB_0009 001-001-0009	Select	
ATB_0010 001-001-0010	Select	
ATB_0011 001-001-0011	Select	

Figura 5.34: Nova Requisição

Quando o botão ”Add Material” é selecionado, é feito o redirecionamento para a página de conclusão da requisição, onde é especificado o utilizador ao qual fica associado, e, por fim, a quantidade apenas para os materiais não unitários (por exemplo: cabos, chips), quando selecionados.

Figura 5.35: Finalizar a Nova Requisição

O cargo que se segue é o de Técnico, a pessoa que faz o controlo das avarias reportadas no sistema, e à semelhança dos cargos anteriores, também este tem uma página principal como mostra a Figura 5.36.

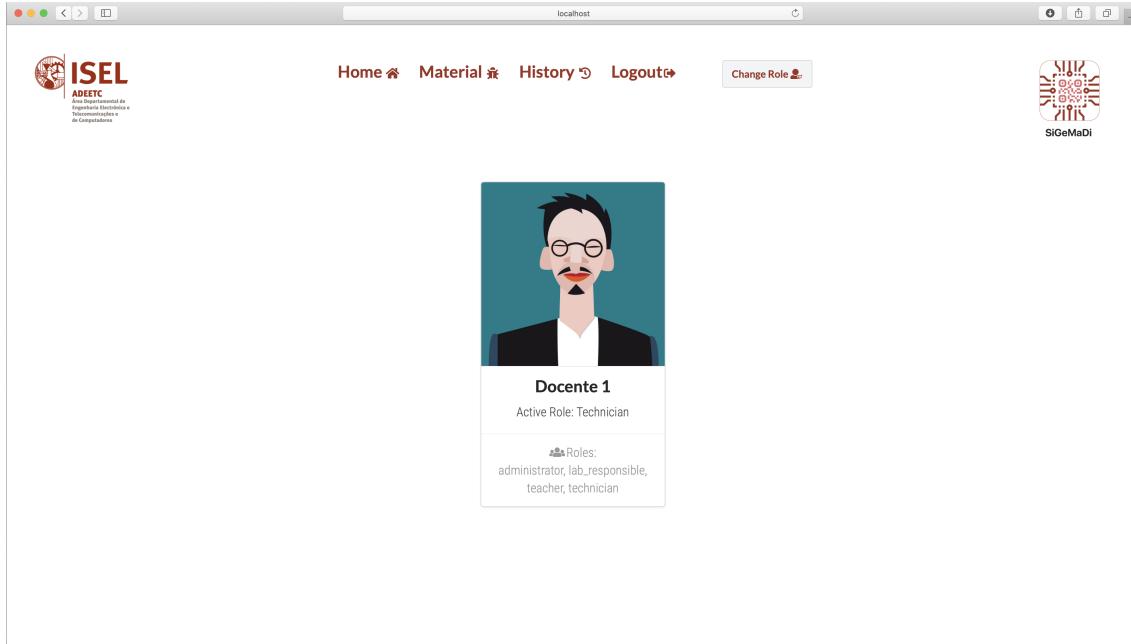


Figura 5.36: Página Principal do Técnico

A visualização das avarias ativas é realizada ao selecionar "Material", onde é apresentada uma lista de reporte de avarias, com um botão para aceder ao detalhe da avaria, onde é possível controlar o estado da avaria, como representado nas Figuras 5.37 e 5.38. O controlo de estado das avarias é realizado através de uma breve descrição da solução do problema e ainda da alteração do estado de cada avaria, nomeadamente, por resolver, resolvido ou avariado, sendo que este último estado significa que o material não está em condições de voltar a ser requisitado. As avarias que passam ao estado de resolvido, libertam o material para que possa ser utilizado para uma requisição, caso passem ao estado avariado, o material não está funcional, logo não é apresentado aquando a seleção de materiais para requisição, à semelhança quando o seu estado é por resolver.

Para visualizar o histórico de avarias seleciona-se "History", de seguida é representada também uma lista com as avarias ocorridas no sistema, ou seja é apresentado qualquer tipo de avaria que não esteja com o estado por resolver, como representada na Figura 5.39. Ao selecionar o botão de detalhe, é redirecionado para os detalhes da avaria, à semelhança com o caso anterior, como ilustrado na Figura 5.40.

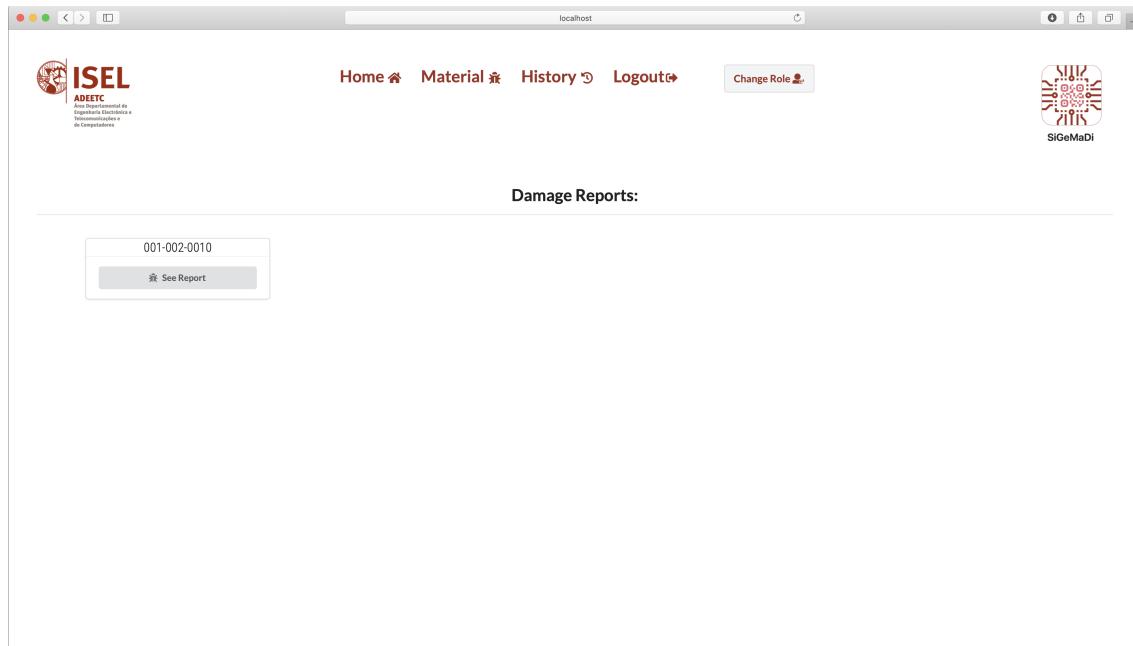


Figura 5.37: Avarias

The screenshot shows a web browser window with the URL 'localhost'. At the top left is the ISEL ADEETC logo. To its right are navigation links: 'Home', 'Material', 'History', and 'Logout'. On the far right is a 'Change Role' button and the SiGeMaDi logo. Below the header, the title 'Material Details' is centered. A red button labeled 'State: to_solve' is present. Above a text input field, the placeholder 'Change Report State: to_solve' is shown. To the right of the state fields, the report details are listed: '001-002-0010', 'USBPort_0010', 'F00001', and '2020-07-17'. Below these details are two large text input fields: 'Report Description' containing 'Fios estragados' and 'Solution Description'. At the bottom of the form is a horizontal bar with 'Confirm' and 'Cancel' buttons.

Figura 5.38: Detalhes da Avaria

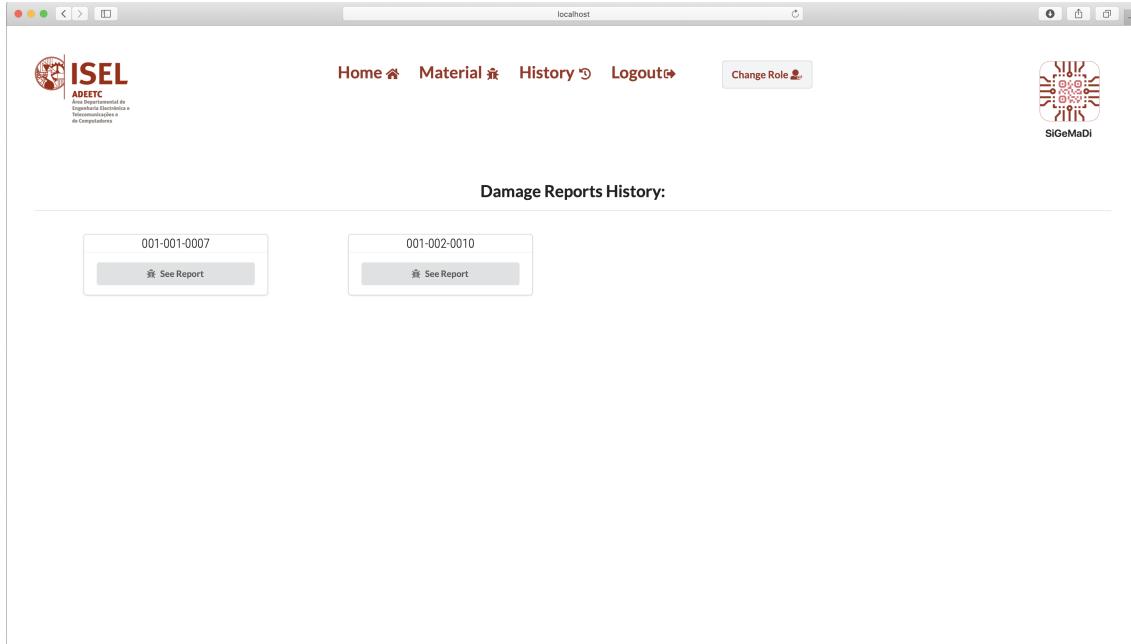


Figura 5.39: Histórico de Avarias

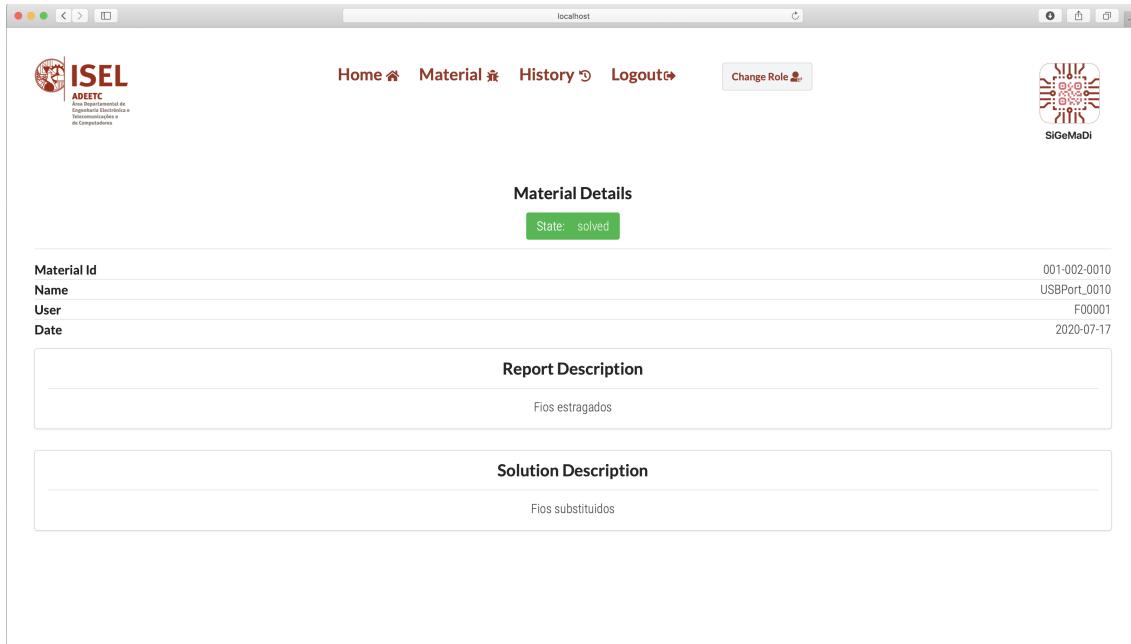


Figura 5.40: Detalhe de Avaria Passada

Por fim, a última interface é a de Docente, este tem a possibilidade de reservar material para as aulas e, como já é comum, visualizar a sua informação pessoal na página principal, e ainda o material presente no sistema e os seus detalhes, como representado nas Figuras 5.41, 5.42 e 5.43.

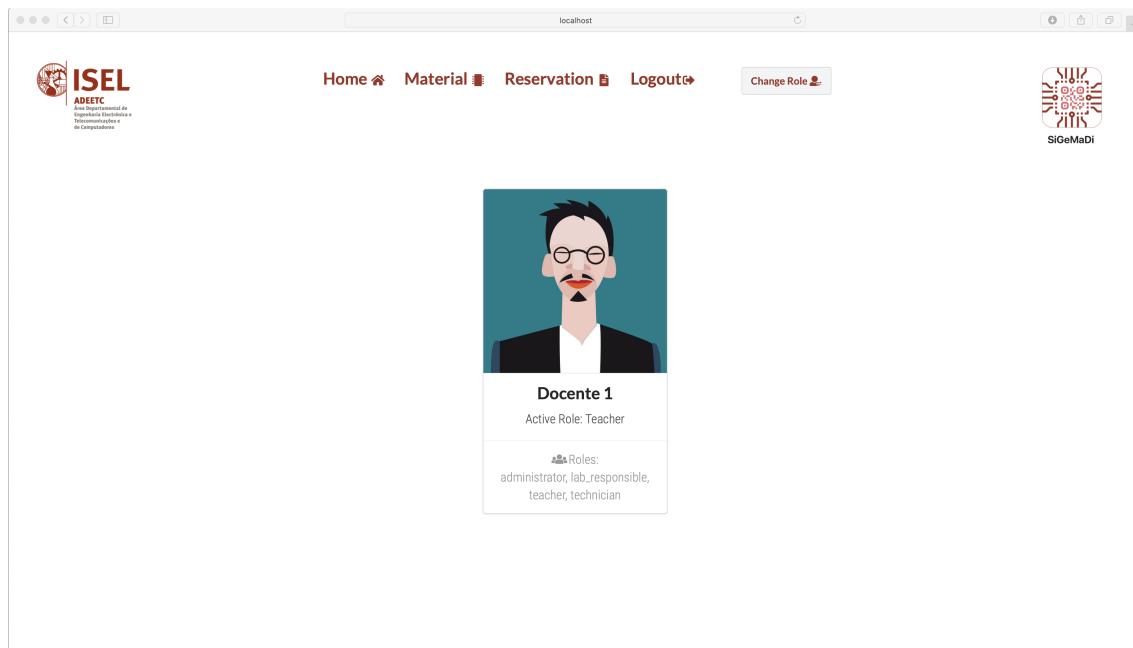


Figura 5.41: Página Principal do Docente

A screenshot of a web browser showing a grid of material items. The title 'Material:' is at the top. Below it are three rows of boxes, each containing a material identifier and a 'Details' button. The first row contains: 'ATF750CL 000-001-0000', 'R_100 000-002-0000', and 'R_1k 000-003-0000'. The second row contains: 'R_2k 000-004-0000', 'ATB_0001 001-001-0001', and 'ATB_0002 001-001-0002'. The third row contains: 'ATB_0003 001-001-0003', 'ATB_0004 001-001-0004', and 'ATB_0005 001-001-0005'. Each box also has a 'Details' button. At the top of the page, there are filter options for 'Filter by Scientific Areas', 'Filter by Subjects', and 'Filter by Material Types', each with dropdown menus and a 'Filter' button. The top navigation bar is identical to Figure 5.41.

Figura 5.42: Material

Material Details

Name: USBPort_0010
Type: USBPort
Scientific Areas: hardware_software
Subjects: LIC

Description: Material used as bridge between software and hardware

More Information:

Figura 5.43: Detalhes do Material

O Docente ao selecionar ”Reservation” e de seguida o botão ”New Reservation”, é redirecionado para uma página onde será apresentada uma lista de tipos de materiais, possíveis de serem selecionados. Esta interface ao contrário da requisição, não apresenta os materiais disponíveis, mas os tipos de materiais, uma vez que o Docente apenas necessita de especificar os tipos e as quantidades para cada tipo de material, a gestão dos materiais a selecionar para a reserva é feita ao nível da base de dados.

Material:

Filter by Scientific Area: All Filter by Subjects: All Filter

+ Add Material to Reservation

ATF750CL 000-001 Select	R_100 000-002 Selected!	R_1k 000-003 Select	R_2k 000-004 Select
ATB 001-001 Select	USBPort 001-002 Select	PDS16 001-003 Selected!	Osciloscopio 003-001 Select
Multímetro 004-001 Select			

Figura 5.44: Reservar Material

Quando o docente tencionar finalizar a reserva, seleciona o botão ”Add Material to Reservation”, que apresentará a página para a finalizar. Para esse efeito é necessário especificar a data, a unidade curricular a leccionar nessa aula, a quantidade de grupos, a quantidade por grupo dos materiais selecionados, e ainda a duração da aula para não haver colisões de aulas e falta de material, como ilustrado nas Figuras 5.45 e 5.46.

Figura 5.45: Finalizar Reserva

Figura 5.46: Escolha da Data para a Reserva

Por fim, o Docente tem ainda a possibilidade de visualizar as suas reservas ao selecionar apenas a opção "Reservation" no topo da página, onde é apresentada uma página com a lista das reservas, e ao selecionar um item da lista são mostrados os detalhes da reserva, nomeadamente as horas, os materiais, a unidade curricular, quantidade de grupos, entre outros. Tem ainda a possibilidade de remover a reserva caso selecione o botão "Delete Reservation" como representado nas Figuras 5.47 e 5.48.

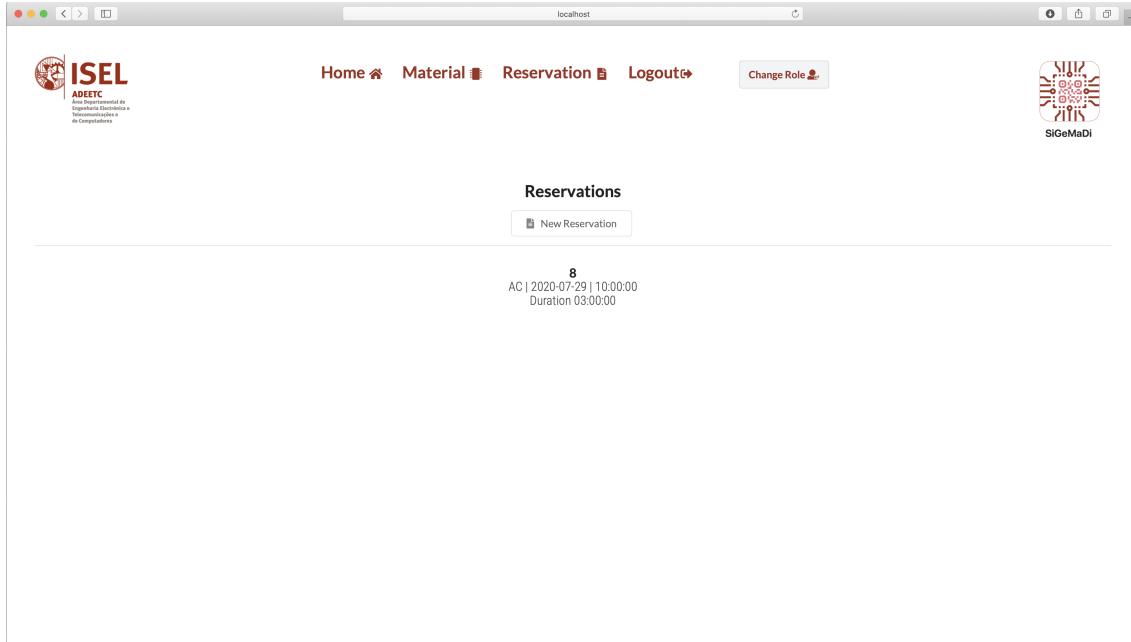


Figura 5.47: Reservas

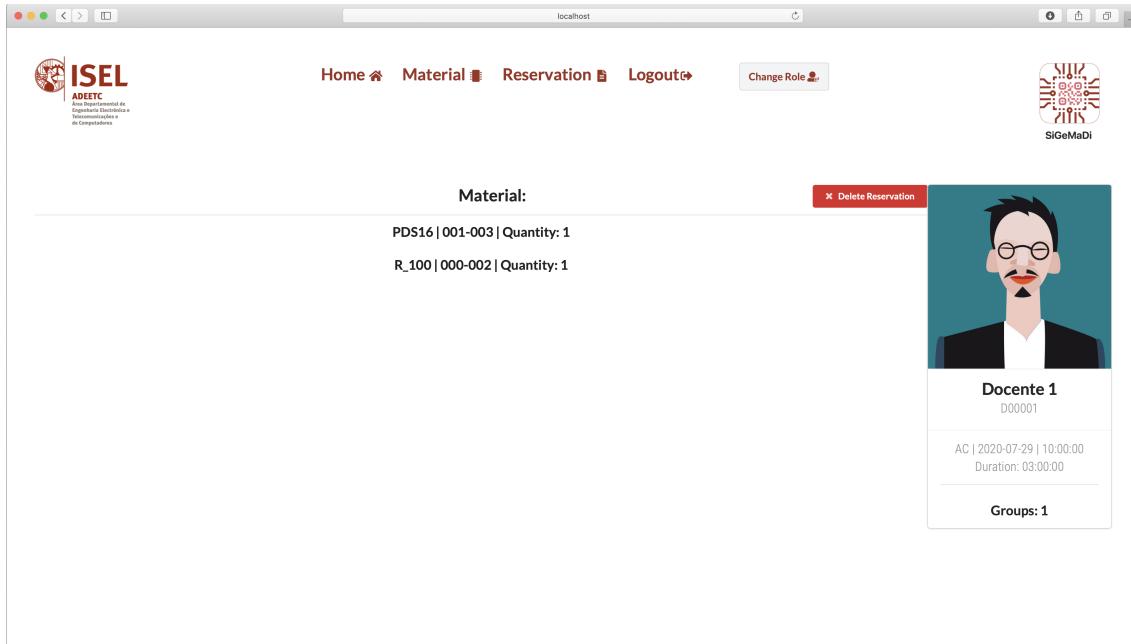


Figura 5.48: Detalhes da Reserva

5.2.1 Detalhes de implementação da aplicação Web

Na implementação de novos módulos na aplicação Web, introduziu-se um novo conceito que a biblioteca *React* adquiriu recentemente, os *Hooks* [35]. Este conceito oferece a possibilidade de implementar componentes com estado, sem que estas sejam representadas sob a forma de uma classe, passando a ser apenas uma função. Desta forma simplificou a estrutura de estado e ciclo de vida do componente, deixando de existir um objeto *state* que armazena as diversas propriedades que necessitam de ser mantidas durante o ciclo de vida do componente. As propriedades são agora representadas por uma variável e método de afetação, com o *Hook useState*. Os métodos de ciclo de vida deixaram de ser representados por diversos métodos, passando a ser apenas um, no caso o *Hook useEffect*. Optou-se pela implementação de novos componentes com este novo conceito, devido à simplificação na gestão de estado e

redução de linhas de código, tornando os componentes mais perceptíveis.

Para realizar a leitura do *QR Code* de cada material através da câmera do computador, recorreu-se a um dos mais variados módulos que a plataforma *npmjs* [36] oferece, no caso, o módulo *react-qr-reader* [37]. Este módulo recorre ainda à biblioteca *jsQR* [38], que executa a leitura da imagem, localizando e extraiendo o *QR Code*. Desta forma a imagem pode ter diferentes orientações, que a leitura é realizada. São utilizados recursos como o *WebRTC standards* para realizar o acesso à câmera e à informação que esta capta. O algoritmo presente neste módulo foi desenvolvido por forma a que a leitura do *QR Code* seja realizada o mais curto tempo possível, colocando a leitura num novo processo. É possível reparar na interface de Funcionário a aplicação deste módulo, como ilustrado nas Figuras 5.25 e 5.33.

Aquando uma reserva de material para uma aula, é necessário especificar a data e hora, e para que a experiência do utilizador seja a mais cómoda possível, utilizou-se um módulo da mesma plataforma, *semantic-ui-calendar-react* [39]. Este módulo oferece uma interface de calendário, onde é possível navegar pelos diversos dias, meses e anos do calendário, e ainda se necessário, selecionar a hora e minutos numa determinada data. Dos diferentes módulos que satisfaziam este requisito, *semantic-ui-calendar-react* foi o que teve maior destaque, devido à sua representação visual, integração no projeto, e ainda a facilidade de utilização. É possível reparar a aplicação deste módulo na interface de Docente, aquando a finalização da reserva, como representado na Figura 5.46.

Um outro módulo utilizado foi o *react-semantic-toasts* [40], que tem como função representar uma mensagem sobreposta à interface representada, para simbolizar um aviso, seja este de sucesso, alerta ou erro. Na iteração com as diferentes ações do sistema, estes avisos são utilizados para informar o utilizador de que a ação foi concluída com sucesso, que ocorreu um erro ou que houve falta informação para a realizar o pedido.

Por fim, o último módulo utilizado foi o *recharts* [41], que tem como função representar dados estatísticos sob a forma de gráficos, uma funcionalidade requerida nos cargos Administrador e Responsável de Laboratório. É possível representar dados estatísticos e filtrá-los com maior facilidade e clareza através deste módulo. A sua aplicação é possível de notar na Figura 5.16.

À semelhança da aplicação móvel, foi necessário integrar um método para a paginação de informação fornecida pela *API*. Desta forma implementou-se um *Infinite Scroll*, com recurso a uma interface da *API Intersection-Observer Api* [42], destinada para este fim. Na implementação do *infinite scroll* é frequentemente utilizada esta interface da *API*, que dado um elemento presente na representação página como referência, sempre que a barra de navegação desce até intersetar esse elemento, é feito um pedido para a página seguinte. A interface disponibiliza a propriedade *intersectionRatio* com valor entre 0 e 1, em que caso o valor seja maior que 0, significa que o elemento referência da interface foi intersetado, e um novo pedido é realizado.

Na passagem de materiais selecionados, para a finalização de reserva ou requisição é feita através de um objeto que armazena informação no *web browser*. A informação é armazenada durante uma sessão, entenda-se sessão como todo o tempo em que o browser permanece aberto, ou até um tempo quase indefinido, mesmo que o browser seja encerrado. Para estas duas opções existem os objetos *sessionStorage* [43] e *localStorage* [44] respetivamente. No caso apresentado é utilizado o objeto *localStorage* devido às suas características, nomeadamente a permanência de informação mesmo que o browser seja encerrado, o que revela uma melhor experiência de utilização. Quando o utilizador encerra o browser por engano, não é necessário que este introduza novamente os dados. No caso do utilizador selecionar os materiais para uma requisição, mas necessitou de consultar uma informação e saiu da página, ao voltar à página de seleção de materiais, estes estão pré-selecionados, reduzindo o tempo de realização de uma requisição.

Por fim, para acrescentar consistência e segurança na utilização da aplicação web, foi implementado o controlo de acessos para o utilizador com sessão iniciada. Ou seja, o utilizador, só conseguirá aceder às interfaces de cargos associados. Um utilizador que apenas desempenha o cargo de Funcionário, não conseguirá aceder a mais nenhuma outra interface. Caso o utilizador tenha mais do que um cargo, conseguirá desempenhar diferentes cargos dentro da mesma sessão, sem que tenha que realizar um novo *login*, através do botão "Change Role" tal como representado na Figura 5.17.

5.3 Componente Servidora

A API segue uma arquitetura *REpresentational State Transfer (REST)*, desta forma, os pedidos enviados para o servidor precisam de ser *stateless*, ou seja, precisam de conter toda a informação necessária para poder compreender o pedido não podendo usufruir de qualquer contexto que o servidor tenha armazenado, os pedidos são realizados recorrendo ao protocolo *HTTP*, e os dados são representados no formato *JavaScript Object Notation (JSON)*.

A API está desenhada de forma a que certos papéis (*roles*) tenham acesso exclusivo a *endpoints* específicos, uma vez que com base no conjunto de papéis definidos para um utilizador. Assim, se este tentar realizar um pedido para um recurso do qual não tem acesso, o servidor irá responder indicando que o utilizador não possui autorização para aceder ao respetivo recurso. No caso de um pedido ser efetuado com sucesso, a componente servidora responderá ao cliente com o recurso que pretendeu aceder sendo que, no caso de ocorrer um erro, esta enviará uma resposta de erro com o código *HTTP* e a respetiva mensagem de erro. Pelo facto de a API ser pública, é necessário a verificação de todos os erros passíveis de serem cometidos pelos clientes ou até mesmo tentativas de forjar algum pedido.

Todos os recursos precisam de ser acedidos recorrendo ao endereço: {Servidor:Porto}/sigemadi/api.

Para a maior parte dos recursos do sistema, podem ser realizadas ações de obtenção, criação, atualização e remoção de dados existentes na base de dados, com exceção de alguns casos apenas permitirem consulta de informação. São listados no anexo A os *endpoints* existentes na API divididos por recurso.

5.4 Detalhes de implementação da componente servidora

Um dos detalhes mais importantes no contexto da implementação da componente servidora tem essencialmente que ver com a lógica transacional do sistema.

Pelo facto de a base de dados estar implementada com PostgreSQL, o modelo transacional revela ser um pouco diferente da convenção, isto pelo facto de a linguagem recorrer a um modelo denominado por *Multiversion Concurrency Control (MVCC)* [45]. Este modelo permite gerir a consistência dos dados pelo facto de aquando da inicialização de uma transação, a mesma observa o estado da base de dados mais atualizado e consistente em memória (recorrendo ao mecanismo de *snapshot*) independentemente do estado atual da base de dados, prevenindo a visualização de estados inconsistentes produzidos por atualizações de informação de transações concorrentes. Assim, não recorre às técnicas de *locking* convencionais existentes noutros sistemas de gestão de bases de dados, o que acaba por reduzir o uso dos mesmos e aumentar a *performance* em ambientes multi-utilizador.

Tal modelo implica alterações no serviço *backend* do sistema. Segundo a documentação da linguagem PostgreSQL [46], mais concretamente na secção do isolamento transacional é referido que para níveis de isolamento superiores a *repeatable read* as aplicações requerem preparativos para que as transações possam ser relançadas (no caso de serem *read/write*) sob pena de poderem existir erros de serialização. Um erro de serialização é uma anomalia resultante da comparação de resultados realizados em concorrência por uma ou mais transações, comparativamente com o resultado simulado em série, na medida em que ocorre no caso de ambos os resultados serem diferentes. Desta forma, quando a transação que se deparou com a anomalia for relançada, já irá observar o estado mais consistente na base de dados.

Desta forma, revelou-se a necessidade de implementar um mecanismo de relançamento transacional na componente servidora. A nossa solução assenta numa classe, denominada por *Retry* com um método também ele chamado *retry*. Este método possuí um contador cujo valor é estipulado pelo argumento que recebe em parâmetro, e realiza um acesso à base de dados num *while-loop*. No caso de a exceção produzida possuir o código *SqlState* número 40001, indicando que se trata de um problema de serialização, é feita uma espera, com tempo aleatório, pela *thread* responsável pelo pedido, e após isso, o acesso é realizado novamente e o contador é decrementado, garantindo a existência de tempo de espera suficiente para que outras *threads* terminem o seu acesso à base de dados.

Este processo repete-se até ao contador chegar a zero, sendo que a operação é rejeitada se até lá o acesso não tiver sido feito com sucesso.

Como mencionado na descrição da *Web App* e *Mobile App*, revelou-se a necessidade de implementar paginação na API pelo facto garantir que não existem grandes volumes de dados a serem transferidos, existindo uma melhor gestão do tráfico de rede. Dessa forma estabeleceu-se um limite de 30 elementos por página e, para os pedidos *GET* adicionou-se um campo *page* na *query string* de forma a permitir aos utilizadores definirem qual a página que pretendem aceder.

Ainda, um dos detalhes que é preciso ter em conta tem que ver com o *handling* de erros provenientes da base de dados e da construção dos pedidos *HTTP* vindos dos clientes.

A gestão de notificações de erros é de extrema importância e, pelo facto de a maior parte das exceções provenientes da base de dados não serem de todo úteis para apresentar ao utilizador, uma vez que apresentam informações demasiado técnicas e de contextos diferentes, foi necessária a criação de um mecanismo de mapeamento de exceções provenientes da base de dados para exceções da componente servidora.

Para tal, todas as funções verificadoras existentes na base de dados, aquando de exceção, geram no corpo da mensagem um código pré-definido, sendo que para cada contexto existe um código único que o representa. Cada código é composto pelas iniciais da mensagem que pretende ser enviada e ainda o código de erro *HTTP* que se pretende usar na resposta ao cliente. Por vezes, é vantajoso enviar no corpo da resposta os parâmetros que, depois de verificados, desencadearam a exceção. Dessa forma, apresenta-se a formatação usada:

<siglas da mensagem><código de erro HTTP>:<parâmetros/argumentos desencadeadores>

Na componente servidora, um *handler* de exceções é responsável por capturar as mesmas que sejam do tipo *PSQLException* e realizar o devido mapeamento para exceções do servidor.

Exemplo

Considerando que a componente servidora chama uma função (ou procedimento) registada na base de dados e que, na sequência de verificações dos argumentos passados em parâmetros das mesmas é produzida uma exceção de invalidação, por parte de um ou vários argumentos (suponhamos que se trata de um identificador de um material), sendo que o código possível de ser passado é min400:001-001-0042 indicando que o material identificado por 001-001-0042 é inválido e não se encontra na base de dados.

Por consequência, a exceção mapeada pela componente servidora é a apresentada de seguida.

```
{  
    "type": "/sigemadi/exceptions/material-not-found"  
    "title" : "Invalid Material",  
    "detail" : "The collection of materials [001-001-0042] doesn't exists",  
    "status": 400  
}
```

Outros dos detalhes a ter em conta tem que ver com a notificação de utilizadores a partir do servidor. Esta podia ser feita de duas formas, através de notificação por *email* ou por SMS. Inicialmente definiu-se que o serviço teria a componente de notificação pelas duas vias, contudo, tendo em conta que o grau de complexidade para a implementação de uma solução com suporte a notificação por SMS é superior à solução baseada em *email*, e ainda pelo facto de a primeira não revelar custos adicionais, tomou-se a decisão de nesta fase implementar uma solução baseada apenas na notificação por *email*, sendo que mais tarde pretende-se incluir uma versão baseada em SMS. Para poderem ser realizadas as notificações, foi definida uma configuração na componente servidora onde se estipula o *host* (*smtp.office365.com*) e o porto (587) que serão usados para realizar o envio da mensagem bem como as credenciais do utilizador que irá realizar o envio da mesma aos utilizadores. Para além disso, ainda é definido o protocolo de envio (*SMTP*) bem como é estipulada a realização do envio recorrendo ao *TLS* (*Transport Layer Security*).

Finalmente, como foi mencionado no Capítulo 3, a solução proposta considera 6 papéis fundamentais: administrador; responsável de laboratório; técnico; docente; funcionário e estudante, sendo que cada papel exerce diferentes funções, razão pela qual é necessário controlar quem tem permissões para aceder a certos recursos. Desta forma, foi decidido gerar um *token JWT(JSON Web Token)* para cada utilizador do sistema implementado, o qual contém os papéis que cada utilizador exerce no *payload*. Este *token* é retornado depois do utilizador se autenticar na componente servidora, e precisa de ser inserido no *header Authorization* sobre o esquema *Bearer* de todos os pedidos, por forma a verificar se o utilizador autenticado tem acesso ao recurso que pretende obter.

6 Resultados Experimentais

No presente capítulo são apresentados e analisados os resultados experimentais provenientes da execução de testes de carga ao serviço *backend* do sistema, de forma a poder compreender a eficiência da solução encontrada para a componente servidora.

6.1 Metodologia

Desta forma, foram realizados dois tipos de testes ao servidor - leitura e escrita (*read/write*) - de modo a observar os tempos atingidos de execução da operação *target*, sendo que os *endpoints* que se escolheram para realizar os testes foram:

- GET /sigemadi/api/materials;
- POST /sigemadi/api/materials.

A seleção é justificada pelo facto de se prever que o primeiro *endpoint* seja o mais utilizado pelos utilizadores do sistema independentemente do cargo que ocupam, a maioria dos cargos requisitam informação proveniente deste URL, e o segundo pelo facto de ser uma operação que realiza alterações na base de dados ao contrário do primeiro.

Para implementar a bateria de testes usou-se a ferramenta *Apache JMeter* [47] configurada para a realização de pedidos *HTTP* sendo que foi necessária a criação de um *thread group* com um número configurável de *threads* representativas de utilizadores do serviço, e como tal, clientes *HTTP*, e ainda com um *ramp-up* fixo, i.e., um valor que define o número de *threads* criadas durante o período de tempo definido no parâmetro. Nos testes realizados, definiu-se o valor de *ramp-up* para 50 segundos, assim por cada x *threads* a serem criadas, o fluxo de *threads* por segundo foi $\frac{x}{50}$.

Um dos aspectos importantes também a analisar nestes testes, é verificar o impacto no desempenho na infraestrutura da implementação da componente servidora tendo por base a utilização de uma *JDBC connection pool* externa, designada por *HikariCP* [48]. Visto que esta implementação permitirá realizar uma melhor gestão das conexões à base de dados.

A bateria de testes foi realizada em quatro contextos diferentes, nomeadamente numa máquina virtual dedicada, pertencente ao departamento (ADEETC) do curso da organização alvo (ISEL) com a implementação da *connection pool*, em duas máquinas locais (1 e 2) também com a mesma implementação e finalmente na máquina local 2 sem a implementação da *connection pool* de forma a compreender o impacto deste componente no desempenho global do sistema. Apresentam-se na Tabela 6.1 as especificações de cada uma das máquinas utilizadas nos testes.

Máquina	Virtual	Local - 1	Local - 2
CPU	Broadwell-noTSX Dual-Core	Intel Core i7 - 8550U Quad-Core 1.80Ghz (Turbo boost: 4.00GHz)	Intel Core i9 Octa-Core 2.30GHz (Turbo boost: 4.00GHz)
Memória RAM	2Gb	DDR4 2400MHz 16Gb	DDR4 2666 MHz 32Gb
SSD	Unknown	256Gb PCIe NVMe M.2	Flash Storage 1Tb
Sistema Operativo	Ubuntu Trusty Tahr LTS	Windows 10 (64 bits)	MacOS 10.15.06 (Catalina - 64 bits)

Tabela 6.1: Especificações das máquinas usadas para a execução dos testes

6.2 Resultados experimentais

A presente subsecção apresenta os resultados experimentais obtidos para as diferentes baterias de testes realizadas. Para obter resultados mais estáveis, cada teste é executado 10 vezes de forma a poder obter um valor médio de tempo para cada máquina. Primeiro apresentam-se os valores obtidos para o *endpoint* com método GET (secção 6.2.1) e em segundo para o *endpoint* com método POST (secção 6.2.2).

6.2.1 GET /sigemadi/api/materials

Para iniciar a bateria de testes configurou-se uma coleção de 500 pedidos *HTTP* do tipo GET de forma a ter um ritmo de 10 *threads/s*, depois aumentou-se para 1000 para alcançar um ritmo de 20, 2000 para ter 40, 3000 para um ritmo de 60 e finalmente 6000 para ter um ritmo de 120 *threads/s*. Os resultados experimentais obtidos podem ser observados na Figura 6.1.

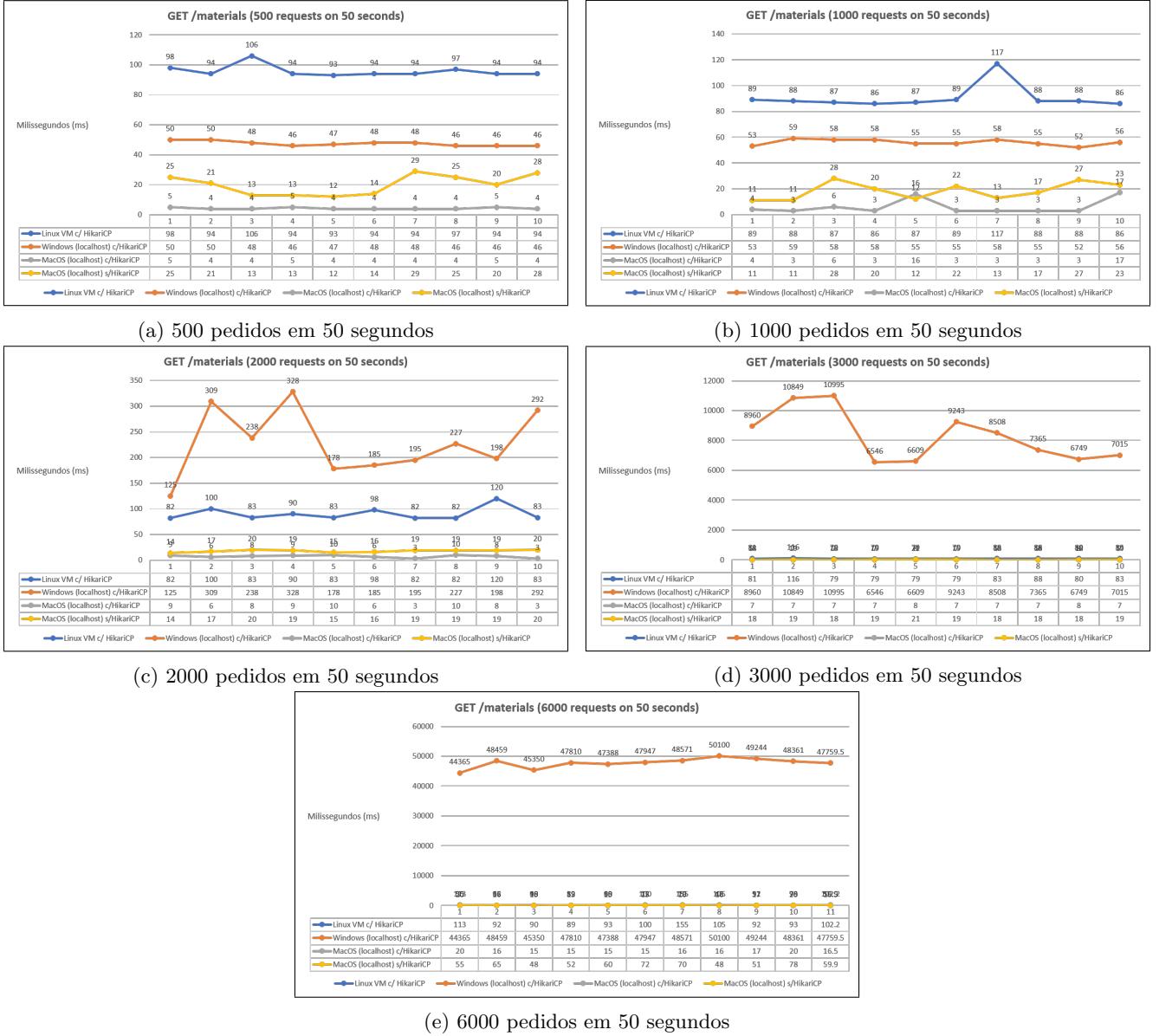


Figura 6.1: Tempos de execução para o endpoint GET /sigemadi/api/materials

Após a análise dos resultados é possível observar que os tempos mais reduzidos se encontraram presentes nas iterações realizadas pela máquina local - 2, com a *connection pool*, apresentando uma média de valores inferior em todas as baterias de teste relativamente aos outros ensaios. Também é possível concluir com os dados experimentais obtidos que as operações de obtenção de dados, na mesma máquina, têm tempos médios de execução mais reduzidos com a *connection pool* do que sem esta. Ainda, podemos concluir que para ritmos até ás 20 *threads/s* a máquina virtual tem um desempenho inferior relativamente às outras configurações, porém, é possível concluir-se que em média, a máquina local - 1 possui os maiores tempos de execução, mais concretamente quando o ritmo de pedidos atinge ou ultrapassa as 40 *threads/s*, atingindo o pico na última bateria de testes. Nesta situação o atendimento de um pedido tem uma média de 47759.5 segundos. Não considerando a máquina local - 1, a máquina virtual continua com tempos superiores aos observados em ambas as configurações na máquina local - 2.

Numa breve análise o fator que mais contribui para a degradação dos tempos da máquina local - 1, poderá ser a combinação do número de *threads* que são injetadas por segundo no serviço e o número de *worker threads* que são criadas aquando da inicialização do serviço, combinando ainda com o número de conexões do *pool* de conexões configuradas, dadas pelo *HirakiCP*, fazendo com que o ritmo de pedidos injetados seja maior do que o ritmo de pedidos executados, fazendo com que o *stack* de pedidos aumente, tendo pedidos à espera para serem executados, o que contribui para um tempo de resposta maior e por consequência resultados mais pobres.

6.2.2 POST /sigemadi/api/materials

Para realizar o pedido de criação de materiais configurou-se que cada pedido era responsável por criar duas unidades do tipo ATB, i.e., com identificador 001-001, tal como pode ser observado no corpo do pedido *HTTP* na Figura 6.2

```
{
  "type": "001-001",
  "quantity": 2
}
```

Figura 6.2: Corpo do pedido *HTTP* usado para criar as unidades

Da mesma forma que para o método GET, começou-se por configurar uma coleção de 500 pedidos *HTTP* do tipo POST de forma a ter um ritmo de 10 *threads/s* sendo que as subsequentes foram configuradas da mesma forma com exceção da última bateria, onde se reduziu o valor de 6000 para 4000 pedidos.

Tal decisão justifica-se pelo facto de como são criados por cada *threads* duas unidades do material tipo ATB, e caso o valor permanece-se 6000 iria implicar que ao todo teriam que ser criadas 120000 unidades do mesmo material, sendo que esse valor ultrapassa o limite estabelecido nesta versão do sistema (existem apenas 4 dígitos representativos do número de material o que no máximo nos permite ter até 9999 unidades), logo, com 4000 o valor final de unidades criadas resume-se a 8000 o que já se encontra dentro do limite estabelecido.

Os resultados experimentais obtidos podem ser observados nas Figuras 6.3 e 6.4.

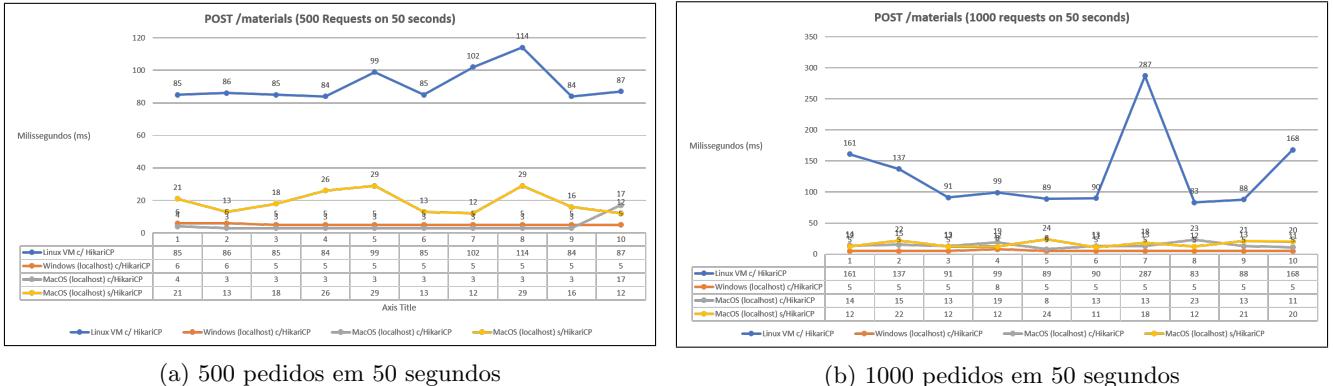


Figura 6.3: Dados recolhidos para o endpoint POST /sigemadi/api/materials

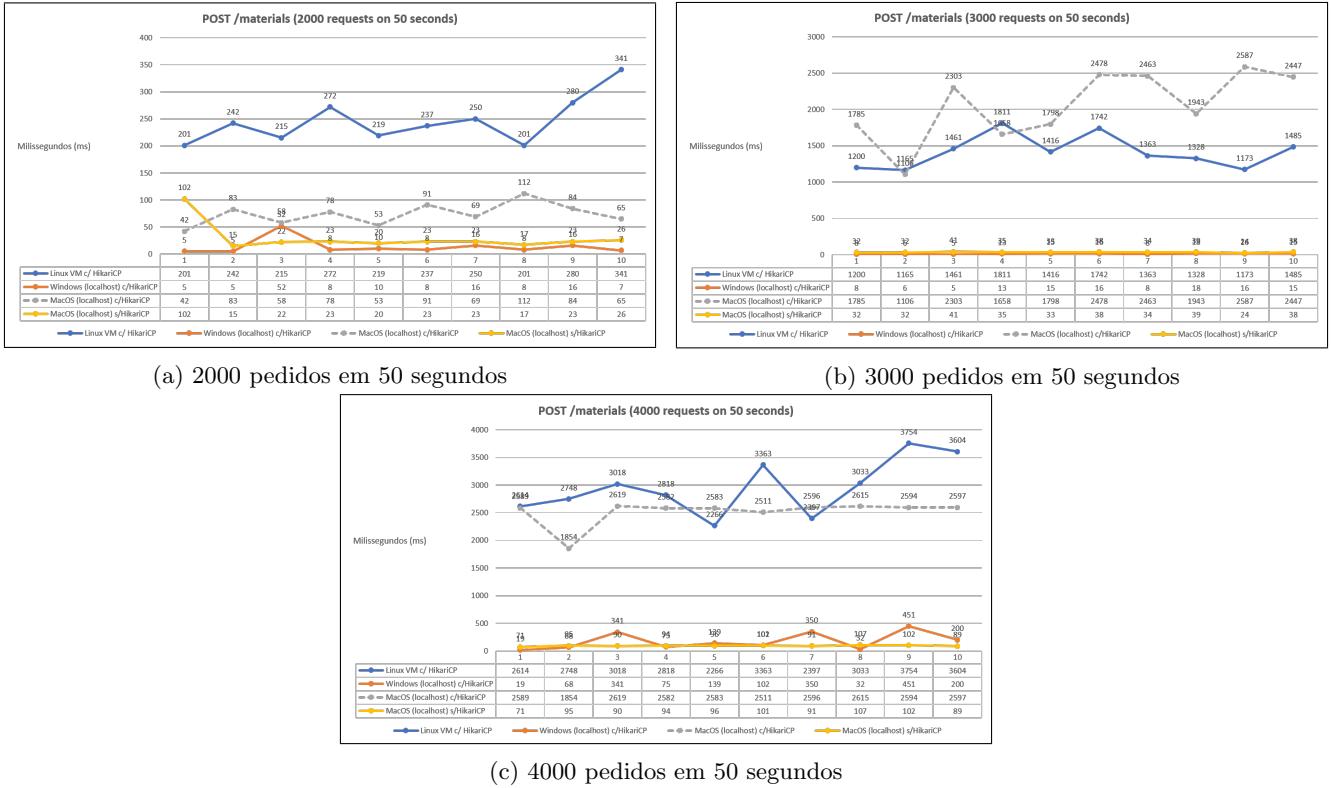


Figura 6.4: Tempos de execução para o endpoint POST /sigemadi/api/materials (continuação)

Com os resultados experimentais obtidos após a execução das baterias de testes, é possível concluir que os tempos de atendimento de um pedido demoram mais tempo na máquina virtual, comparativamente com as outras configurações, na medida em que, à medida que o ritmo aumenta, possuí tempos médios a variar entre os 91.1 ms (milissegundos) e os 2961.5 ms, equivalente a 2.9 segundos aproximadamente. Ainda, é possível extrair que, ao contrário do desempenho observado na execução de pedidos GET, a máquina local - 1 consegue obter a melhor média de tempos diante de todos os outros ensaios, tendo, com o aumento do ritmo, tempos médios a variar entre 5.2 ms e 177.7 ms.

Relativamente à configuração da máquina local - 2 sem *connection pool* é possível concluir que os tempos de execução são similares aos tempos da máquina local - 1 porém esta última obtém uma melhor *performance* para ritmos inferiores a 80 *threads/s*. Tais resultados podem ser influenciados pelo tipo de disco associado à máquina local e por consequência os tempos de escrita na base de dados, bem como a latência que existe na execução de um pedido num servidor que está localizado numa máquina e rede diferentes. Os tempos médios de execução obtidos para a terceira configuração, ou seja, para a máquina local - 2 com *connection pool*, revelam ser muito superiores comparativamente com os tempos da mesma máquina sem o *pool* de conexões. Como estes resultados não eram os expectáveis à priori, esta implementação requer uma análise futura mais detalhada de modo a compreender qual ou quais os fatores que degradam o desempenho.

É importante referir que durante a execução dos testes, existiram situações onde ocorreram erros (assinalados a vermelho no gráfico) que podem ser analisados na tabela 6.2.

	Máquina Virtual						Máquina Local - 1	
	1 ^a	6 ^a	7 ^a	8 ^a	9 ^a	10 ^a	1 ^a	4 ^a
2000 pedidos em 50 segundos	-	-	0.2%	-	0.05%	-	-	-
3000 pedidos em 50 segundos	-	-	-	-	-	-	0.07%	0.07%
4000 pedidos em 50 segundos	0.3%	0.65%	-	0.62%	-	0.03%	-	-

Tabela 6.2: Percentagens de erro resultantes da execução das iterações por diferentes máquinas e baterias de testes

Ao analisar a percentagem de erro gerada aquando da execução de cada tentativa, é possível observar uma maior

taxa na última bateria de testes sendo que tal facto pode ser justificado pelo maior fluxo de pedidos a serem injetados no servidor para serem processados, em combinação com o algoritmo de relançamento transacional, aplicado quando uma operação na base de dados que recorra a um nível de isolamento igual ou superior a *repeatable_read*. As operações na base de dados são relançadas até um limite de 10 vezes ou até existir rejeição, quando existe um número elevado de *threads* a interagir com a base de dados, a probabilidade de ocorrência de erros de serialização é maior, o que leva à execução do algoritmo de relançamento e que o mesmo tenha que ser executado até ao seu limite máximo. Contudo, é importante notar que a taxa de erro obtida face ao número de materiais criados é muito baixa.

7 Conclusões e Trabalho Futuro

7.1 Conclusões

O projeto aqui apresentado teve como principal objetivo desenvolver um sistema de informação focado na gestão de material didático, de forma a desmaterializar o processo existente na organização alvo (ISEL). Este objetivo foi concluído com sucesso na medida em que se conseguiu implementar uma solução que torna possível o mapeamento de informação, habitualmente física, para o meio digital, servindo-se de técnicas, nomeadamente o identificador criado para os materiais existentes na organização - inclusive, para posterior leitura através de *QR Code*. Além da gestão de material, o sistema permite a atribuição de diferentes tipos de papéis de forma a escalar as diferentes responsabilidades e ações possíveis dentro do sistema e a organização dos materiais por áreas científicas, unidades curriculares e tipos de forma a permitir melhor organização da informação.

Na solução proposta foram criados seis tipos de papéis, nomeadamente, administrador, responsável de laboratório, técnico, docente, funcionário e estudante, sendo que estes últimos dois cargos são um pouco diferentes dos restantes na medida em que um utilizador que seja previamente registado no sistema com um destes papéis, não pode assumir mais nenhum cargo que não seja aquele que lhe foi atribuído inicialmente, tal não acontece com os restantes cargos, i.e., um utilizador que previamente seja registado como sendo responsável de laboratório, pode assumir também o cargo de administrador e/ou docente por exemplo.

Tal como foi dito anteriormente, cada papel assume um conjunto de responsabilidades sendo que o administrador opera na definição de cargos sobre os restantes utilizadores do sistema, na criação de áreas científicas, unidades curriculares e consultas de indicadores estatísticos para eventuais tomadas de decisão; o responsável de laboratório pode gerir os ativos associados à área científica que o mesmo representa e também pode consultar os mesmos indicadores anunciados anteriormente; o técnico assume operações de consulta de materiais com avaria (previamente submetidos pelo funcionário) e de submissão de materiais analisados pelo mesmo, sendo que os mesmos poderão ter sido reparados com sucesso ou não, neste último caso, com a implicação de terem que ser removidos do sistema por parte de um utilizador que assuma o papel de responsável de laboratório; o docente opera na consulta de materiais existentes no inventário e na reserva dos mesmos para uma aula laboratorial que este queira dar no futuro.

O sistema está preparado para que não existam conflitos de aulas criadas à mesma altura para o mesmo docente e também para eventuais tentativas de utilização excessiva dos recursos existentes no sistema, ou seja, no caso de um docente pretender criar uma reserva com um determinado número de materiais que somando aos de uma reserva previamente submetida, ultrapassa o valor existente no inventário do sistema. O funcionário é o papel que detém de mais responsabilidades no sistema na medida em que é ele que trata do registo de requisições dos alunos, submete avarias para serem verificadas pelos técnicos e prepara aulas laboratoriais, recorrendo à leitura por *QR Code* de cada material facilitando o trabalho do mesmo, bem como diminui tempos de espera por parte do docente e alunos para dar inicio às aulas e finalmente o estudante opera na consulta de materiais disponíveis para este requisitar, e na eventual submissão de intenções de avaria de forma a que este possa dar *feedback* com base na sua experiência (embora que pouca), para contribuir numa melhor asserção da causa para a respetiva avaria, permitindo uma reparação mais eficaz por parte dos técnicos.

7.2 Trabalho Futuro

Após a conclusão da implementação das funcionalidades previstas para o projeto, e algumas funcionalidades adicionais também concretizadas e já enumeradas na secção anterior, realizou-se uma análise de aspectos que futuramente possam contribuir para o aumento do nível de qualidade do sistema aqui proposto e desenvolvido.

Assim, considera-se que autenticação no sistema da instituição de ensino (ISEL), aproveitando a informação referente ao utilizador que está presente na base de dados da organização. Desta forma, todo o mecanismo implementado à data, baseado na geração e emissão de um *token JWT* com o modo de autenticação *Basic Authentication*, poderá ser substituído.

Associado ao sistema de autenticação pretende-se providenciar uma conexão segura entre o servidor e o cliente através do protocolo *HTTPS* (*Hyper Text Transfer Protocol Secure*). Esta implementação poderá ser desenvolvida recorrendo à *framework Spring Security*, permitindo estabelecer redirecionamento de pedidos *HTTP* para *HTTPS*, no entanto esta solução requer a emissão de um certificado devidamente autorizado por uma entidade certificadora confiável.

Ao utilizar a *framework Spring Security* promove-se também a proteção do serviço contra ataques do tipo *CSRF* (*Cross Site Request Forgery*). Assim, será possível recorrer a um padrão de sincronização de *tokens* (*Synchronizer Token Pattern*) cujo conceito insere-se sobre a emissão em cada pedido modificador de estado (POST, PUT, PATCH, DELETE, etc...) de um *token* aleatório colocado nos *headers*, esperando que numa posterior interação com o servidor

seja realizado uma verificação do *token* emitido anteriormente, também enviado como *header* do lado do cliente. No caso do *token* enviado pelo cliente não for válido, o serviço bloqueia a execução da operação.

Além das questões de segurança pretende-se implementar mais formas alternativas de notificação aos estudantes no contexto de entrega de material para a concretização de uma aula laboratorial, tais como o suporte a SMS e mensagens via serviço *WhatsApp*, este último é considerado pela sua popularidade que o mesmo tem diante dos utilizadores alvo do sistema proposto. Para esta abordagem propõe-se como solução a utilização da *Twilio's REST API* [49], caracterizada por ser uma plataforma para comunicações.

Referências

- [1] Relational databases are not designed for scale. <https://www.marklogic.com/blog/relational-databases-scale/>. consultado no dia 20/04/2020.
- [2] Parago website. <https://www.paragosoftware.com/products/it-inventory-management/>. consultado no dia 01/03/2020.
- [3] Ezofficeinventory website. <https://www.ezofficeinventory.com/solutions/qrcode-asset-tracking-software>. consultado no dia 01/03/2020.
- [4] Sortly website. <https://www.sortly.com/>. consultado no dia 01/03/2020.
- [5] Xamarin website. <https://dotnet.microsoft.com/apps/xamarin>. consultado no dia 01/03/2020.
- [6] React native website. <https://reactnative.dev/>. consultado no dia 01/03/2020.
- [7] Flutter website. <https://flutter.dev/>. consultado no dia 01/03/2020.
- [8] What is xamarin? <https://docs.microsoft.com/en-us/xamarin/get-started/what-is-xamarin>. consultado no dia 02/04/2020.
- [9] Visual studio website. <https://visualstudio.microsoft.com/pt-br/?rr=https%3A%2F%2Fwww.google.com%2F>. consultado no dia 01/03/2020.
- [10] Hot reloading. <https://reactnative.dev/blog/2016/03/24/introducing-hot-reloading>. consultado no dia 02/05/2020.
- [11] React - a javascript library for building user interfaces. <https://reactjs.org/>. consultado no dia 02/04/2020.
- [12] Angular - one framework. mobile & desktop. <https://angular.io/>. consultado no dia 02/04/2020.
- [13] React vs angular vs vue.js — what to choose in 2020? (updated in 2020). <https://medium.com/TechMagic/reactjs-vs-angular5-vs-vue-js-what-to-choose-in-2018-b91e028fa91d>. consultado no dia 02/04/2020.
- [14] Angularjs vs nodejs vs reactjs: Which one would you choose? <https://www.konstantinfo.com/blog/angularjs-vs-nodejs-vs-reactjs/>. consultado no dia 02/04/2020.
- [15] Django website. <https://www.djangoproject.com/>. consultado no dia 02/04/2020.
- [16] Spring boot website. <https://spring.io/projects/spring-boot>. consultado no dia 02/04/2020.
- [17] Spring boot vs django. <https://www.slant.co/versus/158/1746/> spring-boot_vs_django. consultado no dia 02/04/2020.
- [18] Phoenix website. <https://www.phoenixframework.org/>. consultado no dia 20/04/2020.
- [19] The java virtual machine. <https://docs.oracle.com/javase/specs/jvms/se7/html/>. consultado no dia 20/04/2020.
- [20] Java jdbc api. <https://docs.oracle.com/javase/8/docs/technotes/guides/jdbc/>. consultado no dia 20/04/2020.
- [21] Microsoft open database connectivity. <https://docs.microsoft.com/en-us/sql/odbc/microsoft-open-database-connectivity-odbc?view=sql-server-ver15>. consultado no dia 20/04/2020.
- [22] Spring data jpa. <https://docs.spring.io/spring-data/jpa/docs/current/reference/html/#reference>. consultado no dia 20/04/2020.
- [23] Jdbc vs. hibernate: Which to choose? <https://www.devteam.space/blog/hibernate-vs-jdbc-which-to-choose/>. consultado no dia 20/04/2020.
- [24] Vitor Gabriel. Gestão de materiais. <http://bdigital.ipg.pt/dspace/bitstream/10314/977/1/Manual%2017%20-%20Gest%C3%A3o%20de%20Materiais.pdf>, 2005. consultado no dia 01/03/2020.
- [25] Zxing (“zebra crossing”) barcode scanning library for java, android. <https://github.com/zxing/zxing>. consultado no dia 07/06/2020.

- [26] Zbar bar code reader. <http://zbar.sourceforge.net/>. consultado no dia 07/06/2020.
- [27] Introduction to mobile vision. <https://developers.google.com/vision/introduction>. consultado no dia 07/06/2020.
- [28] The world's best ui component suite for building powerful web, desktop, and mobile apps. <https://www.syncfusion.com/>. consultado no dia 07/06/2020.
- [29] What is the community license? <https://www.syncfusion.com/products/communitylicense>. consultado no dia 07/06/2020.
- [30] Xamarin.forms documentation. <https://docs.microsoft.com/en-us/xamarin/xamarin-forms/>. consultado no dia 07/06/2020.
- [31] Xamarin.forms.extended.infinitescrolling. <https://www.nuget.org/packages/Xamarin.Forms.Extended.InfiniteScrolling/1.0.0-preview2>. consultado no dia 07/06/2020.
- [32] Xamarin.forms local databases. <https://docs.microsoft.com/pt-pt/xamarin/xamarin-forms/data-cloud/data/databases>. consultado no dia 07/06/2020.
- [33] Xamarin.essentials: Secure storage. <https://docs.microsoft.com/en-us/xamarin/essentials/secure-storage?tabs=android>. consultado no dia 11/07/2020.
- [34] Xamarin.essentials. <https://docs.microsoft.com/en-us/xamarin/essentials/>. consultado no dia 11/07/2020.
- [35] Introducing hooks. <https://reactjs.org/docs/hooks-intro.html>. consultado no dia 08/06/2020.
- [36] Build amazing things. <https://www.npmjs.com/>. consultado no dia 08/06/2020.
- [37] React-qr-reader. <https://www.npmjs.com/package/react-qr-reader>. consultado no dia 08/06/2020.
- [38] Jsqr. <https://github.com/cozmo/jsQR>. consultado no dia 08/06/2020.
- [39] Semantic-ui-calendar-react. <https://www.npmjs.com/package/semantic-ui-calendar-react>. consultado no dia 08/06/2020.
- [40] React-semantic-toasts. <https://www.npmjs.com/package/react-semantic-toasts>. consultado no dia 08/06/2020.
- [41] Recharts. <https://www.npmjs.com/package/recharts>. consultado no dia 08/06/2020.
- [42] Intersection observer api. <https://developer.mozilla.org/en-US/docs/Web/API/IntersectionObserverAPI>. consultado no dia 08/06/2020.
- [43] Sessionstorage. <https://developer.mozilla.org/pt-BR/docs/Web/API/Window/sessionStorage>. consultado no dia 08/06/2020.
- [44] Localstorage. <https://developer.mozilla.org/pt-BR/docs/Web/API/Window/Window.localStorage>. consultado no dia 08/06/2020.
- [45] 13.1 introduction. <https://www.postgresql.org/docs/9.5/mvcc-intro.html>. consultado no dia 10/06/2020.
- [46] 13.2 transaction isolation. <https://www.postgresql.org/docs/9.5/transaction-iso.html>. consultado no dia 10/06/2020.
- [47] Apache jmeter™. <https://jmeter.apache.org/>. consultado no dia 16/07/2020.
- [48] Hikaricp. <https://github.com/brettwooldridge/HikariCP>. consultado no dia 19/07/2020.
- [49] Twilio. <https://www.twilio.com/P>. consultado no dia 20/07/2020.
- [50] EZOfficeInventory. School inventory management software makes teachers' lives easier. <https://blog.ezofficeinventory.com/school-inventory-management-software-teachers/>, 2017. consultado no dia 01/03/2020.
- [51] How are they different and what are the best sql and nosql database systems? <https://www.xplenty.com/blog/the-sql-vs-nosql-difference/>. consultado no dia 20/04/2020.

- [52] O que é react.js e porque devemos usá-lo? <https://edit.com.pt/blog/o-que-e-reactjs-e-porque-devemos-usa-lo/>. consultado no dia 02/04/2020.
- [53] What and why react.js. <https://www.c-sharpcorner.com/article/what-and-why-reactjs/>. consultado no dia 02/04/2020.
- [54] Desenvolvimento de aplicativos xamarin com visual studio. <https://visualstudio.microsoft.com/pt-br/xamarin/>. consultado no dia 02/04/2020.
- [55] Where do cross-platform app frameworks stand in 2020? <https://www.netsolutions.com/insights/cross-platform-app-frameworks-in-2019/>. consultado no dia 02/04/2020.
- [56] 9 popular cross-platform tools for app development in 2020. <https://hackernoon.com/9-popular-cross-platform-tools-for-app-development-in-2019-53765004761b>. consultado no dia 02/04/2020.
- [57] O melhor e o pior do xamarin. <https://codificar.com.br/o-melhor-e-o-pior-do-xamarin/>. consultado no dia 02/04/2020.
- [58] Graphic design resources for all. <https://medialoot.com/>. consultado no dia 20/04/2020.
- [59] Barcode api overview. <https://developers.google.com/vision/android/barcodes-overview>. consultado no dia 07/06/2020.
- [60] Android barcode scanning library landscape. <https://medium.com/@bherbst/android-barcode-scanning-library-landscape-109292b81b65>. consultado no dia 07/06/2020.
- [61] Zxing vs google vision. <https://medium.com/@lkumar.sakare/zxing-vs-google-vision-fc3be8d83ace>. consultado no dia 07/06/2020.
- [62] Serializable transactions. https://jdbi.org/_serializable_transactions. consultado no dia 10/06/2020.

A Documentação da API

SiGeMaDi (Sistema de Gestão de Material Didático) API documentation

The base part of the URI path is `/sigemadi/api`

Notice that for all following endpoints might occur the following types of response errors:

- 500 (Internal Server Error)
- 401 (Unauthorized)
- 401 (Unauthorized) - Invalid Token

So, if an endpoint doesn't present possible errors, it has to be considered the common ones shown above.

The following sections describe each API endpoint and their specific response errors

Authentication Management

Login

`POST /login`

- Request:
 - Headers:
 - Authorization: Basic Authentication
 - Body: none
- Response:
 - Success:
 - Status-code: 200 (OK)
 - Content-type: application/json
 - Body example:

```
{  
    "status": "The target user was authenticated successfully",  
    "token": "<the server generated JWT (Json Web Token)>",  
    "uri": "/sigemadi/api/users/<user_id>/roles"  
}
```

Material Management

Search material by parameters

```
GET /materials
```

Returns a list of inventory materials filtered by the allowed parameters.

Example

This is an example where the search is narrowed by the material state.

```
GET /materials?state=available
```

Parameter	Available values	Details
state	available ; damaged ; in_repair ; unavailable ;	represents the state of the material
sci_area	1 ; 2 ; 3 ; 4 ; ...	represents the id of the scientific area that material belongs
subject	1 ; 2 ; 3 ; ...	represents the id of the subject where the material is going to be used
type	001-001 ; 001-002 ; 000-003 ; ...	represents the id of the type of the material

- Request:
 - Body: none
- Response:
 - Success:
 - Status-code: 200 (OK)
 - Content-type: application/json
 - Body example (body returned for the example above):

```
{  
    "materials": [  
        ...,  
        {  
            "id": "000-002-0000",  
            "name": "R_100",  
            "available_quantity": 100,
```

```

        "can_be_reported": false
    },
{
    "id": "001-001-0001",
    "name": "ATB_0001",
    "can_be_reported": true
},
{
    "id": "001-001-0002",
    "name": "ATB_0002",
    "can_be_reported": false
},
{
    "id": "001-001-0003",
    "name": "ATB_0003",
    "can_be_reported": false
},
...
]
}

```

- Error:
 - 403 (Forbidden))

Get Material by id

GET /materials/:id

- Request:
 - Path parameters:
 - id - the material identifier
 - Body: none
 - Response:
 - Success:
 - Status-code: 200 (OK)
 - Content-type: application /json
 - Body example:
- ```

{
 "name": "ATB_0001",
 "type": "ATB",
 "description": "Material used to power and interact with electronic systems",
 "sci_area": "hardware_software",

```

```

 "subjects": [
 "LSD",
 "AC",
 "LIC"
],
 "state": "available",
 "can_be_reported": false
}

```

- Error:

- 404 (Not Found) - Material:

```

{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Material Not Found",
 "detail": "The material <material_id> doesn't exist",
 "status": 404
}

```



- 403 (Forbidden)

## Insert new material

POST /materials

- Request:

- Content-type: application/json

- Body:

```

{
 "type": "<the type identifier>",
 "quantity": 5
}

```

- Response:

- Success:

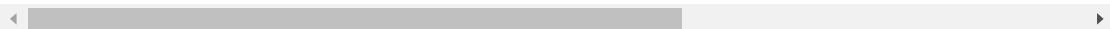
- Status-code: 201 (Created)
- Content-type: application /json
- Body example:

```
{
 "status": "New materials created successfully",
 "identifiers": [
 "001-001-0004",
 "001-001-0005",
 "001-001-0006",
 "001-001-0007",
 "001-001-0008"
],
 "uri": "/sigemadi/api/materials"
}
```

- Error:

- 400 (Bad Request) - Invalid Body:

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Body isn't valid",
 "detail": "The request body contains invalid parameters",
 "status": 400
}
```



- 400 (Bad Request) - Type:

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Invalid Type",
 "detail": "The type <material_type> isn't valid",
 "status": 400
}
```



- 403 (Forbidden)

## Delete existent material

`DELETE /materials/:id`

- Request:

- Path parameters:
  - id - the material identifier
- Body:

```
{
 "quantity": "<quantity of material to be removed - bulk material context>"
}
```

- Response:

- Success:

- Status-code: 200 (OK)
    - Content-type: application/json
    - Body example:

```
{
 "status": "Material with the id <id> was deleted successfully",
 "uri": "/sigemadi/api/materials"
}
```

- Error:

- 400 (Bad Request) - Invalid Body:

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Body isn't valid",
 "detail": "The request body contains invalid parameters",
 "status": 400
}
```

- 400 (Bad Request) - Unit Material removal:

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Quantity Not Equal To One",
 "detail": "You're trying to remove a unit material with a non unit quantity",
 "status": 400
}
```

- 400 (Bad Request) - Material not available :

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Material Not Available",
 "detail": "Material <material_id> is not available",
 "status": 400
}
```

- 404 (Not Found):

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Material Not Found",
 "detail": "The material <material_id> doesn't exist",
 "status": 404
}
```
- 403 (Forbidden)

## Roles management

---

### Get roles

GET /roles

- Request:
  - Body: none
- Response:
  - Success:
    - Status-code: 200 (OK)
    - Content-type: application/json
    - Body example:

```
{
 "roles": [
 "administrator",
 "lab_responsible",
 "staff",
 "student",
 "teacher",
 "technician"
]
}
```

## User management

---

## Get users by parameters

GET /users

Returns a list of users filtered by the allowed parameters.

### Example

This is an example where the search is narrowed by the user role.

GET /users?role=student

| Parameter | Available values                                                         | Details                         |
|-----------|--------------------------------------------------------------------------|---------------------------------|
| role      | administrator ; lab_responsible ; technician ; teacher ; staff ; student | represents the role of the user |
| id        | D<number> ; F<number> ; A<number> ;                                      | represents the user identifier  |

- Request:

- Body:none

- Response:

- Success:

- Status-code: 200 (OK)
    - Content-type: application/json
    - Body example:

```
{
 "users": [
 {
 "id": "A00001",
 "name": "Student 01",
 "photo": "<the user photo>",
 },
 {
 "id": "A00002",
 "name": "Student 02",
 "photo": "<the user photo>"
 },
 ...
]
}
```

```
]
}
```

- Error:
  - 403 (Forbidden)

## Get user by id

GET /users/:id

- Request:
  - Path parameter:
    - id - the id of the user
  - Body:none

- Response:

- Success:
  - Status-code: 200 (OK)
  - Content-type: application/json
  - Body example:

```
{

 "id": "A00001",
 "name": "Student 01",
 "phone_nr": "911111111",
 "email": "student_one@isel.pt",
 "photo": "<image in Base64>"
}
```

- Error:
  - 404 (Not Found):

```
{

 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exce
 "title": "User Not Found",
 "detail": "The user <user_id> doesn't exist",
 "status": 404
}
```

## Get user roles

GET /users/:id/roles

- Request:
  - Path parameters:
    - id - the user identifier
  - Body: none

- Response:
  - Success:
    - Status-code: 200 (OK)
    - Content-type: application/json
    - Body example:

```
{
 "roles": [
 {
 "name": "administrator"
 },
 {
 "name": "lab_responsible",
 "sci_area_id": 1
 },
 {
 "name": "teacher"
 },
 ...
]
}
```

- Error:
  - 404 (Not Found):

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exce
 "title": "User Not Found",
 "detail": "The user <user_id> doesn't exist",
 "status": 404
}
```



## Define roles for a specific user

```
PUT /users/:id/roles
```

- Request:

- Path parameters:
  - id - the user identifier
- Content-type: application/json
- Body:

```
{
 "roles": [
 {
 "role": "teacher"
 },
 {
 "role": "lab_responsible",
 "sci_area": "hardware_software"
 },
 ...
]
}
```

- Observations:

- When specified the role lab\_responsible it's mandatory specify the scientific area that the user is going to be responsible for

- Response:

- Success:

- Status-code: 200 (OK)
- Content-type: application/json
- Body example:

```
{
 "status": "Roles for the user with the id <id> were changed successfully",
 "uri": "/sigemadi/api/users/${id}/roles"
}
```

- Error:

- 400 (Bad Request) - Invalid Body:

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Body isn't valid",
```

```
 "detail": "The request body contains invalid parameters",
 "status": 400
 }
```

- 400 (Bad Request) - User is a Student or Staff Member:

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "User is Student or Staff Member",
 "detail": "The user <user_id> is a student or a staff member",
 "status": 400
}
```

- 400 (Bad Request) - Invalid Roles:

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Invalid Roles",
 "detail": "You've inserted invalid roles in your request body",
 "status": 400
}
```

- 400 (Bad Request) - Non-existent Scientific Area for a lab\_responsible role:

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Non Existant Scientific Area",
 "detail": "The user <id> is a student or a staff",
 "status": 400
}
```

- 404 (Not Found):

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "User Not Found",
 "detail": "The user <user_id> doesn't exist",
 "status": 404
}
```

- 403 (Forbidden)

# Scientific Area management

---

## Get scientific area

GET /scientificareas

- Request:
  - Body: none
- Response:
  - Success:
    - Status-code: 200 (OK)
    - Content-type: application/json
    - Body example:

```
{
 "sci_areas": [
 {
 "name": "geral",
 "id": 0
 },
 {
 "name": "hardware_software",
 "id": 1
 },
 {
 "name": "redes",
 "id": 2
 },
 {
 "name": "sinais",
 "id": 3
 },
 ...
]
}
```

## Get a scientific area by id

GET /scientificareas/:id

- Request:
  - Path parameters:

- id - the id of the scientific area
- Body: none
- Response:
  - Success:
    - Status-code: 200 (OK)
    - Content-type: application/json
    - Body example:

```
{
 "name": "hardware/software",
 "subjects": [
 {
 "name": "LSD",
 "id": 1
 },
 {
 "name": "AC",
 "id": 2
 }
 ...
],
 "types": [
 {
 "name": "ATB",
 "id": "001-001"
 },
 {
 "name": "USBPort",
 "id": "001-002"
 }
 ...
]
}
```

- Error:
  - 404 (Not Found):

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Scientific Area Not Found",
 "detail": "The scientific area <id> doesn't exist",
 "status": 404
}
```

## Create scientific area

```
POST /scientificareas
```

- Request:

- Content-type: application/json

- Body:

```
{
 "name": "<the scientific area name>"
}
```

- Response:

- Success:

- Status-code: 201 (OK)

- Content-type: application/json

- Body example:

```
{
 "status": "The scientific area <scientific area> was created successfully",
 "identifiers": [
 1
],
 "uri": "/sigemadi/api/scientificareas/1"
}
```

- Error:

- 400 (Bad Request) - Invalid Body:

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Body isn't valid",
 "detail": "The request body contains invalid parameters",
 "status": 400
}
```

- 409 (Conflict):

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Scientific Area already exist",
 "detail": "The scientific area <sci_area_id> already exist",
 "status": 409
}
```

- 403 (Forbidden)

## Delete a scientific area

```
DELETE /scientificareas/:id
```

- Request:
  - Path parameter:
    - id - the id of the scientific area
  - Content-type: application/json
  - Body: none
- Response:
  - Success:
    - Status-code: 200 (OK)
    - Content-type: application/json
    - Body example:

```
{
 "status": "The scientific area <scientific area> was deleted successfully",
 "uri": "/sigemadi/api/scientificareas"
}
```

- Error:
  - 404 (Not Found):

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Scientifica Area Not Found",
 "detail": "The scientific area <sci_area_id> doesn't exist",
 "status": 404
}
```

- 403 (Forbidden)

## Subject management

### Get all subjects

```
GET /subjects
```

- Request:
  - Body: none
- Response:
  - Success:
    - Status-code: 200 (OK)
    - Content-type: application/json
    - Body example:

```
{
 "subjects": [
 {
 "name": "LSD",
 "id": 1
 },
 {
 "name": "AC",
 "id": 2
 },
 {
 "name": "LIC",
 "id": 3
 },
 ...
]
}
```

## Get a subject by id

```
GET /subjects/:id
```

- Request:
  - Path parameters:
    - id - the id of the subject
  - Body: none
- Response:
  - Success:
    - Status-code: 200 (OK)
    - Content-type: application/json
    - Body example:

```
{
 "name": "LSD",
 "sci_area": {
 "name": "hardware_software",
 "id": 1
 },
 "types": [
 {
 "name": "ATB",
 "id": "001-001"
 },
 {
 "name": "USBPort",
 "id": "001-002"
 },
 ...
]
}
```

- Error:
  - 404 (Not Found):

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exce
 "title": "Subject Not Found",
 "detail": "The subject <subject_id> doesn't exist",
 "status": 404
}
```



## Create subject

POST /scientificareas/:id/subjects

- Request:
  - Path parameter:
    - id - the id of the scientific area
  - Content-type: application/json
  - Body:

```
{
 "name": "<the subject name>"
}
```

- Response:
  - Success:
    - Status-code: 201 (OK)
    - Content-type: application/json
    - Body example:

```
{
 "status": "The subject <subject> was created successfully",
 "identifiers": [
 1
],
 "uri": "/sigemadi/api/subjects/1"
}
```
  - Error:
    - 400 (Bad Request) - Invalid Body:
 

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Body isn't valid",
 "detail": "The request body contains invalid parameters",
 "status": 400
 }
```
    - 400 (Bad Request) - Scientific Area is Geral :
 

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Scientific Area is Geral",
 "detail": "The scientific area <sci_area_id> is is Geral",
 "status": 400
 }
```
    - 404 (Not Found) - Scientific Area:
 

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Scientifica Area Not Found",
 "detail": "The scientific area <sci_area_id> doesn't exist",
 "status": 404
 }
```
    - 409 (Conflict) - Subject already exist:

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Subject Already Exist",
 "detail": "The subject <subject_id> already exist",
 "status": 409
}
```



- 403 (Forbidden)

## Delete a subject

`DELETE /scientificareas/:id/subjects/:id`

- Request:

- Path parameter:
  - id - the id of the subject
- Content-type: application/json
- Body: none

- Response:

- Success:

- Status-code: 200 (OK)
- Content-type: application/json
- Body example:

```
{
 "status": "The subject with the id <subject_id> was deleted successfully",
 "uri": "/sigemadi/api/subjects"
}
```

- Error:

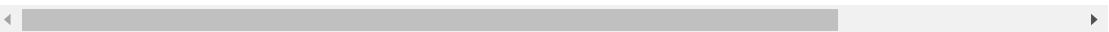
- 404 (Not Found) - Subject doesn't exist:

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Subject Not Found",
 "detail": "The subject <subject_id> doesn't exist",
 "status": 404
}
```



- 404 (Not Found) - Scientific Area doesn't exist:

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Scientific Area Not Found",
 "detail": "The scientific area <sci_area_id> doesn't exist",
 "status": 404
}
```



- 403 (Forbidden)

## Type management

---

### Get all types

GET /types

- Request:
  - Body: none
- Response:
  - Success:
    - Status-code: 200 (OK)
    - Content-type: application/json
    - Body example:

```
{
 "types": [
 {
 "name": "ATB",
 "id": "001-001"
 },
 {
 "name": "USBPort",
 "id": "001-002"
 },
 {
 "name": "PDS16",
 "id": "001-003"
 },
 ...
]
}
```

## Get a type by id

GET /types/:id

- Request:
  - Path parameters:
    - id - the id of the type
  - Body: none

- Response:
  - Success:
    - Status-code: 200 (OK)
    - Content-type: application/json
    - Body example:

```
{
 "name": "ATB",
 "sci_area":
 {
 "name": "hardware_software",
 "id": 1
 },
 "subjects": [
 {
 "name": "LSD",
 "id": 1
 },
 {
 "name": "AC",
 "id": 2
 },
 ...
]
}
```

- Error:
  - 404 (Not Found):

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exce
 "title": "Type Not Found",
 "detail": "The type <type_id> doesn't exist",
 "status": 404
}
```

## Create a new type

POST /scientificareas/:id/types

- Request:

- Path parameters:
  - id - the scientific area identifier

- Content-type: application/json

- Body:

```
{
 "name": "<the type>",
 "description": "<the description>",
 "subjects": [
 "<the subject identifier>",
 ...
]
}
```

- Response:

- Success:

- Status-code: 201 (OK)
- Content-type: application/json
- Body example:

```
{
 "status": "The type <type> was created successfully",
 "identifier": [
 "001-001"
],
 "uri": "/sigemadi/api/types/001-001"
}
```

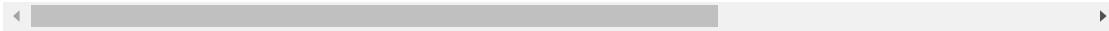
- Error:

- 400 (Bad Request) - Invalid Body:

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Body isn't valid",
 "detail": "The request body contains invalid parameters",
 "status": 400
}
```

- 400 (Bad Request) - Invalid Subjects:

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Invalid Subjects",
 "detail": "The collection of subjects [<subject_ids>] aren't valid",
 "status": 400
}
```



- 404 (Not Found):

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Scientific Area Not Found",
 "detail": "The scientific area <sci_area_id> doesn't exist",
 "status": 404
}
```



- 409 (Conflict) - Type in Scientific Area:

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Type Already Belongs To Sci Area",
 "detail": "The type <type_name> already belongs to scientific area with the id <sci_id>",
 "status": 409
}
```



- 403 (Forbidden)

## Delete a type

`DELETE /scientificareas/:id/types/:id`

- Request:
  - Path parameter:
    - id - the id of the type
  - Content-type: application/json
  - Body: none
- Response:
  - Success:
    - Status-code: 200 (OK)

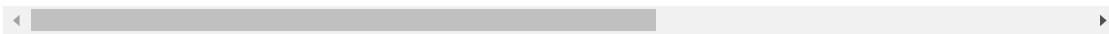
- Content-type: application/json
- Body example:

```
{
 "status": "The type <type> was deleted successfully",
 "uri": "/sigemadi/api/scientificareas/0/types"
}
```

- Error:

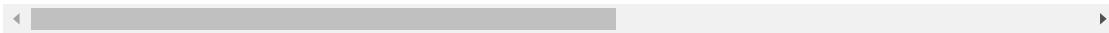
- 400 (Bad Request):

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Not All Materials are Available",
 "detail": "Not all specified materials have the state available to make the operation",
 "status": 400
}
```



- 404 (Not Found):

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Type Not Found in Scientific Area",
 "detail": "The type <type_id> doesn't exists in the specified scientific area",
 "status": 404
}
```



- 404 (Not Found - scientific area):

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Scientific Area Not Found",
 "detail": "The scientific area <sci_area_id> doesn't exist",
 "status": 404
}
```



- 403 (Forbidden)

## Statistics management

---

### Get number of damages per type

```
GET /statistics/damagespertype
```

- Request:
  - Body: none

- Response:

- Success:

- Status-code: 200 (OK)
    - Content-type: application/json
    - Body example:

```
{
 "data": [
 {
 "type_name": "ATB",
 "nr_damages": 2
 },
 {
 "type_name": "USBPort",
 "nr_damages": 4
 },
 {
 "type_name": "PDS16",
 "nr_damages": 12
 },
 ...
]
}
```

- Error:

- 403 (Forbidden)

## Get number of damaged materials per type

```
GET /statistics/damagedmaterials
```

Returns a list of damaged materials filtered by the allowed parameters.

### Example

This is an example where the search is narrowed by the damaged materials where type is equal to ATB (001-001).

```
GET /statistics/damagedmaterials?type=001-001
```

| Parameter | Available values                  | Details                                            |
|-----------|-----------------------------------|----------------------------------------------------|
| type      | 001-001 , 001-002 , 001-003 , ... | represents the damages solved(true) or not (false) |

- Request:
  - Body: none

- Response:
  - Success:
    - Status-code: 200 (OK)
    - Content-type: application/json
    - Body example:

```
{
 "data": [
 {
 "material_name": "ATB_0001",
 "nr_damages": 2
 },
 {
 "material_name": "ATB_0002",
 "nr_damages": 2
 },
 {
 "material_name": "ATB_0003",
 "nr_damages": 1
 },
 ...
]
}
```

- Error:
  - 403 (Forbidden)

## Get average minutes technicians take to solve damages

```
GET /statistics/averagetime-torepair
```

- Request:
  - Body: none

- Response:

- Success:
  - Status-code: 200 (OK)
  - Content-type: application/json
  - Body example:

```
{
 "data": [
 {
 "user_name": "Sr.Joao",
 "time": 23
 },
 {
 "user_name": "Sr.Antonio",
 "time": 23
 },
 ...
]
}
```

- Error:
  - 403 (Forbidden)

## Get top 10 of most used material types

GET /statistics/toptenmostusedmaterialtypes

- Request:
  - Body: none
- Response:
  - Success:
    - Status-code: 200 (OK)
    - Content-type: application/json
    - Body example:

```
{
 "data": [
 {
 "type_name": "ATB",
 "nr_times": 40
 },
 {
 "type_name": "PDS16",
 "nr_times": 27
 },
 ...
]
}
```

```
 ...
]
}
```

- Error:
  - 403 (Forbidden)

## Get top 10 of most used materials

GET /statistics/toptenmostusedmaterials

- Request:
  - Body: none
- Response:
  - Success:
    - Status-code: 200 (OK)
    - Content-type: application/json
    - Body example:

```
{
 "data": [
 {
 "material_name": "ATB_0001",
 "nr_times": 40
 },
 {
 "material_name": "PDS16_003",
 "nr_times": 27
 },
 ...
]
}
```

- Error:
  - 403 (Forbidden)

## Get number of requested hours per type

GET /statistics/numberofrequestedhoursperstype

- Request:
  - Body: none

- Response:
  - Success:
    - Status-code: 200 (OK)
    - Content-type: application/json
    - Body example:

```
{
 "data": [
 {
 "type_name": "ATB",
 "nr_hours": 400
 },
 {
 "type_name": "PDS16",
 "nr_hours": 270
 },
 ...
]
}
```

- Error:
  - 403 (Forbidden)

## Get number of requested hours per material

GET /statistics/numberofrequestedhourspermateriale

Returns a list of requested hours per material filtered by the allowed parameters.

### Example

This is an example where the search is narrowed by the number of requested hours per material where type is equal to ATB (001-001).

GET /statistics/numberofrequestedhourspermateriale?type=001-001

| Parameter | Available values                  | Details                                            |
|-----------|-----------------------------------|----------------------------------------------------|
| type      | 001-001 , 001-002 , 001-003 , ... | represents the damages solved(true) or not (false) |

- Request:
  - Body: none

- Response:
  - Success:
    - Status-code: 200 (OK)
    - Content-type: application/json
    - Body example:

```
{
 "data": [
 {
 "material_name": "ATB_0001",
 "nr_hours": 24
 },
 {
 "material_name": "ATB_0002",
 "nr_hours": 32
 },
 {
 "material_name": "ATB_0003",
 "nr_hours": 47
 },
 ...
]
}
```

- Error:
  - 403 (Forbidden)

## Get requests per type by day

`GET /statistics/requestspertypebyday`

Returns a list of requests per type by day filtered by the allowed parameters.

### Example

This is an example where the search is narrowed by the requests per type where type identifier is equal to 001-001 (ATB) and starts from day 2020-01-01 and ends on day 2020-03-04.

`GET /statistics/requestspertype?type=001-001&from=2020-01-01&to=2020-03-04`

| Parameter | Available values                  | Details                                            |
|-----------|-----------------------------------|----------------------------------------------------|
| type      | 001-001 , 001-002 , 001-003 , ... | represents the damages solved(true) or not (false) |

| Parameter | Available values | Details                    |
|-----------|------------------|----------------------------|
| from      | 2020-01-01 , ... | represents the initial day |
| to        | 2020-01-01 , ... | represents the end day     |

- Request:
  - Body: none

- Response:

- Success:
  - Status-code: 200 (OK)
  - Content-type: application/json
  - Body example:

```
{
 "data": [
 {
 "material_name": "ATB_0001",
 "nr_requests": 24,
 "year": "2020",
 "month": "03",
 "day": "01"
 },
 {
 "material_name": "ATB_0001",
 "nr_requests": 22,
 "year": "2020",
 "month": "03",
 "day": "02"
 },
 {
 "material_name": "ATB_0002",
 "nr_requests": 24,
 "year": "2020",
 "month": "03",
 "day": "01"
 },
 ...
]
}
```

- Error:
  - 403 (Forbidden)

## Get of requests per type by month

```
GET /statistics/requestspertypebymonth
```

Returns a list of requests per type by month filtered by the allowed parameters.

### Example

This is an example where the search is narrowed by the requests per type where type identifier is equal to 001-001 (ATB) and starts from month 2020-01 and ends on month 2020-03.

```
GET /statistics/requestspertype?type=001-001&from=2020-01&to=2020-03
```

| Parameter | Available values                  | Details                                            |
|-----------|-----------------------------------|----------------------------------------------------|
| type      | 001-001 , 001-002 , 001-003 , ... | represents the damages solved(true) or not (false) |
| from      | 2020-01 , ...                     | represents the initial day                         |
| to        | 2020-01 , ...                     | represents the end day                             |

- Request:
  - Body: none
- Response:
  - Success:
    - Status-code: 200 (OK)
    - Content-type: application/json
    - Body example:

```
{
 "data": [
 {
 "material_name": "ATB_0001",
 "nr_requests": 24,
 "year": "2020",
 "month": "03",
 "day": "01"
 },
 {
 "material_name": "ATB_0001",
 "nr_requests": 22,
 "year": "2020",
 "month": "03",
 "day": "02"
 },
 {

```

```

 "material_name": "ATB_0002",
 "nr_requests": 24,
 "year": "2020",
 "month": "03",
 "day": "01"
 },
 ...
]
}

```

- Error:
  - 403 (Forbidden)

## Get of requests per type by year

`GET /statistics/requestspertypebyyear`

Returns a list of requests per type by year filtered by the allowed parameters.

### Example

This is an example where the search is narrowed by the requests per type where type identifier is equal to 001-001 (ATB) and starts from year 2018 and ends on year 2020.

`GET /statistics/requestspertype?type=001-001&from=2018&to=2020`

| Parameter | Available values                  | Details                                            |
|-----------|-----------------------------------|----------------------------------------------------|
| type      | 001-001 , 001-002 , 001-003 , ... | represents the damages solved(true) or not (false) |
| from      | 2020 , ...                        | represents the initial day                         |
| to        | 2020 , ...                        | represents the end day                             |

- Request:
  - Body: none
- Response:
  - Success:
    - Status-code: 200 (OK)
    - Content-type: application/json
    - Body example:

```
{
 "data": [
 {
 "material_name": "ATB_0001",
 "nr_requests": 24,
 "year": "2020",
 "month": "03",
 "day": "01"
 },
 {
 "material_name": "ATB_0001",
 "nr_requests": 22,
 "year": "2020",
 "month": "03",
 "day": "02"
 },
 {
 "material_name": "ATB_0002",
 "nr_requests": 24,
 "year": "2020",
 "month": "03",
 "day": "01"
 },
 ...
]
}
```

- Error:
  - 403 (Forbidden)

## Damage management

---

### Get damages in inventory by parameters

GET /damages

Returns a list of damages filtered by the allowed parameters.

#### Example

This is an example where the search is narrowed by the damages to be repaired.

GET /damages?solved=false

| Parameter | Available values | Details                                            |
|-----------|------------------|----------------------------------------------------|
| solved    | true or false    | represents the damages solved(true) or not (false) |

- Request:
  - Body:none
- Response:
  - Success:
    - Status-code: 200 (OK)
    - Content-type: application/json
    - Body example:

```
{
 "damages": [
 {
 "id": "<the damage id>",
 "material": "001-001-0001",
 "state": "<the damage state>"
 },
 ...
]
}
```
  - Error:
    - 403 (Forbidden)

## Get damage by id

GET /damages/:id

- Request:
  - Path parameter: - id - the damage identifier
  - Body:none
- Response:
  - Success:
    - Status-code: 200 (OK)
    - Content-type: application/json

- Body example:

```
{
 "id": "<the damage id>",
 "material_name": "<material_name>",
 "material_id": "<material_id>",
 "description": "damaged switch",
 "report": "<text that reports the actual damage>",
 "start_date": "<the report date>",
 "close_date": "<the report date>",
 "time_to_resolve": "<the time to resolve damage>",
 "request": "<the request identifier>",
 "user": "<the user identifier>",
 "state": "<the damage state>"
}
```

- Error:

- 404 (Not Found):

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Damage Not Found",
 "detail": "The damage with the id <damage_id> doesn't exist",
 "status": 404
}
```

- 403 (Forbidden)

## Get damages states

GET /damagesstates

- Request:

- Body: none

- Response:

- Success:

- Status-code: 200 (OK)
- Content-type: application/json
- Body example:

```
{
 "states": [
```

```

 "resolvido",
 "por_resolver",
 "sem_solucao"
]
}

```

- Error:
  - 403 (Forbidden)

## Create new damage report

POST /materials/:mId/damages

- Request:
  - Path variables:
    - mld - the material identifier
  - Content-type: application/json
  - Body:
 

```
{
 "user": "<the user identifier>",
 "description": "<damage description>",
}
```

- Response:
  - Success:
    - Status-code: 201 (Created)
    - Content-type: application/json
    - Body example:

```
{
 "status": "New damaged report created",
 "identifier": 1,
 "uri": "/sigemadi/api/damages/1"
}
```

- Error:
  - 400 (Bad Request) - Invalid Body:

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Body isn't valid",
 "detail": "The request body contains invalid parameters",
 "status": 400
}
```



- 400 (Bad Request) - Invalid User:

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Invalid User",
 "detail": "User <user_id> doesn't exists or is invalid",
 "status": 400
}
```



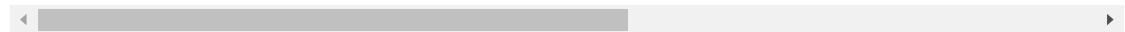
- 400 (Bad Request) - User is not a Student or Staff Member

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "User is Student or Staff Member",
 "detail": "The user <user_id> is a student or a staff",
 "status": 400
}
```



- 400 (Bad Request) - Material associated to a Request:

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Material already associated to a request",
 "detail": "The material <id> is already associated to a request",
 "status": 400
}
```



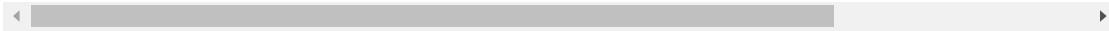
- 400 (Bad Request) - Material not Unit:

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Material Not Unit",
 "detail": "Material <material_id> is not unit",
 "status": 400
}
```



- 404 (Not Found):

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Material Not Found",
 "detail": "The material <material_id> doesn't exist",
 "status": 404
}
```



- 409 (Conflict):

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Damage already exist",
 "detail": "Opened damage with material <material_id> already exist",
 "status": 409
}
```



- 403 (Forbidden)

## Create damage intention

POST /requests/:id/materials/:mId/submission

- Request:

- Path variables:
  - id - the request identifier
  - mld - the material identifier
- Content-type: application/json
- Body:

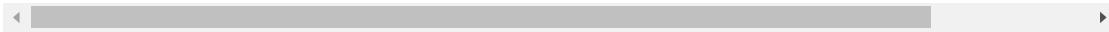
```
{
 "description": "<damage description>"
}
```

- Response:

- Success:
  - Status-code: 201 (Created)
  - Content-type: application/json

- Body example:

```
{
 "status": "Damage submission for the material <mId> on the request <id> was created :
 "uri": "/sigemadi/api/damages/:damageId"
}
```



- Error:

- 400 (Bad Request) - Invalid Body:

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Body isn't valid",
 "detail": "The request body contains invalid parameters",
 "status": 400
}
```



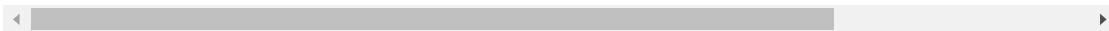
- 400 (Bad Request) - Material not Unit:

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Material Not Unit",
 "detail": "Material <material_id> is not unit",
 "status": 400
}
```



- 404 (Not Found) - Request:

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Request Not Found",
 "detail": "The request <request_id> doesn't exist",
 "status": 404
}
```



- 404 (Not Found) - Material:

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Material Not Found",
 "detail": "The material <material_id> doesn't exist",
 "status": 404
}
```

- 409 (Conflict):
 

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Damage Already Exist",
 "detail": "Opened damage with material <material_id> already exists",
 "status": 409
}
```
- 403 (Forbidden)

## Update existent damage report

PATCH /damages/:id

- Request:
  - Path parameter: - id - the damage identifier
  - Content-type: application/json
  - Body:

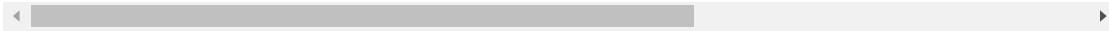
```
{
 "state": "<the damage state>",
 "report": "<text that reports the actual damage>"
}
```

- Response:
  - Success:
    - Status-code: 200 (OK)
    - Content-type: application/json
    - Body example:
  - Error:

```
{
 "status": "Damaged report was updated successfully",
 "uri": "/sigemadi/api/damages/1"
}
```

- 400 (Bad Request) - Invalid Body:

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Body isn't valid",
 "detail": "The request body contains invalid parameters",
 "status": 400
}
```



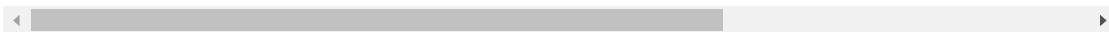
- 400 (Bad Request) - Invalid Damage State:

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Invalid Damage State",
 "detail": "State <state_name> is invalid",
 "status": 400
}
```



- 400 (Bad Request) - Damage is closed:

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "detail": "The damage <damage_id> is closed",
 "status": 400
}
```



- 404 (Not Found):

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Damage Not Found",
 "detail": "The damage with the id <damage_id> doesn't exist",
 "status": 404
}
```



- 403 (Forbidden)

## Cancel damage submission

```
DELETE /requests/:id/materials/:mId
```

- Request:

- Path variables:
    - id - the request identifier
    - mld - the material identifier
  - Content-type: application/json
  - Body: none
- Response:
- Success:
    - Status-code: 200 (OK)
    - Content-type: application/json
    - Body example:

```
{
 "status": "Damage submission for the material <mId> on the request <id> was cancelled",
 "uri": "/sigemadi/api/requests/:rId"
}
```
  - Error:
    - 400 (Bad Request) - Request is closed:

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Request is closed",
 "detail": "The request <request_id> is closed",
 "status": 400
}
```
  - 400 (Bad Request) - Material not Unit:
- ```
{
  "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
  "title": "Material Not Unit",
  "detail": "Material <material_id> is not unit",
  "status": 400
}
```
- 400 (Bad Request) - Material without Damage subscription:
- ```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
```

```
 "title": "Material Without Damage Subscription",
 "detail": "The material <material_id> of the request <request_id> does not have a damage subscription",
 "status": 400
 }
```



- 404 (Not Found) - Request:

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Request Not Found",
 "detail": "The request <request_id> doesn't exist",
 "status": 404
}
```



- 404 (Not Found) - Material:

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Material Not Found",
 "detail": "The material <material_id> doesn't exist",
 "status": 404
}
```



- 403 (Forbidden)

## Reservation management

---

### Get all incoming reservations

```
GET /reservations
```

- Request:
  - Body: none
- Response:
  - Success:
    - Status-code: 200 (OK)
    - Content-type: application/json
    - Body example:

```
{
 "reservations": [
 {
 "id": "<the reservation id>",
 "subject": "<the class subject>",
 "user": "<the user identifier>",
 "user_name": "<the user name>",
 "date": "<the date for the reservation>",
 "hour": "<the hour for the reservation>"
 },
 ...
]
}
```

- Error:
  - 403 (Forbidden)

## Get reservations of a user

GET /users/:id/reservations

- Request:
  - Path parameter:
    - id - the user identifier
  - Body: none
- Response:
  - Success:
    - Status-code: 200 (OK)
    - Content-type: application/json
    - Body example:

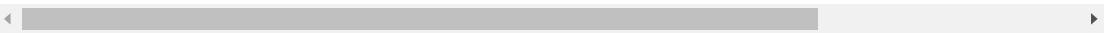
```
{
 "reservations": [
 {
 "id": "<the reservation id>",
 "subject": "<the reservation subject>",
 "groups": "<the number of groups>",
 "date": "<the date for the reservation>",
 "hour": "<the hour for the reservation>"
 },
 ...
]
}
```

```
]
 }
}
```

- Error:

- 404 (Not Found):

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "User Not Found",
 "detail": "The user <user_id> doesn't exist",
 "status": 404
}
```



- 403 (Forbidden)

## Get reservation by id

GET /reservations/:rsrvId

- Request:

- Path parameter:
  - rsrvId - the id of the reservation
- Body: none

- Response:

- Success:

- Status-code: 200 (OK)
  - Content-type: application/json
  - Body example:

```
{
 "user_id": "<the user identifier>",
 "user_name": "<the user name>",
 "user_photo": "<the user photo>",
 "nr_groups": "<number of groups>",
 "date": "<date of the reservation>",
 "hour": "<hour of the reservation>",
 "subject": "<the reservation subject>",
 "materials": [
 {
 "type_id": "001-001",
 "type_name": "ATB",
 "material": "Material 1",
 "quantity": 100,
 "unit": "kg",
 "notes": "Note 1",
 "status": "Available",
 "last_update": "2023-10-01T12:00:00Z"
 },
 {
 "type_id": "002-002",
 "type_name": "Tool A",
 "material": "Tool A",
 "quantity": 1,
 "unit": "unit",
 "notes": "Note 2",
 "status": "Available",
 "last_update": "2023-10-01T12:00:00Z"
 },
 {
 "type_id": "003-003",
 "type_name": "Tool B",
 "material": "Tool B",
 "quantity": 1,
 "unit": "unit",
 "notes": "Note 3",
 "status": "Available",
 "last_update": "2023-10-01T12:00:00Z"
 }
]
}
```

```

 "quantity": 1
 },
 {
 "type_id": "001-002",
 "type_name": "USBPort",
 "quantity": 1
 },
 {
 "type_id": "000-001",
 "type_name": "ATF750CL",
 "quantity": 2
 },
 ...
]
}

o Error:
 ■ 404 (Not Found):
 {
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Reservation Not Found",
 "detail": "The reservation <reservation_id> doesn't exist",
 "status": 404
 }

 ■ 403 (Forbidden)

```

## Create a new reservation

POST /reservations

- Request:
  - Content-type: application/json
  - Body:

```
{
 "user_id": "<the user id>",
 "subject": "<the class subject identifier>",
 "date": "<the date for the reservation>",
 "duration": "<the duration time>",
 "groups": "<number of groups in a class>",
 "materials": [
 {

```

```

 "type": "<the material type>",
 "quantity": "<the quantity per group>"
 },
 {
 "type": "<the material type>",
 "quantity": "<the quantity per group>"
 },
 {
 "type": "<the material type>",
 "quantity": "<the quantity per group>"
 },
 ...
]
}

```

- Response:

- Success:

- Status-code: 201 (OK)
    - Content-type: application/json
    - Body example:

```

{
 "status": "The reservation was created successfully",
 "identifiers": [
 1
],
 "uri": "/sigemadi/api/reservations/1"
}

```

- Error:

- 400 (Bad Request) - Invalid Body:

```

{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Body isn't valid",
 "detail": "The request body contains invalid parameters."
}

```

- 
- 400 (Bad Request) - Invalid User:

```

{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Invalid User",
 "detail": "User <user_id> doesn't exists or is invalid",
}

```

```
 "status": 400
 }
```

- 400 (Bad Request) - User is not a Teacher:

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "User Not a teacher",
 "detail": "The user <user_id> isn't a teacher",
 "status": 400
}
```

- 400 (Bad Request) - Invalid Subject:

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Invalid Subject",
 "detail": "The subject <subject_id> isn't valid",
 "status": 400
}
```

- 400 (Bad Request) - Invalid Type:

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Invalid Types",
 "detail": "Some of the types inserted are invalid",
 "status": 400
}
```

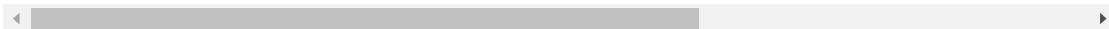
- 400 (Bad Request) - Requesting more materials then existent:

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Inserted More Materials then Existent",
 "detail": "Staff member <user_id> requested more material of type <type_id> (<quantity> <unit>)",
 "status": 400
}
```

- 400 (Bad Request) - Invalid duration time:

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Duration Time Invalid"
}
```

```
 "title": "Invalid Duration Time",
 "detail": "The target duration time <time> is invalid. You can only pass times of 1:00:00 or less.",
 "status": 400
 }
```



- 400 (Bad Request) - Reservation Conflict (Sum of materials more than existent):

```
{\n "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",\n "title": "Sum of Quantity of Conflict Reservation More than Existent Quantity",\n "detail": "The sum of quantity needed for a reservation, conflicting with another reservation, is more than the existent quantity.",\n "status": 400\n}
```



- 409 (Conflict):

```
{\n "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",\n "title": "Reservation Already Exist",\n "detail": "The user <user_id> already has a reservation for the date and time passed.",\n "status": 409\n}
```



- 403 (Forbidden)

## Delete a reservation

DELETE /reservations/:rsrvId

- Request:
  - Path parameter:
    - rsrvId - the reservation identifier
  - Body: none
- Response:
  - Success:
    - Status-code: 200 (OK)
    - Content-type: application/json
    - Body example:

```

{
 "status": "The reservation was deleted successfully",
 "uri": "/sigemadi/api/requests"
}

○ Error:
■ 404 (Not Found):

{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Reservation Not Found",
 "detail": "The reservation <reservation_id> doesn't exists",
 "status": 404
}

```



■ 403 (Forbidden)

## Request management

---

### Get all requests

GET /requests

Returns a list of requests filtered by the allowed parameters.

#### Example

This is an example where the search is narrowed by the user role.

GET /requests?date=2019-05-12

| Parameter | Available values | Details                               |
|-----------|------------------|---------------------------------------|
| date      | yyyy-MM-dd       | represents the date of the requests   |
| active    | true ; false     | represents if the requests are active |

- Request:

- Body: none

- Response:

- Success:

- Status-code: 200 (OK)
    - Content-type: application/json
    - Body example:

```
{
 "requests": [
 {
 "id": "<the request identifier>",
 "user_id": "<the user identifier>",
 "user_name": "<the user name>",
 "user_photo": "<the user photo>",
 "start_date": "<initial date of the request>",
 "start_hour": "<initial hour of the request>",
 "close_date": "<close date of the request>",
 "close_hour": "<close hour of the request>"
 },
 ...
]
}
```

- Error:

- 403 (Forbidden)

## Get request by id

GET /requests/:id

- Request:

- Path parameter:

- id - the id of the request

- Body: none

- Response:

- Success:

- Status-code: 200 (OK)
    - Content-type: application/json
    - Body example:

```

{
 "user_id": "<the user identifier>",
 "user_name": "<the user name>",
 "user_photo": "<the user photo>",
 "start_date": "<initial date of the request>",
 "start_hour": "<initial hour of the request>",
 "close_date": "<end date of the request if request is closed>",
 "close_hour": "<end hour of the request if request is closed>",
 "materials": [
 {
 "id": "001-001-0001",
 "name": "ATB 0001",
 "quantity": 1,
 "start_date": "<the start date>",
 "start_hour": "<the close date>",
 "close_date": "<the start hour if request is closed>",
 "close_hour": "<the close hour if request is closed>",
 "damage": { // if material suffered a damage submission
 "description": "description"
 }
 },
 {
 "id": "001-002-0001",
 "name": "PDS16 0001",
 "quantity": 1,
 "start_date": "<the start date>",
 "start_hour": "<the close date>",
 "close_date": "<the start hour if request is closed>",
 "close_hour": "<the close hour if request is closed>",
 "damage": { // if material suffered a damage submission
 "description": "description"
 }
 },
 {
 "id": "000-001-0000",
 "name": "ATF750CL",
 "quantity": 3,
 "start_date": "<the start date>",
 "start_hour": "<the close date>",
 "close_date": "<the start hour if request is closed>",
 "close_hour": "<the close hour if request is closed>",
 "damage": { // if material suffered a damage submission
 "description": "description"
 }
 },
 ...
]
}

```

- Error:

- 404 (Not Found):

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Request Not Found",
 "detail": "The request <request_id> doesn't exist",
 "status": 404
}
```



- 403 (Forbidden)

## Get history of requests of an user

GET /user/:id/requests

- Request:

- Path parameters:

- id - the user identifier

- Body: none

- Response:

- Success:

- Status-code: 200 (OK)

- Content-type: application/json

- Body example:

```
{
 "current": {
 "id": "<the request id>",
 "user_id": "<the user identifier>",
 "user_name": "<the user name>",
 "user_photo": "<the user photo>",
 "start_date": "<the starting date>",
 "start_hour": "<the starting hour>"
 },
 "older": [
 {
 "id": "<the request id>",
 "user_id": "<the user identifier>",
 "user_name": "<the user name>",
 "user_photo": "<the user photo>",
 "start_date": "<the starting date>",
 "start_hour": "<the starting hour>"
 }
]
}
```

```

 "start_hour": "<the starting hour>",
 "close_date": "<the closing date>",
 "close_hour": "<the closing hour>",
 },
 ...
]
}

```

- Error:

- 404 (Not Found):

```

{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "User Not Found",
 "detail": "The user <user_id> doesn't exist",
 "status": 404
}

```



- 403 (Forbidden)

## Get requests that can be notified

GET /types/: typeId/requests

- Request:

- Path parameters:
  - typeId - the type identifier
- Body: none

- Response:

- Success:
  - Status-code: 200 (OK)
  - Content-type: application/json
  - Body example:

```

{
 "requests": [
 {
 "id": "<identifier>",
 "user_id": "<user_id>",
 "quantity": "<quantity>",
 "start_date": "<start_date>",
 }
]
}

```

```

 "start_hour": "<start_hour>"

 },

 ...

]

}

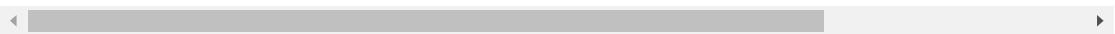
```

- Error:
  - 404 (Not Found):

```

{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Type Not Found",
 "detail": "The type <type_id> doesn't exist",
 "status": 404
}

```



- 403 (Forbidden)

## Create new request

POST /requests

- Request:

- Content-type: application/json
- Body:

```

{
 "user_id": "<the user identifier>",
 "materials_requested": [
 {
 "material_id": "<the material id>",
 "quantity": "<quantity of the material>"
 },
 {
 "material_id": "<the material id>",
 "quantity": "<quantity of the material>"
 },
 ...
]
}

```

- Response:

- Success:

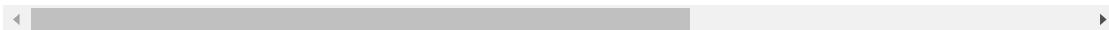
- Status-code: 201 (Created)
- Content-type: application/json
- Body example:

```
{
 "status": "Request created successfully",
 "identifier": [
 1
],
 "uri": "/sigemadi/api/requests"
}
```

- Error:

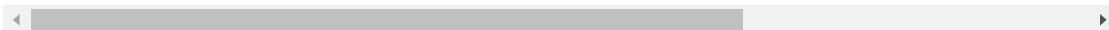
- 400 (Bad Request) - Invalid Body:

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Body isn't valid",
 "detail": "The request body contains invalid parameters",
 "status": 400
}
```



- 400 (Bad Request) - Invalid User:

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Invalid User",
 "detail": "User <user_id> doesn't exists or is invalid",
 "status": 400
}
```



- 400 (Bad Request) - Invalid Materials:

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Invalid Materials",
 "detail": "The collection of materials [<material_ids>] doesn't exist",
 "status": 400
}
```



- 400 (Bad Request) - Unit materials not Unitary:

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception"
```

```
 "title": "Unit materials not Unitary",
 "detail": "You've inserted unit materials with quantity more than one",
 "status": 400
 }
```

■ 400 (Bad Request) - Duplicated Materials:

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Duplicated Materials",
 "detail": "You've inserted duplicated in the request body",
 "status": 400
}
```

■ 400 (Bad Request) - User has already an Open Request:

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "User has already an open request",
 "detail": "The user <user_id> has already an open request",
 "status": 400
}
```

■ 400 (Bad Request) - Material exists in an Open Request:

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Materials exists in an open request",
 "detail": "You've inserted materials that exists in an open request",
 "status": 400
}
```

■ 403 (Forbidden)

## Add material to a request

PUT /requests/:id/materials

- Request:
  - Path parameters:
    - rId - the request identifier

- Content-type: application/json
- Body:

```
{
 "materials": [
 {
 "material_id": "<the material id>",
 "quantity": "<quantity of the material>"
 },
 {
 "material_id": "<the material id>",
 "quantity": "<quantity of the material>"
 }
]
}
```

- Response:

- Success:
    - Status-code: 200 (OK)
    - Content-type: application/json
    - Body example:

```
{
 "status": "Material added to the request successfully",
 "uri": "/sigemadi/api/requests"
}
```

- Error:
    - 400 (Bad Request) - Invalid Body:

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Body isn't valid",
 "detail": "The request body contains invalid parameters",
 "status": 400
}
```

- 400 (Bad Request) - Invalid Materials:

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Invalid Materials",
 "detail": "The collection of materials [<material_ids>] doesn't exist",
 "status": 400
}
```

- 400 (Bad Request) - Unit Materials not Unitary:
 

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Unit materials not Unitary",
 "detail": "You've inserted unit materials with quantity more than one",
 "status": 400
}
```
- 400 (Bad Request) - Duplicated Materials:
 

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Duplicated Materials",
 "detail": "You've inserted duplicated in the request body",
 "status": 400
}
```
- 404 (Not Found):
 

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Request Not Found",
 "detail": "The request <request_id> doesn't exist",
 "status": 404
}
```
- 403 (Forbidden)

## Delete material from a request

`DELETE /requests/:id/materials/:mId`

- Request:
  - Path parameters:
    - id - the request identifier
    - mId - the material identifier
  - Body:

```
{
 "quantity": "<a number for the quantity>"
}
```

- Response:

- Success:

- Status-code: 200 (OK)
    - Content-type: application/json
    - Body example:

```
{
 "status": "Material deleted from request successfully",
 "uri": "/sigemadi/api/requests/<id>"
}
```

- Error:

- 400 (Bad Request) - Invalid Body:

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Body isn't valid",
 "detail": "The request body contains invalid parameters",
 "status": 400
}
```

- 400 (Bad Request) - Quantity not Equal to One

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Quantity Not Equal To One",
 "detail": "You're trying to remove a unit material with a non unit quantity",
 "status": 400
}
```

- 400 (Bad Request) - Material Not Associated to Request:

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Material Not Associated to a Request",
 "detail": "The material <material_id> doesn't exists in the request <request_id> com",
 "status": 400
}
```

- 400 (Bad Request) - Request is closed:

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Request is closed",
 "detail": "The request <request_id> is closed",
 "status": 400
}
```

- 400 (Bad Request) - Removing more quantity the existent

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Remove More Quantity then Existent",
 "detail": "You're trying to remove from target request more material quantity than existent",
 "status": 400
}
```

- 404 (Not Found) - Request:

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Request Not Found",
 "detail": "The request <request_id> doesn't exist",
 "status": 404
}
```

- 404 (Not Found) - Material:

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Material Not Found",
 "detail": "The material <material_id> doesn't exist",
 "status": 404
}
```

- 403 (Forbidden)

## End request

PATCH /requests/:id

- Request:
  - Path parameters:
    - id - the request identifier
  - Body: none
- Response:
  - Success:
    - Status-code: 200 (OK)
    - Content-type: application/json
    - Body example:

```
{
 "status": "Terminate request successfully",
 "uri": "/sigemadi/api/requests"
}
```
  - Error:
    - 400 (Bad Request) - Request is closed:

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Request is closed",
 "detail": "The request <request_id> is closed",
 "status": 400
}
```
  - 404 (Not Found) - Request:

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Request Not Found",
 "detail": "The request <request_id> doesn't exist",
 "status": 404
}
```
- 403 (Forbidden)

## Send SMS to users

POST /sms

- Request:
  - Content-type: application/json
  - Body:
 

```
{
 "type": "<the material type>",
 "quantity": "<quantity of material of that type needed>"
}
```
- Response:
  - Success:
    - Status-code: 201 (OK)
    - Content-type: application/json
    - Body example:
 

```
{
 "status": "SMS sent to the users",
 "users": [
 {
 "identifier": "<the user 1 identifier>",
 "name": "<the user 1 name>",
 "photo": "<the user photo>",
 "phone_number": "<the user phone number>",
 "request_id": "<the user request that the material was in>"
 },
 {
 "identifier": "<the user 2 identifier>",
 "name": "<the user 2 name>",
 "photo": "<the user photo>",
 "phone_number": "<the user phone number>",
 "request_id": "<the user request that the material was in>"
 },
 ...
]
}
```
  - Error:
    - 400 (Bad Request) - Invalid Body:
 

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Body isn't valid",
 "detail": "The request body contains invalid parameters",
 "status": 400
}
```

- 400 (Bad Request) - Invalid Type:

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Invalid Type",
 "detail": "The type <type_id> isn't valid",
 "status": 400
}
```

- 400 (Bad Request) - Request more quantity than existent:

```
{
 "type": "https://github.com/carolina-santoss/PS_1920v_G12_SiGeMaDi/wiki/API-Exception",
 "title": "Request More Quantity then Exist",
 "detail": "You've requested more quantity for the material type <type_id> then existent",
 "status": 400
}
```

- 403 (Forbidden)