IPG Politécnico da Guarda
Polytechnic of Guarda

RELATÓRIO DE PROJETO

Licenciatura em Engenharia Informática

Vanderley Barreto do Espírito Santo Quaresma

setembro | 2019

Escola Superior de Tecnologia e Gestão

Instituto Politécnico da Guarda

PLATAFORMA DE GESTÃO DE PERFIS DE NEGÓCIO

VANDERLEY BARRETO DO ESPÍRITO SANTO QUARESMA

RELATÓRIO PARA A OBTENÇÃO DO GRAU DE LICENCIADO

EM ENGENHARIA INFORMÁTICA

Setembro/2019

Agradecimentos

Agradeço todos aqueles que de alguma forma ajudaram para que a realização deste projeto fosse possível, nomeadamente à empresa Armis e principalmente à minha orientadora, a Professora Doutora Maria Clara Silveira, pela disponibilidade e dedicação demostrado na realização do mesmo.

Agradeço também aos meus familiares que estiveram presentes durante a minha jornada neste curso, nomeadamente a minha irmã Adedmy Barreto Quaresma e meu cunhado Serguey De Sousa Fernandes.

Por fim, agradeço a todos os professores do curso pela dedicação e disponibilidade que demostraram ao longo do meu percurso neste curso.

Muito Obrigado.

Ficha de identificação | Elementos Identificativos

Aluno:	Vanderley Barreto Do Espírito Santo Quaresma
Email:	vanderleyquaresma@live.com.pt
Número:	1012331
Curso:	Licenciatura em Engenharia Informática
Ano letivo:	2018/2019
Estabelecimento de Ensino:	Escola Superior de Tecnologia e Gestão Instituto Politécnico da Guarda
Orientadora:	Professora Doutora Maria Clara Santos Pinto Silveira
Empresa:	Armis Sourcing, LDA

Resumo

O presente documento descreve o projeto realizado na empresa Armis, onde foi desenvolvida uma plataforma de gestão de perfis de identidades e acessos com a capacidade de analisar os dados referentes às funções dos utilizadores de uma organização.

Face ao desenvolvimento das tecnologias, atualmente a gestão de perfis de identidades e acessos têm sido cruciais no que toca à segurança dos utilizadores e integridade dos bens e informações de uma organização, onde sistema eficazes com recursos eficientes podem oferecer uma boa gestão das identidades e acessos, acrescendo assim facilidades e eficiências aos colaboradores na realização das suas tarefas.

A solução pretendida resume-se numa plataforma *web* de gestão e criação de perfis de negócios, onde o objetivo é examinar identidades existentes, atributos de identidade e respetivas permissões de cada utilizador de forma a encontrar padrões relativos ao acesso e funções de cada um. Padrões esses que permitirão clarificar as funções que um utilizador deverá ter quando registado no sistema, onde será possível adquirir uma gestão de segurança otimizada baseada na função de cada colaborador. A plataforma permitirá gerir as identidades na medida que será possível criar, editar, eliminar, importar e atribuir permissões.

Palavras-chave

Gestão de Perfis de Negócio, Gestão de Acessos, Segurança, SQL Server, ASP.NET, C#, Angular, Plataforma *web*.

Abstract

This document describes the project carried out at Armis, where an identity and access profile management platform was developed with the ability to analyze data relating to the users' roles in an organization.

In the face of technology development, identity and access profile management has now been crucial when it comes to user security and the integrity of an organization's assets and information, where effective systems with efficient resources can offer good identity and access management, thus adding facilities and efficiencies to employees in carrying out their tasks.

The intended solution is a web-based business management and profiling platform, where the goal is to examine each user's existing identities, identity attributes and permissions to find patterns for their access and roles. These standards will clarify the roles that a user should have when registered in the system, where it will be possible to acquire optimized security management based on the role of each employee. The platform will manage identities as it will be possible to create, edit, delete, import and assign permissions.

Palavras-chave

Business Profile Management, Acess Management, Security, SQL Server, ASP.NET, C#, Angular, *web* Platform.

Índice Geral

Agradecimentos i

Ficha de identificação | Elementos Identificativos ii

Resumo iii

Abstract iv

Índice de figuras viii

Índice de tabelas x

Lista de siglas e acrónimos xi

1 Introdução 1

1.1 Caraterização sumária da instituição 1

1.2 Motivação | Enquadramento 2

1.3 Descrição do problema 3

1.4 Objetivos 3

1.5 Plano de estágio | Etapas do Estágio 4

1.6 Estrutura do documento 5

2 Estado da Arte 6

2.1 Role Mining 6

2.2 Gestão de identidades 7

2.3 Trabalhos Relacionados 7

2.3.1 *Microsoft Identity Manager* - MIM 8

2.3.2 *IBM Security Acess Manager* – ISAM 9

2.3.3 Okta 10

3 Metodologia 13

3.1 Metodologia *Scrum* 13

3.2 Detalhe da metodologia no desenvolvimento da PGPN 14

4 Análise de requisitos 15

4.1 Diagrama de contexto 15

4.2 Diagrama de casos de uso 16
4.3 Descrição de casos de uso 17
4.3.1 Criar Aplicação 17
4.3.2 Editar Aplicação 17
4.3.3 Consultar Aplicação 18
4.3.4 Eliminar Aplicação 18
4.3.5 Importar Estruturas 19
4.3.6 Adicionar Permissões 20
4.3.7 Adicionar Funções 20
4.4 Diagrama de classes 21
4.5 Dicionário de dados 23
5 Tecnologias 26
5.1 HTML 26
5.2 CSS 26
5.3 Angular 26
5.4 Git 26
5.5 SQL Server 27
5.6 Microsoft SQL Server Management Studio 27
5.7 ASP.NET 27
5.8 Microsoft Power BI 27
6 Implementação 28
6.1 Role Mining 28
6.2 Gestão de identidades 32
6.2.1 Applications 32
6.2.2 Repositories 33
6.2.3 Users 34
6.2.4 Structures 35

6.2.5 Fuctional Roles 36
6.2.6 Procedimento da implementação de Gestão de Identidades 37
7 Verificação e validação 41
8 Conclusões 45
Bibliografia 46
A 1. Código 49
A 1.1. Código Structure Controller 49
A 1.2. Código Structure Component 53
A 1.3. Código HTML Structure 64
A 1.4. Código Structure Service 68
A 1.5. Codigo User servisse 69
A 1.6. Código HTML User 81
A 1.7. Ilustração da interface Home da PGPN 86
A 1.8. Ilustração da interface Role Minig da PGPN 87
A 1.9. Ilustração da interface Gestão Identidades da PGPN 88
A 2. Descrição de casos de uso 89
A 2.1. Criar Utilizador 89
A 2.2. Editar Utilizadores 89
A 2.3. Consultar Utilizadores 90
A 2.4. Eliminar Utilizador 90
A 2.5. Importar Utilizadores 91
A 2.6. Adicionar Funções a Utilizador 92

Índice de figuras

Figura 1 - Logótipo Armis Group 2
Figura 2 - Magic Quadrant for Acess Management [5] 8
Figura 3 - Atribuição de um utilizador a uma função [6] 9
Figura 4 – Diagrama geral ISAM [9] 10
Figura 5 - Automated user Lifecycle Management [11] 11
Figura 6 - Diagrama desenvolvimento Scrum 14
Figura 7- Diagrama de contexto da PGPN 15
Figura 8 - Diagrama de casos de uso da PGPN 16
Figura 9 - Diagrama de classes de Role Mining 22
Figura 10 - Diagrama de classes de Gestão de Identidades 22
Figura 11 - Vista Home 28
Figura 12 - Vista Role Mining 29
Figura 13 - Role Mining User Controller 30
Figura 14 - Modelo da classe “User” 31
Figura 15 - Esboço dos serviços da classe “User” 31
Figura 16 - Vista Applications 32
Figura 17 - Vista Detalhes Application 33
Figura 18 - Vista Repositories 33
Figura 19 - Vista Detalhes Repositories 34
Figura 20 - Vista Users 34
Figura 21 - Vista Detalhes Users 35
Figura 22 - Vista Structures 35
Figura 23 - Vista Detalhes Structures 36
Figura 24 - Vista Functional Roles 36
Figura 25 - Vista Detalhes Functional Roles 37
Figura 26 - Esboço do Modelo da Classe "Structure" 37
Figura 27 - Esboço do controlador da classe “Structure” 38
Figura 28 - Esboço do ficheiro HTML Eliminar Structure 39
Figura 29 - Esboço do ficheiro HTML Editar Structure 39
Figura 30 - Esboço do ficheiro HTML criar Structure 40
Figura 31 - Validação ao Criar Aplicação 41

Figura 32 - Código de validação ao criar Aplicações 41
Figura 33 - Validação de campos existentes 42
Figura 34 - Código de validação dos campos existentes 42
Figura 35 - Validação ao importar ficheiros 43
Figura 36 - Código de validação ao importar ficheiros 43
Figura 37 - Validação dos campos com dependências 44
Figura 38 - Código validação dos campos com dependências 44

Índice de tabelas

Tabela 1 - Tabela das atividades desenvolvida por cada membro da equipa 5
Tabela 2 - Comparação entres as plataformas estudadas 11
Tabela 3 - Descrição do caso de uso "Criar Aplicação" 17
Tabela 4 - Descrição de caso de uso "Editar Aplicação" 17
Tabela 5 - Descrição de caso de uso "Consultar Aplicação" 18
Tabela 6 - Descrição de caso de uso "Eliminar Aplicação" 19
Tabela 7 - Descrição de caso de uso "Importar Estruturas" 19
Tabela 8 - Descrição de caso de uso "Adicionar Permissões" 20
Tabela 9 - Descrição de caso de uso "Adicionar Funções" 21
Tabela 10 - Matriz CRUD 23
Tabela 11 - Dicionário de dados da tabela "Application" 23
Tabela 12 - Dicionário de dados da tabela "Structure" 24
Tabela 13 - Dicionário de dados da tabela "User" 24
Tabela 14 - Dicionário de dados da tabela "FunctionalRole" 25
Tabela 15 - Operações das tabelas 25

Lista de siglas e acrónimos

CSS Cascading Style Sheets

HTML HyperText Markup Language

IPG Instituto Politécnico da Guarda

ISAM IBM Security Access Manager

IOT Internet of Things

IBM International Business Machines

MIM Microsoft Identity Manager

Oracle Oracle Corporation

PGPN Plataforma De Gestão De Perfis De Negócio

TI Tecnologias de informação

UC Unidade Curricular

1 Introdução

O presente relatório documenta, no âmbito da Unidade Curricular (UC) Projeto de Informática do curso de Engenharia Informática, do Instituto Politécnico da Guarda, a realização de estágio na empresa Armis.

Ultimamente o tema da segurança nas organizações no que toca à gestão de identidades e acessos, tem sido alvo de grande preocupação por partes das mesmas. Preocupações essas resultantes da garantia da segurança das informações e acessos a recursos não autorizados.

Baseado nessas preocupações, visou-se a implementação de uma plataforma *web* para criação e gestão de perfis de negócio (PGPN), em que foi decidido seguir com uma solução capaz de analisar os dados referentes às funções dos utilizadores numa organização. O processo *Role Mining* faz a análise aos mapeamentos de cada utilizador com os recursos que o mesmo tem acesso, agilizando desta forma o processo de modificação de permissões na organização. Este projeto tem como finalidade obter uma gestão de segurança otimizada baseada na função de cada colaborador dentro da organização.

1.1 Caraterização sumária da instituição

A Armis Group, fundada em 2005, é uma entidade que oferece soluções na área de Tecnologias de Informação (TI) que tem como missão oferecer soluções de grandes dimensões aos clientes com ambição, eficiência e segurança cujo propósito é surpreender as expectativas dos mesmos [1]. A designação Armis deriva da palavra armas em latim, traduz-se na busca de tecnologias inovadoras e eficientes, alcançando a excelência no desenvolvimento e implementação de soluções digitais complexas.

Atendendo às necessidades de responder às especificidades de cada projeto e negócio com excelência, a Armis Group (Figura 1) é composta por:

- **Armis IT** - Armis Information Technology – Centra esforços na evolução e desenvolvimento de soluções ambiciosas à medida do cliente, como, Portais, Aplicações *Mobile, Business Intelligence*, Segurança e Gestão de Identidades, Testes e Certificação de Software;

- **Armis ITS** - Armis Intelligent Transport Systems – desenvolve soluções inteligentes na área dos transportes;

- **Armis Digital Sport** - Armis Digital Sport Technology – centra-se na conceção de soluções digitais especializadas na indústria desportiva;

- **Armis Sourcing** - Armis Sourcing Tecnology – Presta serviços de Sourcing através da alocação de recursos especializados aos nossos clientes.

Com a sede no Porto conta ainda com escritórios na cidade de Lisboa, São Paulo (Brasil) e Macau (China), com cerca de cento e vinte colaboradores e a existência de vários projetos internacionais em países como o Egito, Espanha, Angola e Irlanda [1].

Figura 1 - Logótipo Armis Group

1.2 Motivação | Enquadramento

Este projeto foi desenvolvido no âmbito da unidade curricular (UC) Projeto de Informática do curso de Engenharia Informatica do IPG, onde durante a realização do mesmo foi efetuado um estágio em colaboração com a Armis, com duração de duzentas e oitenta horas. O presente projeto consiste no desenvolvimento de uma solução na área de segurança, com finalidade de ser implementada em vários clientes. A "Plataforma de Gestão de Perfis de Negócio", refere-se a uma plataforma *web* com o propósito de criar e gerir perfis de negócio numa organização. A motivação do mesmo resume-se ao nível de importância que apresenta, tanto ao nível de desenvolvimento do hard e soft skills como o uso de tecnologia de extrema relevância, o caso da *framework angular*, que surgem como um incentivo.

1.3 Descrição do problema

Face à importância da segurança nas organizações e tendo em vista a implementação de uma plataforma de gestão de perfis de negócio, a Armis decidiu seguir como uma solução capaz de analisar os dados referentes às funções dos utilizadores nas organizações, onde através do processo *Role Mining* será possível analisar os mapeamentos de cada utilizador de acordo com o recurso que o mesmo tem acesso. A solução permitirá também através do processo Gestão de Identidades, criação e gestão de identidades e seus atributos, em que será possível atribuir funções e permissões às mesmas identidades.

A implementação de uma solução desse género trará benefícios como:

- Redução de custos de suporte à infraestrutura, reduzindo a sobrecarga administrativa necessária para manualmente atribuir funções e padrões de acessos.

- Aumento da eficiência e produtividade, a automatização dos processos de atribuição implicará rapidez a acessos dos recursos necessários por parte dos colaboradores, tornando-o produtivo.

- Melhorar a qualidade e precisão, reduzindo ou eliminando introdução manual dos dados propensa a erros pelos administradores do sistema, e maior precisão na atribuição de acessos consoante a função a desempenhar pelo colaborador.

- Melhorar a segurança e privacidade, garantindo acessos aos recursos certos e atribuição automática dos direitos de acesso para contas de utilizador.

1.4 Objetivos

O projeto tem como objetivo desenvolver uma aplicação web em equipa que suporte a criação e gestão de perfis de negócio numa organização. Onde será possível examinar identidades (identificação digital atribuída a uma entidade) existentes, atributos de identidade e respetivas permissões de um utilizador numa organização, de forma a encontrar padrões relativos ao acesso e funções de cada um. Padrões esses que permitirão clarificar as funções que um utilizador deverá ter quando registado no sistema.

A implementação desta solução irá oferecer à organização (“Armis”) oportunidades como:

- Melhorar o processo de análise de dados;
- Reduzir os custos administrativos, reduzindo a carga atual colocada na área de TI;
- Atribuição de perfis com maior precisão;
- Identificação de utilizadores que operam fora do padrão normal;
- Deteção e eliminação de funções ou privilégios de utilizadores redundantes;
- Eliminação de possíveis falhas de segurança e minimizar os riscos consequentes.

Concretamente, o projeto tem como objetivo a gestão e criação de identidades, nomeadamente quanto à criação, edição, eliminação, importação e atribuições de funções e permissões.

1.5 Plano de estágio | Etapas do Estágio

Apresenta-se a seguir o planeamento e as atividades das fases da implementação do projeto realizado em equipa.

1. Etapa (29/07 – 31/07): Preparação de ambiente (base de dados de perfis e ambiente);
   1.1. Desenho de base de dados.
   1.2. Preparação do ambiente de desenvolvimento (base de dados).
2. Etapa (01/08 – 14/08): Implementação – desenvolvimento Back-End;
   2.1. Criação do Projeto (*Visual Studio).*
   2.2. Criação de camada de acesso aos dados.
   2.3. Criação dos modelos e controladores.
3. Etapa (15/08 – 30/08): Implementação – Desenvolvimento Front-End;
   3.1. Integração do *framework Angular*.
   3.2. Criação de modelos, serviços e componentes.
   3.3. Autenticação.
   3.4. Criação dos layouts (HTML, CSS)
   3.5. Testes funcionais.
4. Etapa (02/09 – 03/09): Implementação num servidor da Armis, via IIS.

As atividades realizadas foram divididas por membros da equipa de desenvolvimento constituído por: Diogo Pinheiro, Vanderley Quaresma e André Rodrigues. Na tabela a seguir apresentam-se os membros da equipa e as classes (ver diagrama de classes na secção 4.4) que os mesmos implementaram no projeto.

Tabela 1 - Tabela das atividades desenvolvida por cada membro da equipa

Equipa	**Classes Implementadas de *Role Mining***	**Classes Implementadas de Gestão de Identidades**
Diogo Pinheiro	ApplicationRole, Repositories, entitlements	ApplicationRole, Repositories, entitlements
Vanderley Quaresma	User, Structure, Structure-User	User, Structure, Structure-User
André Rodrigues	Application	Application, FuncionalRole_ApplicationRole, FunctionalRole, FunctionalRole_User

1.6 Estrutura do documento

O presente documento é constituído por oito capítulos, em que no primeiro é feito uma introdução sobre sistema "Plataforma *web* para gestão de perfis de negócio" e os objetivos propostos para desenvolvimento da mesma, no segundo é feita um estudo sobre o estado da arte, onde são apresentadas uma breve descrição dos módulos da mesma e de algumas plataformas existentes que se enquadram com os propósitos da mesma solução. No capítulo seguinte, o terceiro, são apresentadas e descritas as metodologias utilizadas no desenvolver do projeto. No quarto, é realizada toda a análise de requisitos que identificam funcionalidades da solução a desenvolver. No quinto capítulo são apresentadas as tecnologias usadas para o desenvolvimento do sistema. No sexto é apresentada a implementação do sistema com ilustrações através de *screens* feitos à plataforma. No sétimo é apresentada algumas verificações e validações realizadas no sistema. E por fim é descrita a conclusão tiradas relativamente ao sistema desenvolvido.

2 Estado da Arte

De acordo com objetivos já especificados para a realização do projeto, neste capítulo são apresentadas as investigações realizadas incluindo as soluções já existentes, de forma a obter uma visão mais ampla e real, onde as informações obtidas podem ser benéficas no desenvolvimento da plataforma.

Inicialmente é apresentada uma breve descrição das partes constituintes do projeto, *Role Mining* e Gestão de Identidades, e de seguida é apresentado o levantamento das informações sobre a solução, indicando algumas soluções existentes. Finalizando com a comparação das funcionalidades das tecnologias existentes e as funcionalidades da proposta para o projeto.

2.1 Role Mining

O *Role Mining, Entitlements Discovery* e *Role Consolidation* são módulos que podem ser usados para preencher o *Identity Warehouse* (Armazéns de identidades) com a combinação certa de utilizadores e respetivas funções [2].

O processo de *Role Mining* descobre relacionamentos entre utilizadores com base nas permissões de acesso semelhantes, que podem ser agrupadas logicamente para formar uma função [2].

Na implementação do presente projeto este processo é uns dos módulos a implementar, onde será possível analisar os mapeamentos de cada utilizador com cada recurso que o mesmo tem acesso, agilizando desta forma o processo de modificação de permissões na organização. Num ambiente de negócio, as funções de cada utilizador são definidas consoante a competência, autoridade e responsabilidade do cargo.

2.2 Gestão de identidades

A gestão de identidades e acesso é um processo crítico no que toca à proteção de sistemas, dados e aplicações da organização contra acessos não autorizados. E uma solução bem-sucedida da mesma pode ser um acréscimo ao negócio na redução dos custos de gestão de TI [3]. A gestão de identidades consiste no processo da definição de uma identidade para cada entidade (pessoas, hardwares ou até mesmo processos previamente definidos), associando a mesma determinados atributos, onde através dessa identidade é possível autenticar e autorizar acessos a aplicativos, sistemas ou redes.

Em seguida são salientadas algumas funcionalidades chaves que podem contribuir para bom funcionamento de uma possível solução [3]:

- Identificação e monitorização dos utilizadores, definindo quem tem acesso a dados.
- Descoberta e proteção de dados sensíveis, efetuando o tracking (mapeamento) dos utilizadores no acesso aos dados.
- Bloqueio de acessos não autorizados, fortalecendo aplicações que contém dados sensíveis e criando exceções privilegiadas de acesso.

No presente projeto este módulo vai ser implementado, para gerir os utilizadores de acordo com os recursos que os mesmos terão acesso, onde será possível criar perfis, atribuir permissões e funções.

2.3 Trabalhos Relacionados

Neste subcapítulo é feita uma análise de algumas soluções de gestão de perfis de identidades e acessos com foco nos líderes dos últimos anos como, a Microsoft, Oracle, IBM, Okta e a SAP. Analisadas nos estudos feitos em 2018 pela Gartner, empresa líder em pesquisa e aconselhamento, com missão de fornecer orientações aos seus clientes consoante a escolha de tecnologias. A Gartner desenvolveu uma metodologia nomeada Gartner Magic Quadrant ilustrada na figura 2, que permite visualizar a classificação de várias tecnologias do mesmo ramo, face à integridade de visão e a habilidade de execução, onde as mesmas são classificadas em quatro quadrantes: Leaders (lideres) – alta integridade de visão e alta habilidade de execução, Challengers (desafiadores) – baixa integridade de visão e alta habilidade de execução, Niche Players – baixa integridade de

visão e baixa habilidade de execução e os Visionaries (visionários) – alta integridade de visão e baixa habilidade de execução [4].

Figura 2 - Magic Quadrant for Acess Management [5]

2.3.1 *Microsoft Identity Manager* - MIM

Microsoft Identity Manager (MIM) permite às organizações gerir os utilizadores, credencias e acessos. Através da MIM as organizações podem simplificar a gestão do ciclo de identidades com fluxos de trabalho automatizados, regras de negócio e uma integração fácil com plataformas heterogéneas e todo Data Center [5].

Com o propósito de estender as funcionalidades (recursos) do MIM, a Microsoft lançou o Microsoft BHOLD para adcionar o controle de acesso baseado em funções que cada utilizador possui, permitindo às organizações definir funções de utilizadores e parâmetros de controlo de acesso a aplicações e dados confidencias, incluindo serviços e ferramentas que simplificam o modelo das relações entre funções dentro da organização. Sendo elas mapeadas para permissões, e para verificar se as funções estão corretamente aplicadas

aos utilizadores [6]. A figura 3 ilustra como um utilizador individual pode ser atribuído a uma função no BHOLD.

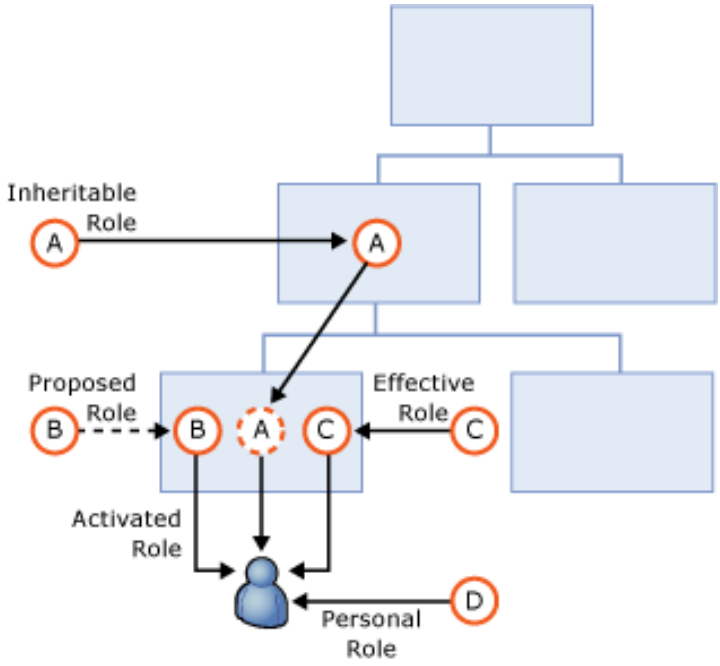


Figura 3 - Atribuição de um utilizador a uma função [6]

No diagrama, a função "A" é atribuída a uma unidade organizacional como uma função herdável, o que implica que seja herdada por suas unidades organizacionais membros e todos os utilizadores pertencentes a elas, a função "B" é atribuída como uma função proposta para uma unidade organizacional onde é ativa antes que o utilizador naquela unidade possa ser autorizado com permissões da função, a função "C" é uma função efetiva, logo, as suas permissões se aplicam automaticamente a todos os utilizadores da unidade. E função "D" são vinculadas diretamente aos utilizadores, logo suas permissões se aplicam imediatamente a esses utilizadores [6].

Atualmente a solução está a sofrer uma migração para Azure AD ("Azure Active Dirrectory") em que passa a fornecer revisões de acesso que substituem alguns recursos do BHOLD [7].

2.3.2 *IBM Security Acess Manager* – ISAM

O IBM Security Access Manager "ISAM" permite às organizações manter a segurança à medida que vão adotando novas tecnologias como web, mobile, IoT e tecnologias em

nuvem, fornecendo uma gestão simplificada de acesso e autenticação. Pode ser implementado localmente ou num dispositivo virtual [8].

O ISAM ajuda a encontrar um equilíbrio entre a segurança e usabilidade através do uso de acesso baseado em risco, *single login*, controlo de gestão de acesso integrado, federação de identidades e autenticação de múltiplos fatores móveis [9]. A figura seguinte ilustra o organograma geral do ISAM.

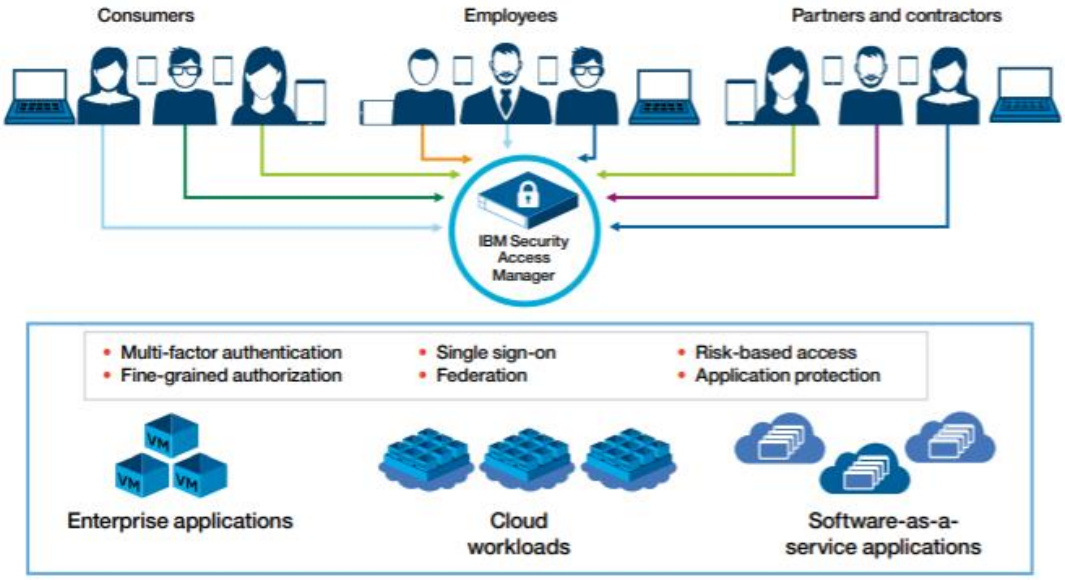


Figura 4 – Diagrama geral ISAM [9]

2.3.3 Okta

Ao longo de últimos anos a Okta tem desenvolvido soluções para suporte de gestão de identidades e acessos (*Identity Acess Management*), uma dela à *Lifecycle Management* (gestão de ciclo de vida) para gestão de ciclo de vida das identidades, com propósito de automatizar as tarefas de autorização e aprovisionamento de acessos à aplicativos, conforme ilustra a figura 5. Onde os utilizadores têm acesso instantâneo a tudo a que têm permissão, após serem atribuídos a um grupo ou a uma função [10]. A solução consiste em:

- Detetar automaticamente os atributos de diferentes utilizadores em diferentes aplicativos, juntamente com permissões de acesso que têm a esses aplicativos;
- Combinar grupos de diretórios com grupos de aplicativos, atribuir e revogar licenças, e criar um processo de exclusão personalizado;
- Criar e desativar contas em aplicativos;

- Acesso e aprovisionamento de aplicativos vinculados aos estados do ciclo de vida;
- Importar os ficheiros CSV para qualquer aplicativo.

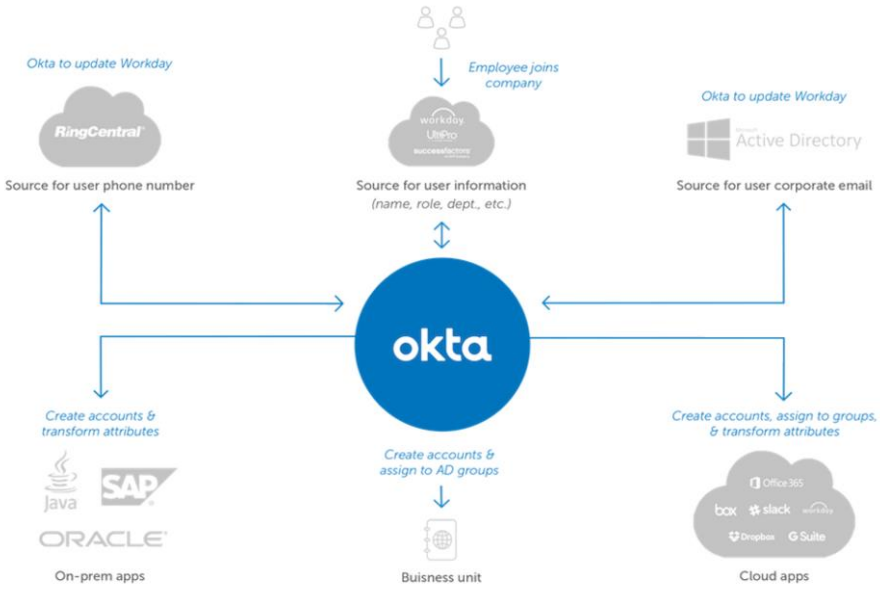


Figura 5 - Automated user Lifecycle Management [11]

Apos a descrição de algumas soluções existentes, em seguida é realizado uma comparação de funcionalidades relevantes das mesmas relativamente à realização do presente projeto, conforme mostra a tabela 1.

Tabela 2 - Comparação entres as plataformas estudadas

	MIM	**ISAM**	**Okta**	**PGPN**
Criação de perfis de negócio	×		×	×
Gestão de identidades e acessos	×	×	×	×
Atribuição de funções à utilizadores	×			×
Atribuição de permissões por funções	×	×	×	×
Importação de ficheiros CSV	×		×	×
Integração com outros sistemas	×	×	×	×

Atribuição de utilizadores à grupos (por exemplo a estrutura organizacional)	×		×	×
Gestão de ciclo de vida das identidades	×		×	×
Website responsivo	×	×	×	×

Após a comparação das soluções na Tabela 1 acima apresentada, chegou-se à conclusão de que o MIM (*Microsoft Identity Manager*) se encaixa mais em todos os objetivos propostos para o desenvolvimento da plataforma, e a implementação da mesma deve-se ao facto de ser uma solução com custos reduzidos para realização das tarefas da empresa em relação ao MIM.

3 Metodologia

No processo desenvolvimento de um sistema é necessário seguir certas normas com finalidade de ajudar nas realizações das tarefas, onde podem ser utilizados diversos tipos de metodologias de desenvolvimento de software existentes, tais como, metodologia estruturada, metodologia orientada a objetos e metodologias de desenvolvimento ágil [12].

Para o desenvolvimento da plataforma, a empresa optou pela escolha da metodologia de desenvolvimento ágil, nomeadamente a metodologia *Scrum,* face ao seu enquadramento a problemas adaptativos e complexos, o que vai de acordo com a realidade da mesma.

3.1 Metodologia *Scrum*

Scrum é uma *framework* da metodologia ágil para desenvolvimento iterativo e incremental de software, onde permite gestão de atividades complexas, ao mesmo tempo que se produz de uma maneira criativa, produtos com o maior valor possível. Incentiva as equipas a organizarem-se, quando as mesmas trabalham num problema específico, de forma a observarem e refletirem sobre as falhas e as conquistas, com propósito de melhorarem [13].

A metodologia é composta por alguns termos que compoêm o processo da mesma como [14]:

- *Product Backlog* – lista ordenada das funcionalidades a serem desenvolvidas no projeto, gerida pelo *Product Owner*.
- *Sprint Planing* – reunião onde são inspecionadas as *product backlog* relevantes a serem desenvolvidas e duração do desenvolvimento.
- *Sprint Backlog* – visão geral do trabalho a ser desenvolvido para atingir a meta de uma *sprint*, consoante a *product backlog* apresentados.
- *Sprint* – Período não superior a trinta dias, onde o projeto é desenvolvido.
- *Daily Scrum* – reuniões diárias com duração de 15 minutos, onde a equipa de desenvolvimento revê e planeia as tarefas do dia seguinte, as atualizações são refletidas no *sprint backlog*.
- *Sprint Review* – reunião realizadas na conclusão do *sprint* para a revisão do trabalho concluído por parte das equipas com os *stakeholders*, para avaliarem o progresso do mesmo.

- *Sprint Retrospective* – reunião com duração de três horas ou menos, onde é feita a reflecção sobre o *sprint* passado e planeamento de melhorias a serem implementadas no próximo *sprint*.

No desenvolver do projeto cada sprint contará como um incremento, repetindo várias vezes até ser atingido o propósito final. A figura seguinte ilustra o processo de uma sprint.

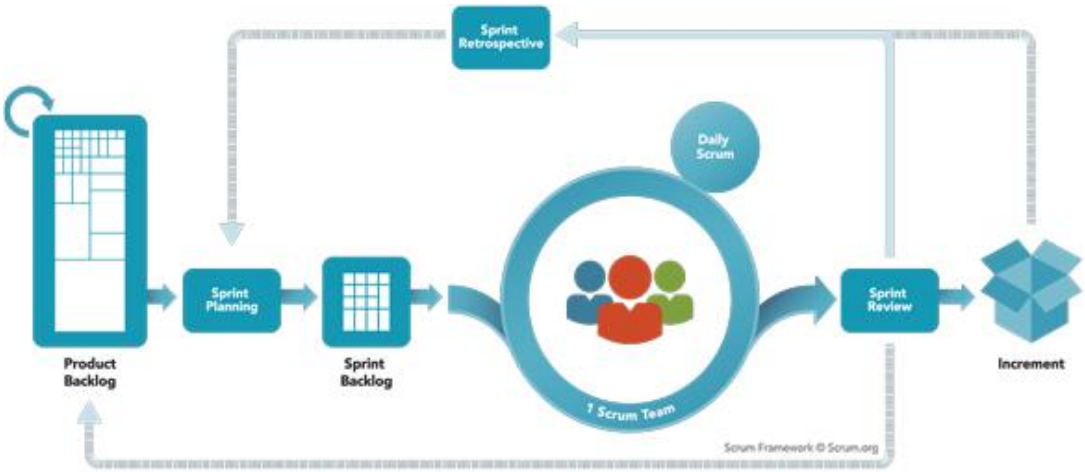


Figura 6 - Diagrama desenvolvimento Scrum

3.2 Detalhe da metodologia no desenvolvimento da PGPN

Numa primeira fase foram "levantadas" as *product backlog* (listas das funcionalidades) pelo gestor do projeto (Francisco Falcão), de seguida no *sprint planing*, com duração de alguns minutos, foram realizadas inspeções das product backlog levantadas pelos mesmo. Para identificar as prioridades no processo seguinte "*sprint backlog*" foi analisado o trabalho a desenvolver e a duração do mesmo, para atingir uma sprint, com base nas listas das funcionalidades apresentadas *(product backlog)* pela equipa de desenvolvimento e gestor do projeto. Após conclusão das fases de planeamento, prosseguiu-se para a implementação (arranque da sprint) pela equipa de desenvolvimento, onde foram implementadas as *product backlog*, e durante essa fase foram realizadas reuniões diárias com o gestor do projeto para revisão das funcionalidades já implementadas e as atividades seguintes a realizar *(daily scrum)*.

Na conclusão da sprint foram realizadas reuniões (gestor de projeto e a equipa) para rever o trabalho desenvolvido *(sprint review)* e melhoria a ser implementada no sprint seguinte *(sprint retrospective)*. Durante o desenvolvimento da PGPN foram realizados alguns sprints de acordo com as propostas que iam sendo apresentas pelo gestor do projeto.

4 Análise de requisitos

A análise de requisitos de um sistema é um processo importante no desenvolvimento do mesmo, de forma a estruturar, planear e recolher informações antes da implementação, minimizando assim erros que podem surgir após a conclusão do mesmo.

Durante a fase inicial deste projeto, a equipa na Armis, determinou vários requisitos que identificam funcionalidades da solução a desenvolver. Nas secções seguintes serão descritas essas funcionalidades.

4.1 Diagrama de contexto

O diagrama de contexto consiste em apresentar as relações estabelecidas entre o sistema e o meio ambiente, apresenta uma visão geral das principais funções onde as entradas do sistema são produzidas pelas entidades externas e as saídas pelo sistema.

A figura 7 ilustra, o diagrama de contexto que apresenta alguns dos fluxos do sistema PGPN, onde a entidade externa é o utilizador autenticado que desempenha o papel de administrador.

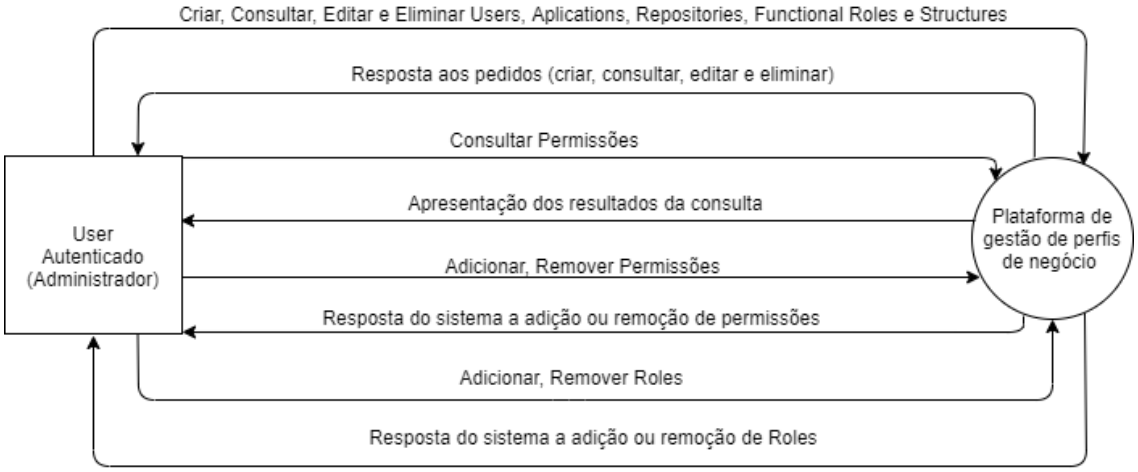


Figura 7- Diagrama de contexto da PGPN

4.2 Diagrama de casos de uso

A figura seguinte apresenta o diagrama de casos de uso que ilustra as funcionalidades do sistema, enquanto a sua interação com os atores. Os casos de uso gerir utilizadores, aplicações, repositórios, estruturas e funções consistem em: criar, consultar, editar e eliminar os mesmos.

Figura 8 - Diagrama de casos de uso da PGPN

O ator CSV, consiste num ficheiro com os dados das classes (Application, Repository, Structure, User, Functional Role) que é exportado exteriormente de uma máquina.

4.3 Descrição de casos de uso

Neste capítulo o objetivo é, de uma forma detalhada explicar os casos de uso ilustrados na figura 8. Os casos de uso gerir utilizadores e gerir aplicação consiste na criação, na consulta, na edição e na eliminação dos atributos dos mesmos. Em seguida são descritos esses mesmos casos de uso.

4.3.1 Criar Aplicação

Nesta secção apresenta-se a descrição de um caso de uso para criar um aplicativo, conforme mostra a tabela 3.

Tabela 3 - Descrição do caso de uso "Criar Aplicação"

Nome	Criar Aplicação
Descrição	O caso de uso consiste na criação de uma aplicação
Pré-condição	Login válido
Caminho Principal	1. O ator seleciona o ícone “New”; 2. O sistema apresenta o formulário para criar a aplicação; 3. O ator introduz os campos: Application ID (no caso é o external ID que é introduzido), Name, Description, Repository; 4. O ator seleciona a opção “Create”; 5. O sistema cria uma nova aplicação e apresenta na aba das aplicações.
Caminho Alternativos	4 a) O sistema devolve um alerta indicando que não estão corretos. Campos que só permitem números e estava a introduzir carateres, vice-versa.
Suplementos ou adornos	Verificação se criação é realizada com os campos devidamente preenchidos: no campo Application ID (external ID) só pode introduzir carateres com tamanho 50, no Name só carateres com tamanho máximo 50 e no Description só carateres com tamanho máximo 100, e de forma não criar campos nulos.

4.3.2 Editar Aplicação

A tabela seguinte consiste na descrição de caso de uso para editar uma aplicação.

Tabela 4 - Descrição de caso de uso "Editar Aplicação"

Nome	Editar Aplicação
Descrição	O caso de uso consiste em editar uma aplicação
Pré-condição	Login válido

Caminho Principal	1. O ator seleciona a secção do campo a editar em “Details”, “Modify”; 2. O sistema apresenta o formulário com os campos que permite editar: Name, Description, Repository; 3. O ator introduz os dados nos campos respetivos e seleciona a opção “Modify”; 4. O sistema guarda e dá sucesso.
Caminho Alternativos	3 a) O sistema devolve um alerta indicando que não estão corretos. os campos que só permitem números e estava a introduzir carateres, vice-versa.
Suplementos ou adornos	Verificação se ao editar os campos são devidamente preenchidos: no campo Name só carateres com tamanho máximo 50 e no Description só carateres com tamanho máximo 100.

4.3.3 Consultar Aplicação

A tabela seguinte consiste na descrição de um caso de uso pra consultar aplicações.

Tabela 5 - Descrição de caso de uso "Consultar Aplicação"

Nome	Consultar Aplicação
Descrição	O caso de uso consiste na consulta de uma aplicação
Pré-condição	Login válido
Caminho Principal	1. O ator seleciona na aba “Application”; 2. O sistema apresenta a lista das aplicações com ordenação alfabética e crescente; 3. O ator pesquisa por nome; 4. O sistema apresenta a lista consoante ao nome pesquisado.
Caminho Alternativos	3 a) Não existe o nome pesquisado.

4.3.4 Eliminar Aplicação

A tabela seguinte consiste na descrição de caso de uso para eliminar um aplicativo.

Tabela 6 - Descrição de caso de uso "Eliminar Aplicação"

Nome	Eliminar Aplicação
Descrição	O caso de uso consiste na eliminação de uma aplicação
Pré-condição	Login válido
Caminho Principal	1. O ator seleciona na aba “Application”; 2. O sistema apresenta a lista de aplicação; 3. O ator seleciona a aplicação e o botão “Delete”; 4. O sistema apresenta a mensagem para confirmar a eliminação; 5. O ator seleciona a opção “Confirm”; 6. O sistema finaliza a eliminação;
Caminho Alternativos	4 a) O sistema devolve um alerta em que os dados a eliminar têm dependências com a tabela “Application Role” e não pode eliminar.
Suplementos ou adornos	Verificação se a eliminação é realizada com os campos sem dependências.

4.3.5 Importar Estruturas

A tabela seguinte consiste na descrição de caso de uso para importar ficheiros CSV com dados correspondentes a estrutura organizacional.

Tabela 7 - Descrição de caso de uso "Importar Estruturas"

Nome	Importar Estruturas
Descrição	O caso de uso consiste na importação dos dados da Estrutura dum ficheiro CSV.
Pré-condição	Login válido
Caminho Principal	1. O ator seleciona a aba “Estruturas”; 2. O sistema apresenta vista das Estruturas; 3. O ator seleciona o ícone “Import File” 4. O sistema apresenta a janela de ambiente de trabalho para seleção do ficheiro; 5. O ator seleciona o ficheiro CSV que pretende importar; 6. O sistema atualiza a vista das estruturas com os dados importados;

Caminho Alternativos	6 a) O sistema devolve um alerta, em que os campos do ficheiro a importar não são compatíveis com os do sistema. Por exemplo o campo “Structure ID” só permite números e o ficheiro a importar conter no campo carateres.
Suplementos ou adornos	Verificação se a importação é realizada com os campos no ficheiro devidamente corretos, por exemplo na coluna Structure ID só deve conter números.

4.3.6 Adicionar Permissões

Este caso de uso tem como objetivo adicionar permissões, isto é, consoante a classe selecionada se pode adicionar permissões. Por exemplo na classe utilizadores o adicionar permissões consiste na adição de “ApplicationRoles” a que o utilizador tem permissão.

Tabela 8 - Descrição de caso de uso "Adicionar Permissões"

Nome	Adicionar Permissões
Descrição	O caso de uso consiste no adicionar permissões a utilizadores.
Pré-condição	Login válido
Caminho Principal	1. O ator seleciona a aba de utilizadores; 2. O sistema apresenta vista de utilizadores; 3. O ator seleciona o utilizador e o botão “Details” 4. O sistema apresenta a vista de detalhes do utilizador; 5. O ator seleciona a aba de permissões; 6. O sistema apresenta a lista de permissões; 7. O ator clica no botão “+”, seleciona a permissão e clica no botão “Add”; 8. O sistema atualiza lista das permissões com as permissões adicionadas.
Caminho Alternativos	7 a) O ator cancela a operação de adicionar.

4.3.7 Adicionar Funções

Este caso de uso tem objetivo adição de uma função, onde o ator adiciona funções a um utilizador ou á estruturas organizacionais que o mesmo pertence.

Tabela 9 - Descrição de caso de uso "Adicionar Funções"

Nome	Adicionar Funções
Descrição	O caso de uso consiste no adicionar funções a utilizadores ou à estrutura, por exemplo se utilizador pertence a uma unidade organizacional é-lhe adicionado imediatamente as funções pertencentes a essa unidade ou-lhe é adicionado diretamente funções independente da unidade organizacional a que pertencente.
Pré-condição	Login válido
Caminho Principal	1. O ator seleciona a aba pretendida; 2. O sistema apresenta vista da aba; 3. O ator seleciona a secção do campo e clica no botão “Details” 4. O sistema apresenta a vista de detalhes; 5. O ator seleciona a aba das funções; 6. O sistema apresenta lista das funções; 7. O ator clica no ícone “+”, seleciona a função e clica no botão “Add”; 8. O sistema atualiza secção das funções com as funções adicionadas.
Caminho Alternativos	7 a) O ator cancela a operação adiciona.

4.4 Diagrama de classes

O diagrama de classes é uma representação da estrutura e relações entre as classes que servem de modelo para objectos. Apresenta as interações entre as classes e as resposabilidades das mesmas na realização das operações solicitadas pelos atores.

Nas figuras seguintes estão ilustrados os diagramas de classes para o sistema em desenvolvimento, onde se pode visualizar os relacionamentos entre classes, e as operações que podem ser realizadas nas mesmas. Como o projeto é composto por duas partes também foram elaborados dois diagramas de classes para o mesmo, pelo motivo dos dados de *Role Mining* seram utilizados para carregar no *dashboard* do Power BI. A figura 9 ilustra o diagrama de classes de *Role Mining*.

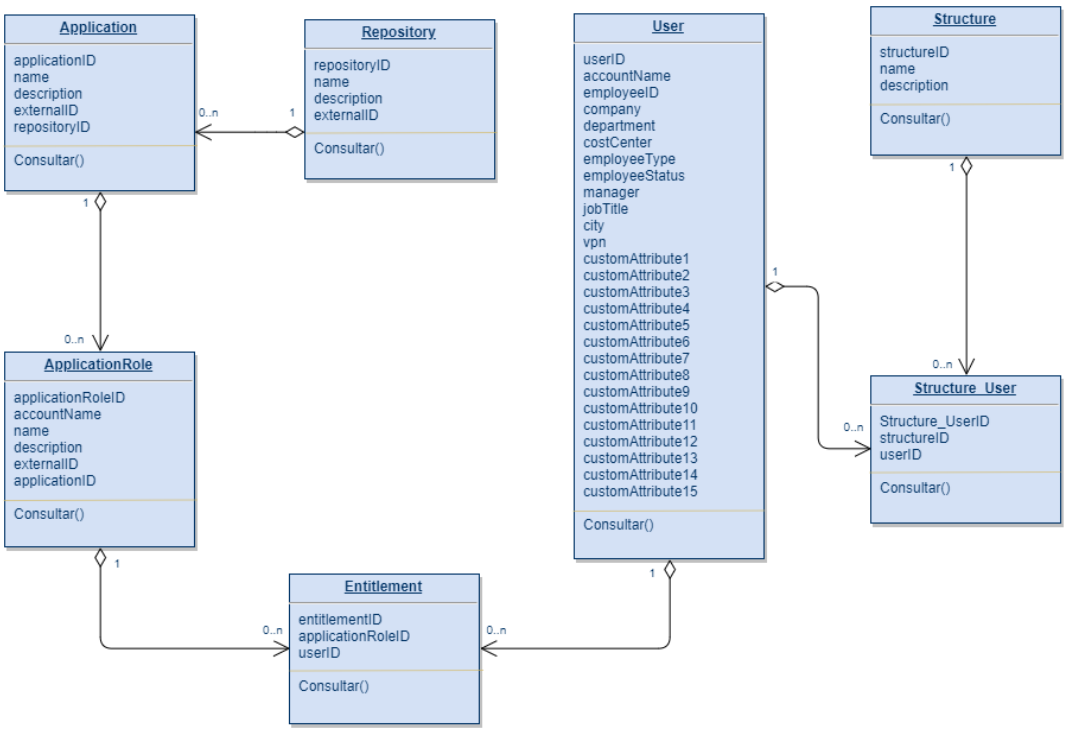


Figura 9 - Diagrama de classes de Role Mining

A figura 10 a seguir corresponde a diagrama de classes de Gestão de Identidades.

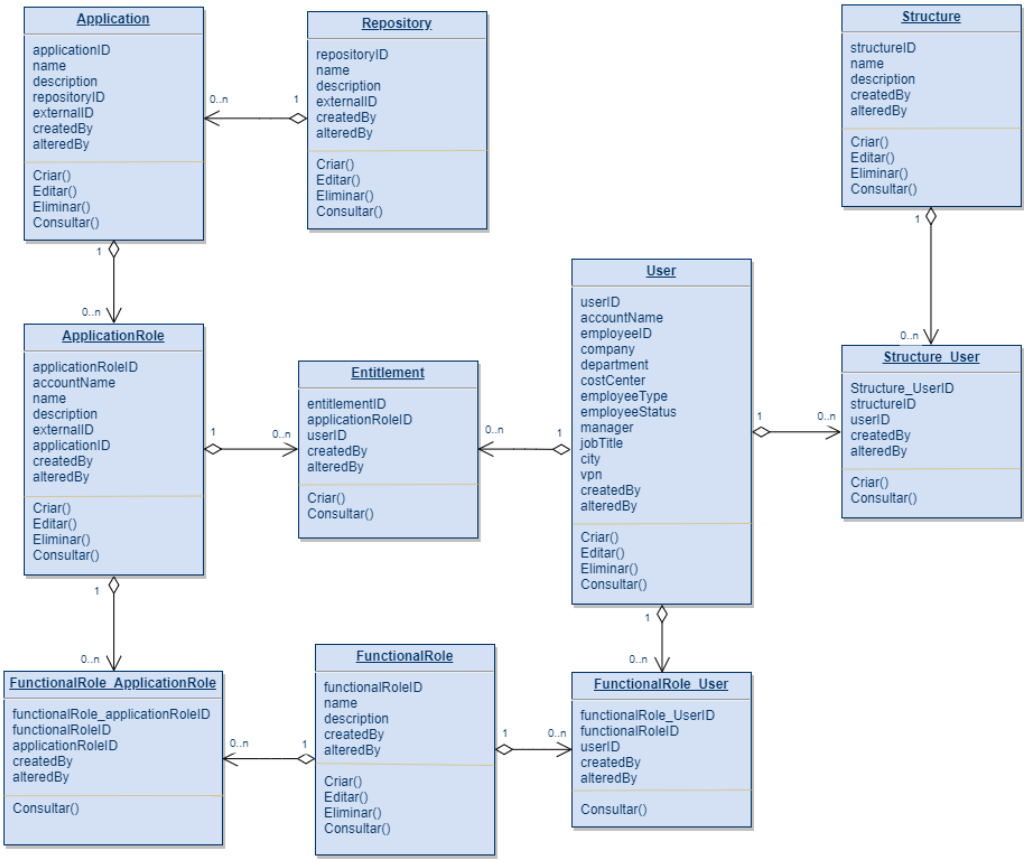


Figura 10 - Diagrama de classes de Gestão de Identidades

A tabela 10 ilustra a matriz CRUD relativamente ao diagrama de classes Gestão de Identidades.

Tabela 10 – Matriz CRUD

Classe/CRUD	**Criar**	**Editar**	**Consultar**	**Eliminar**
Application	X	X	X	X
Repository	X	X	X	X
Structure	X	X	X	X
Structure_User	X		X	
User	X	X	X	X
Entitlement	X		X	
Functional Role	X	X	X	X
Application Role	X	X	X	X
FunctionalRole_ ApplicationRole			X	
FunctionalRole_User			X	

4.5 Dicionário de dados

Neste capítulo pretende-se descrever cada atributo das classes de gestão de Identidades, consoante ao tipo de dados, tamanho, restrições e as operações que podem ser realizadas nos mesmos.

Na tabela 11 é apresentada a descrição dos campos da classe “Application” consoante ao tipo de dados, tamanho, restrições e a descrição dos mesmos.

Tabela 11 - Dicionário de dados da tabela "Application"

Application				
Nome	**Tipo de Dados**	**Tamanho**	**Descrição**	**Restrições**
applicationID	Integer	__	Chave primária	Não Nulos, Maior que zero
name	Varchar	100	Nome da aplicação	Não Nulo
description	Varchar	100	Descrição da aplicação	Não Nulo
externalID	Varchar	50	Chave primária para apresentação	Não Nulo
repositoryID	Integer	__	Chave estrangeira refente à tabela "Repositories"	Não Nulo
createdBy	Varchar	50	Detalhe de criação (data, hora, nome)	Não Nulo
alteredBy	Varchar	50	Detalhe da alteração (data, hora, nome)	Não Nulo

Na tabela 12 é apresentada a descrição dos campos da classe “Structure” consoante ao tipo de dados, tamanho, restrições e a descrição dos mesmos.

Tabela 12 - Dicionário de dados da tabela "Structure"

Structure				
Nome	**Tipo de Dados**	**Tamanho**	**Descrição**	**Restrições**
structureID	Integer	__	Chave primária	Não Nulos, Maior que zero
name	Varchar	100	Nome da estrutura	Não Nulo
description	Varchar	100	Descrição da estrutura	Não Nulo
createdBy	Varchar	50	Detalhe de criação (data, hora, nome)	Não Nulo
alteredBy	Varchar	50	Detalhe da alteração (data, hora, nome)	Não Nulo

Na tabela 13 é apresentada a descrição dos campos da classe “User” consoante ao tipo de dados, tamanho, restrições e a descrição dos mesmos

Tabela 13 - Dicionário de dados da tabela "User"

User				
Nome	**Tipo de Dados**	**Tamanho**	**Descrição**	**Restrições**
userID	Integer	__	Chave primária	Não Nulos, Maior que zero
accountName	Varchar	50	Nome do utilizador ou conta	Não Nulo
employeeID	Varchar	50	ID do colaborador	Não Nulo
company	Varchar	100	Nome da empresa	Não Nulo
department	Varchar	100	Departamento	Não Nulo
costCenter	Varchar	50	Centro de Custos	Não Nulo
employeeType	Varchar	100	Tipo de colaborador	Não Nulo
employeeStatus	Integer	1	Estado do colaborador	Não Nulo
manager	Varchar	50	Manager do colaborador	Não Nulo
jobTitle	Varchar	50	Título do cargo	Não Nulo
city	Varchar	50	Cidade	Não Nulo
vpn	Integer	1	Estado de acesso a vpn	Não Nulo
createdBy	Varchar	50	Detalhe de criação (data, hora, nome)	Não Nulo
alteredBy	Varchar	50	Detalhe da alteração (data, hora, nome)	Não Nulo

Na tabela 14 é apresentada a descrição dos campos da classe “Functional Role” consoante ao tipo de dados, tamanho, restrições e a descrição dos mesmos

Tabela 14 - Dicionário de dados da tabela "FunctionalRole"

Functional Role				
Nome	**Tipo de Dados**	**Tamanho**	**Descrição**	**Restrições**
functionalRoleID	Integer	__	Chave primária	Não Nulos, Maior que zero
name	Varchar	100	Nome da função	Não Nulo
description	Varchar	100	Descrição da função	Não Nulo
createdBy	Varchar	50	Detalhe de criação (data, hora, nome)	Não Nulo
alteredBy	Varchar	50	Detalhe da alteração (data, hora, nome)	Não Nulo

Na tabela 15 é apresentada a descrição das operações possíveis das tabelas Application, Structure, User e Functional Role ilustradas nas figuras 11, 12, 13 e 14.

Tabela 15 - Operações das tabelas

Operações das Tabelas	
Nome	**Descrição**
Criar ()	Conjunto de operações que permite criar campos das tabelas.
Editar ()	Conjunto de operações que permite editar os campos das tabelas.
Consultar ()	Conjunto de operações que permite obter ou visualizar os campos das tabelas.
Eliminar ()	Conjunto de atividades que permite eliminar campos da tabela.

5 Tecnologias

No que toca a desenvolvimento de plataformas *web*, atualmente são vastas as listas de tecnologias para a implementação das mesmas. Em seguida são apresentados grupos de tecnologias e ferramentas utilizadas no desenvolvimento do projeto em estudo.

5.1 HTML

HTML é uma linguagem de marcação padrão para criação de páginas *web*, representados por tags para renderizar os conteúdos da página, conteúdos esses que são interpretados pelo browser e exibidos aos utilizadores [15]. As tags *HTML* rotulam partes do conteúdo como table, heading, parágrafo, que os navegadores não exibem, mas as usam para renderizar o conteúdo da página. Esta tecnologia foi utilizada para criação dos conteúdos das vistas da plataforma.

5.2 CSS

CSS é uma linguagem de folhas de estilo, utilizado para definir estilos a páginas *web*, incluindo design, layout e tamanho de telas, por meio de linguagens de marcação como *HTML*, *XML e CSV* [16]. Para definir estilos a uma página *web* pode se fazer de várias formas, umas delas e a mais comum é criar o ficheiro à parte e fazer a sua referência no ficheiro da página. A mesma tecnologia foi utilizada para formatar os conteúdos das vistas da plataforma atribuindo estilos.

5.3 Angular

Angular é um *framework open-source* para desenvolvimento front-end, baseado em *TypeScript* e criado pelos desenvolvedores da google para facilitar o desenvolvimento de aplicações *web* e mobile. A *framework* é uma reescrita completa do AngularJS desenvolvida pela mesma equipa que a construiu [17]. A mesma disponibiliza vários recursos para facilitar criação de aplicações como vinculação de dados, diretrizes básicas de modelos, validação de formulários, roteamento, componentes reutilizáveis, injeção de dependências. Foi utilizada para o desenvolvimento de todo o front-end da plataforma.

5.4 Git

O Git é um sistema de controle de versões distribuído *open-source*, projetado para lidar com projetos de desenvolvimento de software, onde são usados para registrar o histórico de versões do projeto desde o seu inicio num repositório online [18]. Durante a desenvolvimento do projeto foi utilizado para o controle de versões o *TortoiseGit*, uma interface *open-source* do *Windows Shell* para o sistema de controle de versão Git [19]. O

uso desta tecnologia facilitou o desenvolvimento em ambientes de equipa na medida em que permitiu com que os membros da equipa trabalhassem em simultâneo. No caso foi privatizado o repositório, permitindo acessos só a colaboradores autenticados por questões de segurança.

5.5 SQL Server

SQL é um sistema de gestão de base de dados (SGBD) relacional desenvolvida pela Microsoft. A principal ferramenta de interface deste sistema é o *SQL Server Management Studio* [20].

5.6 Microsoft SQL Server Management Studio

Microsoft SQL Server Management Studio é um software para a configuração, gestão e administração de todos os componentes do Microsoft SQL Server [21]. Este software é utilizado no presente projeto para a criação e gestão da base de dados referente ao mesmo.

5.7 ASP.NET

ASP.NET é a plataforma para desenvolvimento de aplicações *web* da Microsoft, sucessor da tecnologia ASP, que permite que, através de uma linguagem de programação integrada na .NET *framework*, criar páginas dinâmicas. As aplicações para esta plataforma são programadas em linguagens como C#, F# e *Visual Basic* [22]. No desenvolvimento deste projeto foi escolhido ASP.NET Core por ser ideal para criação de web APIs e pela existência de template em Angular, onde o ASP.NET para desnvolvimneto de back-end (RESTful API) e o Angular para o desenvolvimento front-end (interface utilizador).

5.8 Microsoft Power BI

Power BI é um serviço de análise de negócios da Microsoft, com o objetivo de fornecer visualizações interativas e dinâmicas dos dados onde podem ser partilhas ou incorporadas nas aplicações ou sites. Com uma interface simples permite com que os utilizadores criem os seus próprios relatórios e *dashboards.* As vantagens do uso do *Power BI* relativamente a outras farramentas de visualização consiste na capacidade de carregar visualizações personalizadas através do *AppSource,* no acesso à informação em tempo real, no acesso da informação em qualquer aplicativo móvel, no uso empresarial, nas atualizações mensais com novas funções e a facilidades de uso [23]. Esta ferramenta é utilizada para analisar os mapeamentos dos utilizadores para criar de relatórios e *dashboards*.

6 Implementação

O presente projeto é composto por dois processos, nomeadamente o "*Role Mining*" e "Gestão de Identidades". Inicialmente foi realizada a criação da base de dados através da tecnologia "*Microsoft SQL Server Management Studio"* para as partes constituintes do projeto. Em seguida avançou-se para a criação do projeto no "*Microsoft Visual Studio"*, em que a sua implementação é explicada a seguir com apresentação das interfaces desenvolvidas.

Inicialmente para aceder à plataforma, o utilizador ("Administrador") realiza o seu login, onde após isso, é reencaminhado para interface inicial da plataforma "Figura 11" onde tem permissões para realizar atividades na mesma.

Figura 11 - Vista Home

A vista acima ilustrada consiste na interface inicial da plataforma, apresentada quando o utilizador (administrador) acede a mesma, após a sua devida autenticação, onde está ilustrado *dashboard* do Power BI.

6.1 Role Mining

Role Mining é o modulo onde é possível analisar e visualizar os mapeamentos de cada utilizador com os recursos que o mesmo tem acesso, como está ilustrado na figura 12. Foi

possível utilizar e analisar esses mesmos dados no *dashboard* do Power BI que esta na página “Home” da plataforma ilustrado na figura 11.

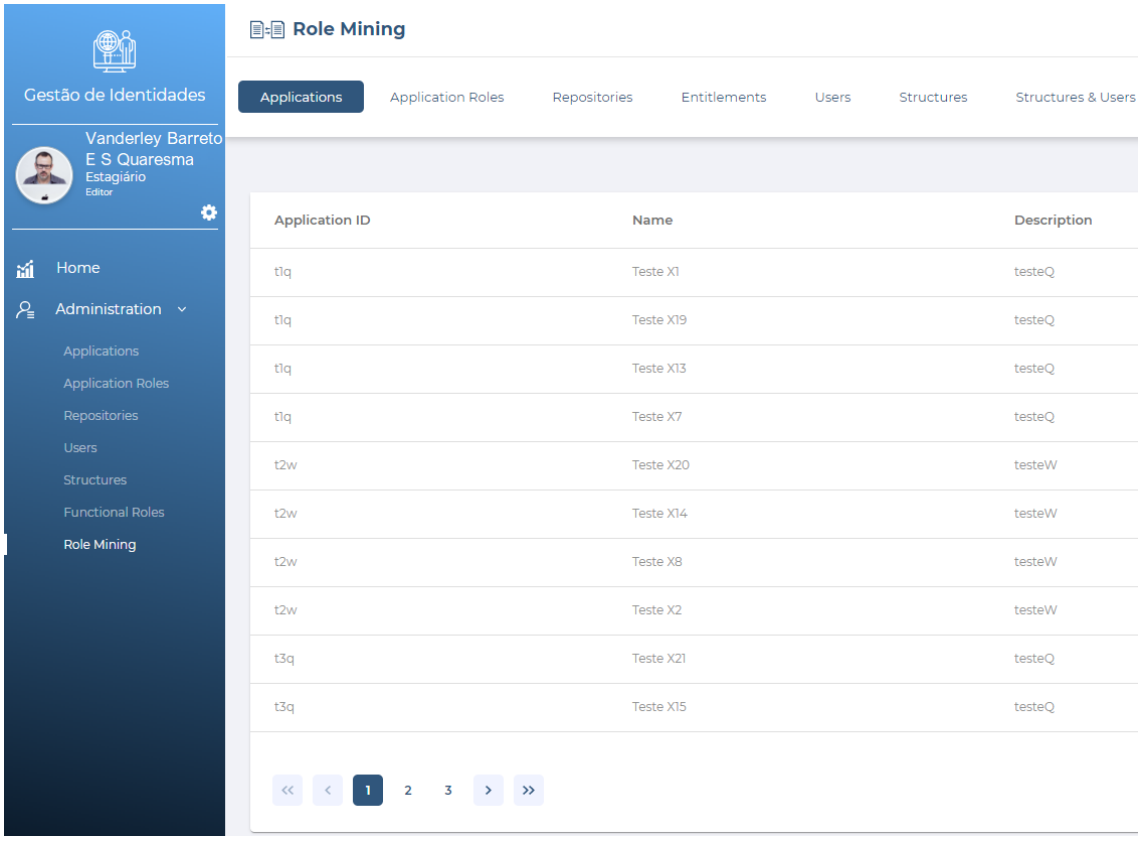


Figura 12 - Vista Role Mining

A ilustração consiste na vista do administrador que permite visualizar todas as aplicações e o repositório em que se encontra armazenada. O application ID ilustrado na vista das aplicações refere-se ao external ID no back-end e na base de dados.

Na ilustração é apresentado também aba de “Users”, “Structures” e “Structures & Users” correspondente às classes que foram desenvolvidas durante o estágio, onde:

Users - consiste na vista onde os administradores podem visualizar cada utilizador registado no sistema e os seus atributos.

Structures - consiste na vista do administrador que permite visualizar as estruturas organizacionais existentes, que posteriormente são atribuídas a cada utilizador.

Structures & Users - vista do administrador que permite visualizar os utilizadores e a estrutura a que cada um pertence.

Procedimento de implantação do *Role Mining*

Numa primeira fase no desenvolvimento do Back-End foi realizado a criação dos controladores e modelos das classes desenvolvidas (Users, Structures, Structures & Users) e a conexão com base de dados e criação da migração. Em seguida são ilustrados os esboços dos códigos da implementação de controladores e modelos.

```
1 reference
public class RM_UserController : Controller
{
    private readonly RoleMiningContext _context;
    private RM_UserRepository RM_UserRepository;
    private RM_EntitlementRepository RM_EntitlementRepository;
    private RM_Structure_UserRepository RM_Structure_UserRepository;

    /*Construtor*/
    0 references | 0 exceptions
    public RM_UserController(RoleMiningContext context)
    {
        RM_UserRepository = new RM_UserRepository(context);
        RM_EntitlementRepository = new RM_EntitlementRepository(context);
        RM_Structure_UserRepository = new RM_Structure_UserRepository(context);
        _context = context;
    }

    // GET: RM_User
    // Devolve lista de 'RM_Users'
    [HttpGet]
    [EnableCors("AllowAll")]
    0 references | 0 requests | 0 exceptions
    public IEnumerable<RM_User> GetRM_Users()
    {
        return RM_UserRepository.FindRM_Users();
    }

    [HttpGet("page")]
    [EnableCors("AllowAll")]
    0 references | 0 requests | 0 exceptions
    public IEnumerable<RM_User> GetRM_UsersBySize([FromQuery] string filter, [FromQuery]
    int pageIndex, [FromQuery] int pageSize,[FromQuery] string field, [FromQuery] string order)
    {
        return RM_UserRepository.FindRM_UsersBySize(filter, pageIndex, pageSize, field, order);
    }
```

Figura 13 - Role Mining User Controller

A figura 12 consiste no código do controlador da classe “User” onde são desenvolvidos os métodos HTTP (GET, POST, PUT, DELETE), em que o GET serve para recuperar ou pegar recursos, o POST serve para atualizar ou inserir um recurso, o PUT serve para inserir um recurso e o DELETE para excluir um recurso. Para o *Role Mining* só disponível o GET.

Na figura 14 ilustra o código do modelo correspondente a classe “User”.

```
public class RM_User
{
    [Key]
    5 references | 0 exceptions
    public string accountName { get; set; }
    5 references | 0 exceptions
    public string employeeID { get; set; }
    3 references | 0 exceptions
    public string company { get; set; }
    3 references | 0 exceptions
    public string department { get; set; }
    3 references | 0 exceptions
    public string costCenter { get; set; }
    3 references | 0 exceptions
    public string employeeType { get; set; }
    3 references | 0 exceptions
    public int employeeStatus { get; set; }
    3 references | 0 exceptions
    public string manager { get; set; }
    3 references | 0 exceptions
    public string jobTitle { get; set; }
    3 references | 0 exceptions
    public string city { get; set; }
    3 references | 0 exceptions
    public int vpn { get; set; }
```

Figura 14 - Modelo da classe "User"

Na etapa seguinte foi feito o desenvolvimento do Front-End, onde foi realizado a instalação da *framework* Angular através de linhas de comando e as bibliotecas necessárias para usabilidade do mesmo. Em seguida foi desenvolvido os serviços e componentes das classes. Na figura seguinte é ilustrada os serviços desenvolvidos da classe "User" no front-end, consoates a rotas já criadas no controlador no back-end.

```
getUsers(): Observable<User[]> {
    return this.http.get<User[]>(this.userUrl);
}

getUsersByPage(filter: string, pageIndex: number, pageSize: number, field: string,
 order: string): Observable<any> {
   const url =
   `${this.userUrl}/page?filter=${filter}&pageIndex=${pageIndex}&pageSize=${pageSize
   }&field=${field}&order=${order}`;
    return this.http.get<User[]>(url);
}

getUserById(userID: number): Observable<any> {
    const url = `${this.userUrl}/${userID}`;
    return this.http.get(url);
}
```

Figura 15 - Esboço dos serviços da classe "User"

Após o desenvolvimento dos componentes e serviços, seguiu-se para o desenvolvimento da vista na implementação do ficheiro HTML e CSS, e finalização da interface.

6.2 Gestão de identidades

Gestão de identidades é o modulo onde é feita a criação e gestão de perfis de identidades, os seus atributos e respetivas permissões de forma a encontrar padrões de acessos e funções de cada utilizador. Esses padrões permitem clarificar as funções que um utilizador tem quando registado no sistema.

As vistas apresentadas a seguir consiste nas interfaces das entidades que compoêm o sistema.

6.2.1 Applications

A seguir está ilustrada na figura 16 a interface de aplicações, relativamente a vista do administrador.

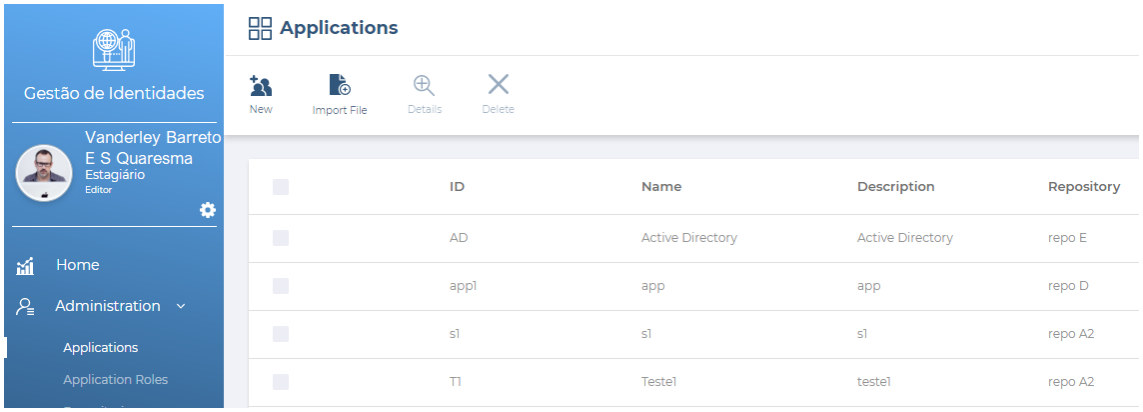


Figura 16 - Vista Applications

A vista acima apresentada, ilustra a vista do administrador, em que permite visualizar todas as aplicações e seu respetivo repositório. Nesta vista o administrador pode criar, importar, pesquisar e eliminar uma aplicação. É possível também editar uma aplicação recorrendo a detalhe da aplicação (figura 17) onde podem ser adicionadas as permissões e visualizados os utilizadores, que têm acesso a essa aplicação e sua respetiva função, como podemos observar na imagem seguinte.

Nota: O application ID da vista das aplicações ilustrado na figura 16 refere-se ao external ID na base de dados. No caso de uma atualização (Update) ou criação, altera-se o external ID e não o application ID (chave primária) de origem, de forma a chave primária não ser alterada e criada automaticamente.

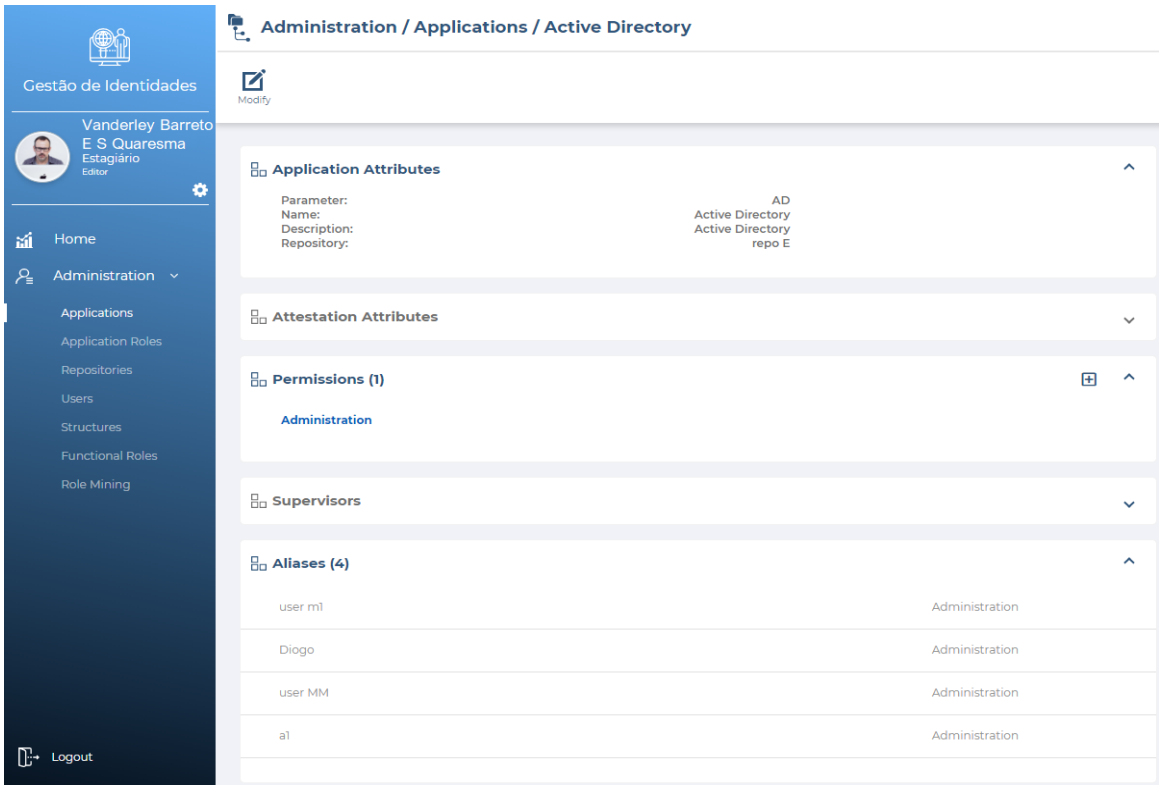


Figura 17 - Vista Detalhes Application

6.2.2 Repositories

A seguir está ilustrada na figura 18 a interface de repositórios, relativamente a vista do administrador.

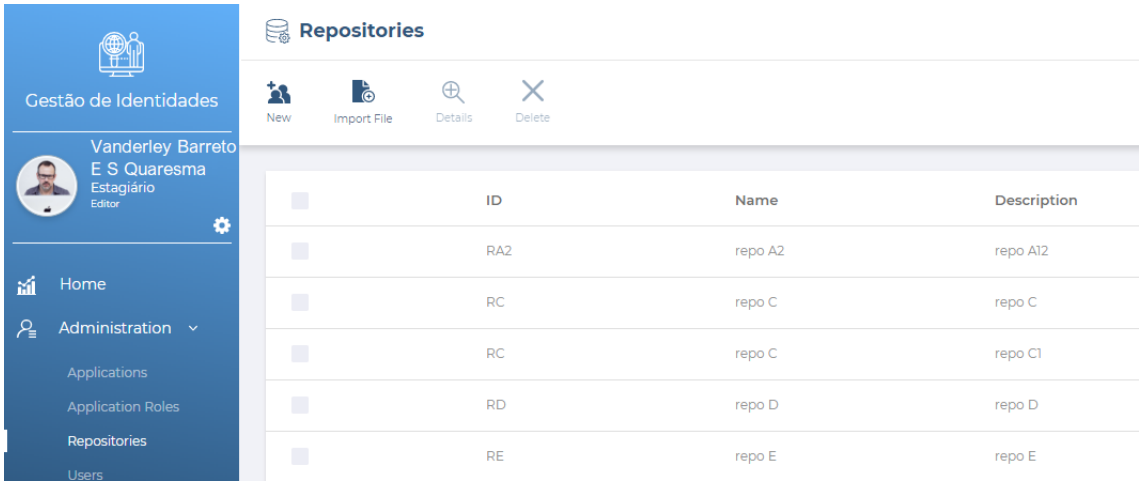


Figura 18 - Vista Repositories

Nesta vista do administrador podem ser visualizadas os repositórios existentes, onde estão armazenadas as aplicações. Para além de visualizar, o administrador pode criar, pesquisar e eliminar um repositório. Pode também editar um repositório recorrendo ao detalhe do

mesmo (figura 19), onde podem também adicionar as permissões. O ID do repositório ilustrado na figura 18 da vista dos repositórios refere-se ao external ID na base de dados. No caso de uma atualização ou criação, se atualiza ou se altera o external ID e não o ID (chave primária) de origem, de forma a chave primária não ser alterada e ser criada automaticamente.

Figura 19 - Vista Detalhes Repositories

6.2.3 Users

Em seguida está ilustrada na figura 20 a vista do administrador da interface dos utilizadores.

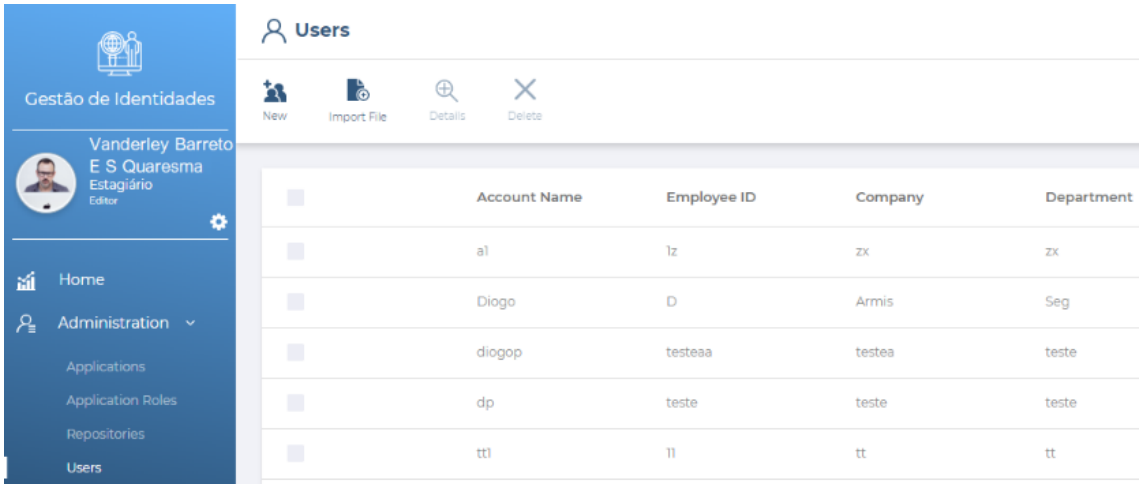


Figura 20 - Vista Users

A vista apresentada consiste na vista do administrador onde podem ser visualizadas os utilizadores e os seus dados. Nesta vista o utilizador pode criar, pesquisar, eliminar e também editar recorrendo ao detalhe do utilizador (figura 21), onde também pode adicionar "permissões" e "estrutura organizacional" aos utilizadores.

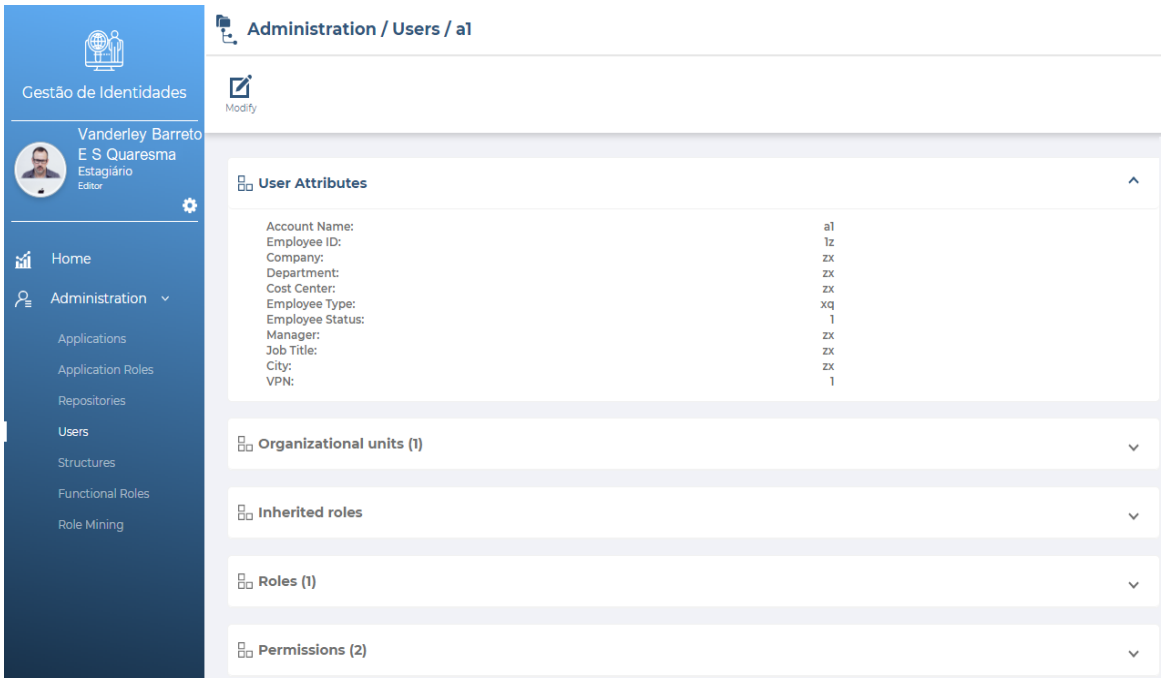


Figura 21 - Vista Detalhes Users

6.2.4 Structures

A seguir é ilustrada na figura 22 a vista do administrador da interface das estruturas.

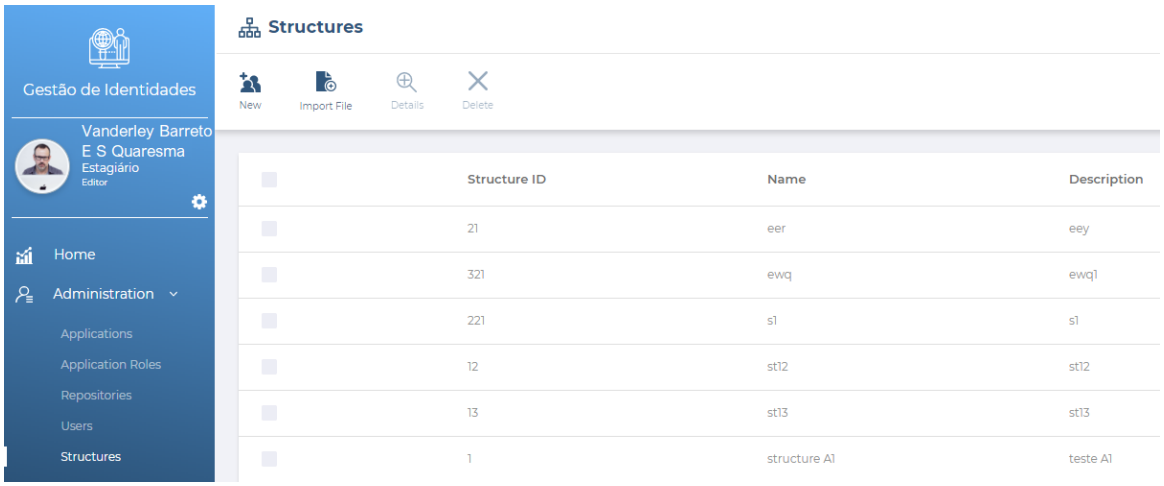


Figura 22 - Vista Structures

Esta vista consiste na vista do administrador onde se pode visualizar as estruturas organizacionais da organização e onde se pode criar, pesquisar, eliminar e editar. Para editar uma estrutura recorre-se ao detalhe do mesmo (figura 23), onde é possível também adicionar utilizadores e funções à estrutura. A figura 23 ilustra essa situação.

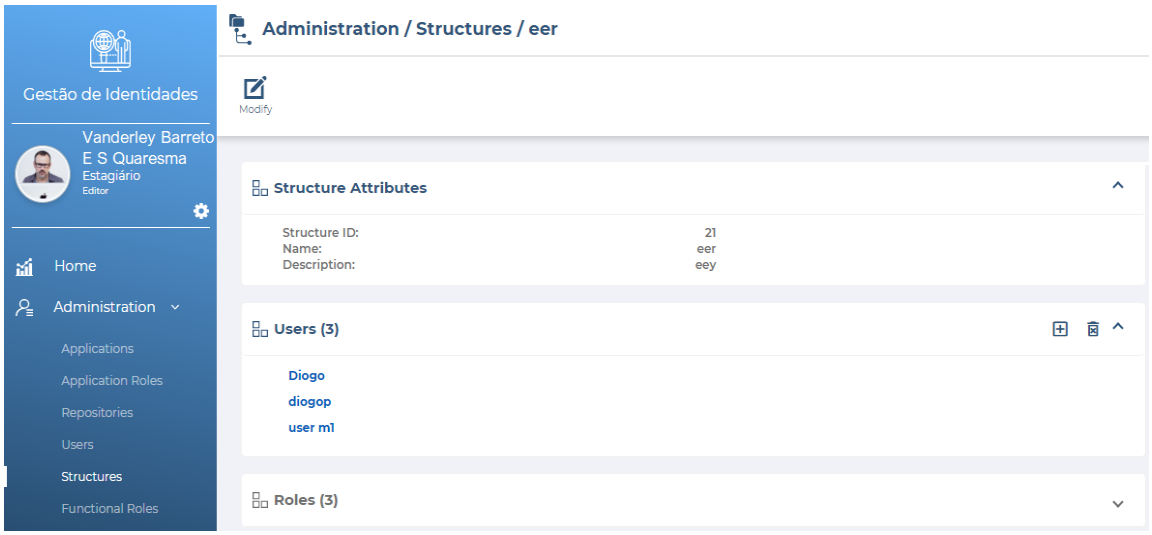


Figura 23 - Vista Detalhes Structures

6.2.5 Fuctional Roles

A figura seguinte (figura 24) ilustra a vista do administrador da interface das funções.

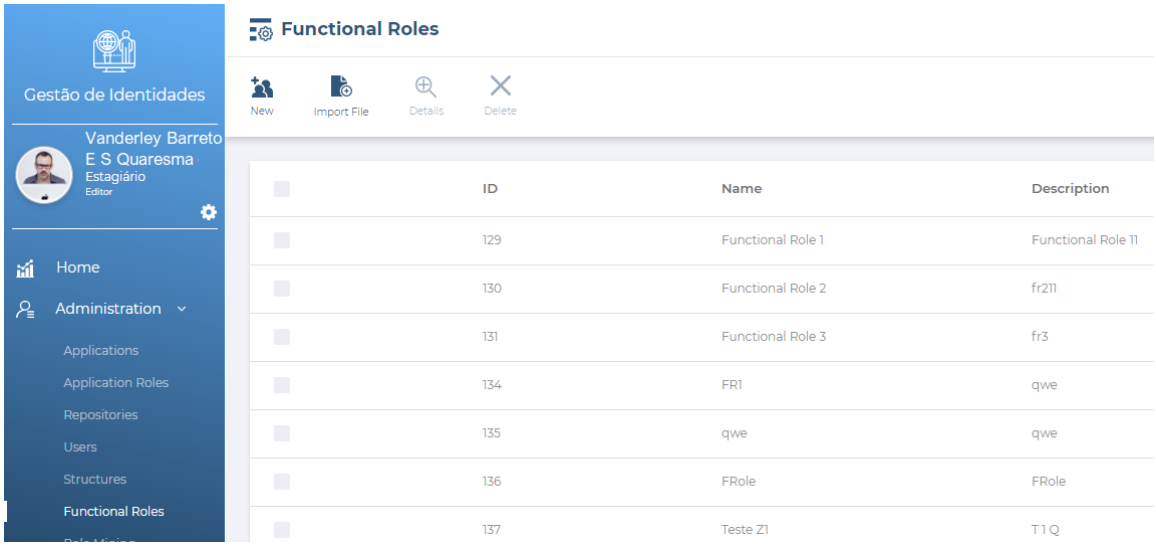


Figura 24 - Vista Functional Roles

A figura 24 apresentada, ilustra a vista do administrador onde se pode visualizar Funções que posteriormente são atribuídas aos utilizadores (na vista dos utilizadores) e à estrutura (na vista das estruturas). Nesta vista como realizadas nas outras, o administrador pode criar, pesquisar, eliminar e editar uma função. Para editar o administrador recorre ao detalhe da função (figura 25), onde pode-se também adicionar permissões a função o que os utilizadores que exercer essa função usufruir da mesma permissão, adicionar os utilizadores que vão exercer a função e adicionar estruturas que pertencem à mesma.

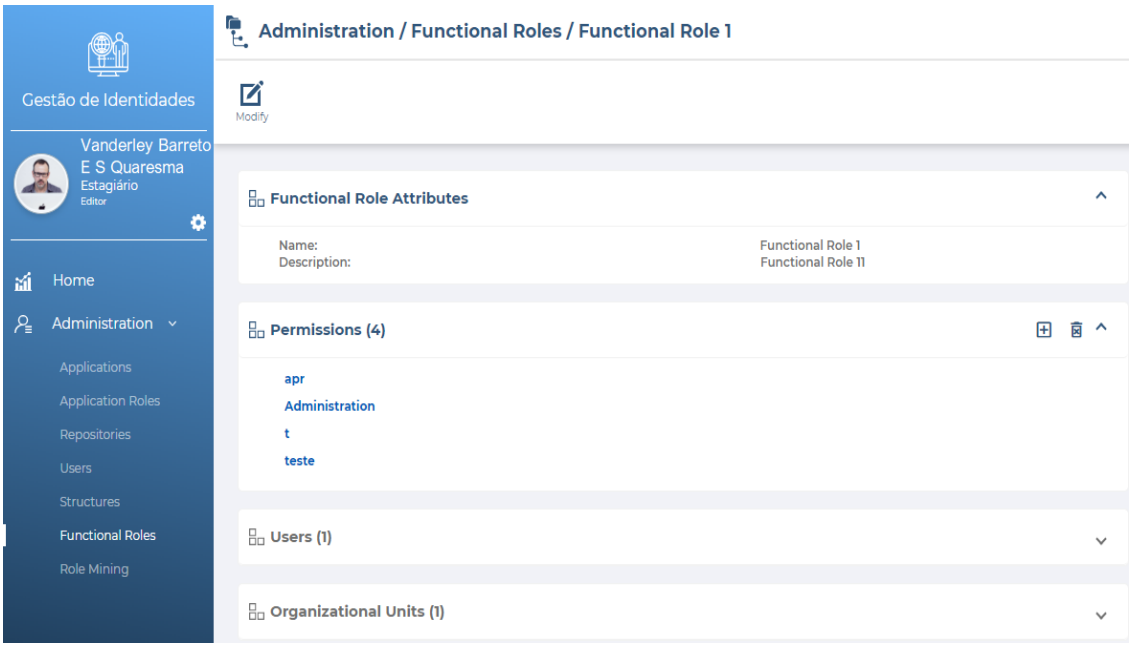


Figura 25 - Vista Detalhes Functional Roles

6.2.6 Procedimento da implementação de Gestão de Identidades

Na implementação da parte de Gestão de identidades, inicialmente foi feito o desenvolvimento do Back-End, onde foram implementadas os controladores e modelos das classes desenvolvidas durante o estágio. Após está implementação foram criadas as migrações e conexão com a base de dados correspondentes. A figura seguinte ilustra o código do modelo da classe “Strcture”.

```
29 references
public class Structure
{
    10 references | 0 exceptions
    public int structureID { get; set; }
    10 references | 0 exceptions
    public string name { get; set; }
    7 references | 0 exceptions
    public string description { get; set; }
    1 reference | 0 exceptions
    public string createdBy { get; set; }
    1 reference | 0 exceptions
    public string alteredBy { get; set; }
}
```

Figura 26 - Esboço do Modelo da Classe "Structure"

A figura 27 ilustra o esboço do controlador da classe Structure com os métodos HTTP para controlo dos serviços request e response.

```
1 reference
public class StructureController : Controller
{
    private readonly RoleMiningContext _context;
    private ApplicationRepository ApplicationRepository;
    private ApplicationRoleRepository ApplicationRoleRepository;
    private RepositoryRepository RepositoryRepository;
    private EntitlementRepository EntitlementRepository;
    private StructureRepository StructureRepository;
    private Structure_UserRepository Structure_UserRepository;
    private UserRepository UserRepository;
    private FunctionalRoleRepository FunctionalRoleRepository;
    private FunctionalRole_UserRepository FunctionalRole_UserRepository;

    /*Construtor*/
    0 references | 0 exceptions
    public StructureController(RoleMiningContext context)...

    // GET: Structure
    // Devolve lista de 'Structures'
    [HttpGet]
    [EnableCors("AllowAll")]
    0 references | 0 requests | 0 exceptions
    public IEnumerable<Structure> GetStructures()
    {
        return StructureRepository.FindStructures();
    }

    [HttpGet("page")]
    [EnableCors("AllowAll")]
    0 references | 0 requests | 0 exceptions
    public IEnumerable<Structure> GetStructuresBySize([FromQuery] string filter, [FromQuery]
     int pageIndex, [FromQuery] int pageSize, [FromQuery] string field, [FromQuery] string order)
    {
        return StructureRepository.FindStructuresBySize(filter, pageIndex, pageSize, field, order);
    }

    // GET: Structure/5
    // Devolve cada 'Structure' individualmente, consoante o id
    [HttpGet("{id}")]
    [EnableCors("AllowAll")]
    0 references | 0 requests | 0 exceptions
    public IActionResult GetStructureById([FromRoute] int id)
    {
        if (!ModelState.IsValid)
        {
            return BadRequest(ModelState);
        }

        var structure = StructureRepository.FindStructureById(id);

        if (structure == null)
        {
            return NotFound("Structure doesn't exist.");
        }

        var structureDto = new StructureDTO(structure);

        return Ok(structureDto);
    }
```

Figura 27 - Esboço do controlador da classe "Structure"

Em seguida avançou-se para o desenvolvimento do Front-End, onde foram criados os componentes e os serviços das classes. Também foram criados os componentes do "Create", "Edit", e do "Delete". Em seguida foram implementadas as vistas das classes

incluindo para criar, editar, eliminar e de detalhes através de desenvolvimento de ficheiros HTML e CSS. Nas figuras 28 e 29 estão ilustrados exemplos dessas implementações.

```
<div style="display:flex; flex-direction:column; justify-content:space-between;
height:100%; align-items:center;" *ngIf="!isLoading">
  <div style="text-align:center">
    <p> Confirms Delete? </p>
  </div>
  <div>
    <button type="button" (click)="close()">Cancel</button>
    <button type="button" (click)="deleteStructures()" class="delete-button">
      Delete
    </button>
  </div>
</div>

<div *ngIf="isLoading" style="display: flex; justify-content: center; align-
items: center; height:100%;">
  <mat-progress-spinner  mode="indeterminate" [diameter]="50">
  </mat-progress-spinner>
</div>
```

Figura 28 - Esboço do ficheiro HTML Eliminar Structure

A figura 29 ilustra o ficheiro HTML da interface para eliminação de uma estrutura.

```
<div *ngIf="!isLoading" style="height: 100%">
  <div style="height: 100%">
    <form style="width:100%; height: 100%" [formGroup]="regForm">
      <div class="form-title">
        Structure
      </div>
      <div class="form-field" [ngClass]="{'error': nameError}">
        <hr class="div-up"/>
        <label>Name</label>
        <input id="name" type="text" formControlName="name" matInput
        placeholder="Insert" (focus)="nameError = false; erro = false">
      </div>
      <div class="form-field" [ngClass]="{'error': descError}">
        <label>Description</label>
        <input id="description" type="text" formControlName="description" matInput
        placeholder="Insert" (focus)="descError = false; erro = false">
        <div style="height:20px">
          <span class="errorMessage" *ngIf='erro==true'>{{mensagem}}</span>
        </div>
      </div>
      <div class="footer">
        <hr class="div-down"/>
        <div class="buttons-container">
          <button type="button" (click)="close()"> Cancel </button>
          <button type="button" (click)="putStructure()"> Modify </button>
        </div>
      </div>
    </form>
  </div>
</div>

<div class="spinner" *ngIf="isLoading">
  <mat-progress-spinner color="primary" mode="indeterminate">
  </mat-progress-spinner>
</div>
```

Figura 29 - Esboço do ficheiro HTML Editar Structure

A figura 30 ilustra o ficheiro HTML da interface para criação de uma estrutura.

```
<div *ngIf="!isLoading" style="height: 100%">
  <div style="height: 100%">
    <form style="width:100%; height:100%" [formGroup]="regForm">
      <div class="form-title">
        Structure
      </div>
      <div class="form-field" [ngClass]="{'error': idError}">
        <hr class="div-up" />
        <label>Structure ID</label>
        <input id="structureID" type="text" formControlName="structureID"
        matInput                placeholder="Insert" (click)="idError = false;
        erro = false">
      </div>
      <div class="form-field" [ngClass]="{'error': nameError}">
        <label>Name</label>
        <input id="name" type="text" formControlName="name" matInput
        placeholder="Insert" (click)="nameError = false; erro = false">
      </div>
      <div class="form-field" [ngClass]="{'error': descError}">
        <label>Description</label>
        <input id="description" type="text" formControlName="description"
        matInput placeholder="Insert" (click)="descError = false; erro = false">
      </div>
      <div style="height:20px">
        <span class="errorMessage" *ngIf='erro==true'>{{mensagem}}</span>
      </div>
      <div class="footer">
        <hr class="div-down" />
        <div class="buttons-container">
          <button type="button" (click)="close()"> Cancel </button>
          <button type="button" (click)="postStructure()"> Create </button>
        </div>
      </div>
    </form>
  </div>
</div>

<div class="spinner" *ngIf="isLoading">
  <mat-progress-spinner color="primary" mode="indeterminate">
  </mat-progress-spinner>
</div>
```

Figura 30 - Esboço do ficheiro HTML criar Structure

7 Verificação e validação

Durante o desenvolvimento deste sistema, foram realizados diversos tipos de teste e validações, nomeadamente testes funcionais, crucias para um melhor funcionamento da plataforma e correção de possíveis falhas. Ao nível de validações foram realizados testes que permitem que a criação, por exemplo de aplicações, seja concluída com dados devidamente inseridos. A figura seguinte ilustra esta situação.

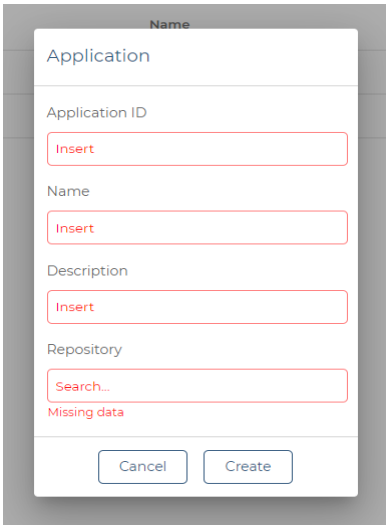


Figura 31 - Validação ao Criar Aplicação

Na figura seguinte é ilustrada o código de validação ao criar uma aplicação, para verificar se os campos foram preenchidos, ou seja, para impedir que sejam criados campos nulos.

```
. . .
else {
      this.erro = true;

      console.log('teste');

      if (this.regForm.value.externalID == null ||
this.regForm.value.externalID.trim() == "") {
        this.idError = true;
      }

      if (this.regForm.value.name == null || this.regForm.value.name.trim() ==
"") {
        this.nameError = true;
      }

      if (this.regForm.value.description == null ||
this.regForm.value.description.trim() == "") {
        this.descError = true;
      }

      if (this.regForm.value.repositoryID == null) {
        this.repositoryError = true;
      }

      this.mensagem = "Missing data";
    }
```

Figura 32 - Código de validação ao criar Aplicações

Como podemos observar o código, consiste numa condição para verificar se os campos “externalID”,“name”, “description” são preenchidos no “reForm”, de forma a não serem postados campos vazios.

Foram realizadas validações também para verificar se os dados inseridos já existem, o que podemos observar na ilustração seguinte.

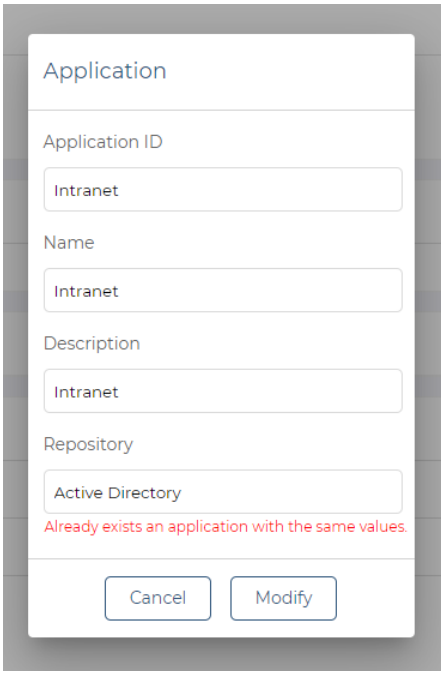


Figura 33 - Validação de campos existentes

A figura seguinte ilustra o código de validação dos dados para verificar se já existem.

```
foreach (Application a in applications) {
    if (a.applicationID != id)
    {
        return Ok(Json(new { message = "Already exists an application with the same values.", erro = true }));
    }
}
```

Figura 34 - Código de validação dos campos existentes

Como podemos observar o código consiste em uma condição “if” para verificar se os valores a introduzir já existem no repositório da aplicação, caso existam é devolvido um alerta com a mensagem “Already exists na application with these values”.

Ainda foram realizadas verificações ao importar os ficheiros CSV, de forma a verificar se os campos são compatíveis com os campos existentes nos ficheiros.

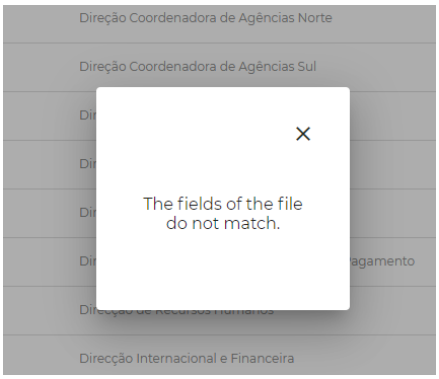


Figura 35 - Validação ao importar ficheiros

A figura seguinte consiste na ilustração do código onde é feita a validação dos campos a importar, para verificar se são compatíveis.

```
if (results.meta.fields[0] == "name" &&
            results.meta.fields[1] == "description" &&
            results.meta.fields[2] == "externalID" &&
            results.meta.fields[3] == "repositoryID"
          ) {
            for (let i = 0; i < results.data.length; i++) {
               if (this.totalApplications.find(rep => (rep.externalID ==
                results.data[i].externalID && rep.name == results.data[i].name
                && rep.description == results.data[i].description))

                || insertedApplications.find(rep => (rep.externalID ==
                results.data[i].externalID && rep.name == results.data[i].name
                && rep.description == results.data[i].description))

                || !this.repositories.find(r => r.repositoryID ==
                results.data[i].repositoryID )) {
                    repetidos++;

              } else {
                nRepetidos++;
                insertedApplications.push(results.data[i]);
              }
            }

...

} else {
            this.openDialogWarning(repetidos, nRepetidos, true, 'The fields of
            the file do not match.');
            evt.target.value = '';
            this.getApplications();
            this.getApplicationsByPage(this.filter, this.currentPage,
this.pageSize, this.sortActive, this.sortDirection);
          }
```

Figura 36 - Código de validação ao importar ficheiros

A ilustração consiste na apresentação do código, onde é feita condição “if”, para verificar

se os ficheiros a importar são compatíveis com os do sistema, caso não seja devolve um alerta com mensagem “The fields of the file do not match”.

Foram realizadas verificações também ao eliminar, para verificar se os dados a eliminar tem dependências.

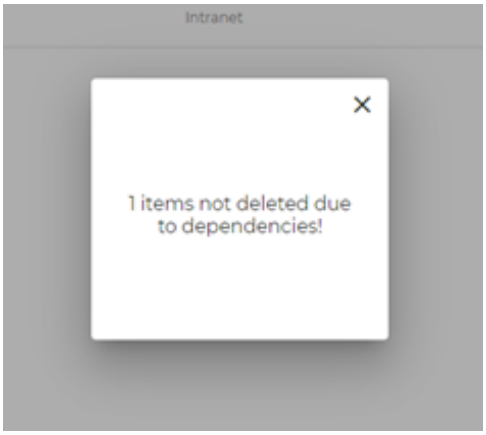


Figura 37 - Validação dos campos com dependências

A figura seguinte consiste na ilustração do código que permite a validação dos dados a eliminar para observar se há dependências.

```
if (!this.applicationRoles.find(ar => ar.applicationID ==
this.data.applications[i].applicationID)) {

this.applicationService.deleteApplication(this.data.applications[i].applicationID
).pipe(takeUntil(this.unsubscribe)).subscribe(result => {
            apagados++;
            if ((apagados + nApagados) == this.data.applications.length) {
              this.close();
              if (nApagados > 0) {
                this.openDialogWarning(0, 0, true, nApagados + ' items not
                deleted due to dependencies!');
              } else {
                this.openDialogWarning(0, 0, true,'Items deleted with success!');
              }
            }
          });
      }
```

Figura 38 - Código validação dos campos com dependências

O código consiste na implementação de uma condição “if / else” para verificar se dados a serem apagados têm dependências a outras classes, caso tenham é devolvido um alerta com a mensagem “items not delete due to dependencies!”, senão é devolvido a mensagem “Items delected with sucess”. **Nota:** Os testes acima indicados foram realizados a todas a entidades existentes no sistema.

8 Conclusões

O projeto foi desenvolvido em contexto de estágio na Armis, onde o seu desenvolvimento foi uma mais valia a nível de aprendizagem no que toca a desenvolvimento de hard e soft skills e uma experiência desafiadora. O objetivo foi desenvolver uma plataforma web para criação e gestão de perfis de negócio, em que eram criados e geridos identidades e acessos, através de atribuição de permissões e funções pra cada utilizador, onde a sua implementação foi concluído com sucesso.

O processo esperado consistiu na implementação dos ficheiros de "*Role Mining*", que contêm os mapeamentos dos utilizadores consoante os grupos de segurança que cada um tem acesso. Posteriormente foi possível utilizar e analisar esses mesmos dados no *dashboard* do *Power BI* que esta na página "*Home*".

Foi possível iniciar o processo de criação e gestão dos perfis de negócio, onde foi de encontro aos objetivos mencionados ao longo do projeto. Refere-se ainda que a implementação do projeto vai ter continuidade após a conclusão do estágio. Futuramente prevê-se melhorias ao nível da gestão de acessos.

Concluindo pode-se dizer que o desenvolvimento dessa plataforma trará valor acrescentado à organização, como redução de custos de suporte à infraestrutura, aumento da eficiência e produtividade, melhoria na qualidade e precisão e melhorias na segurança e privacidade.

E relativamente ao estágio posso concluir que foi de grande valia tanto no adquirir do conhecimento como na experiência profissional adquirida no decorrer do mesmo.

Bibliografia

[1] Armis, «Porquê ARMIS | Armis Group», 2018. Disponível em: http://www.armis.pt/porque/. [Acedido: 19-Set-2019].

[2] Oracle, «Oracle Identity Analytics Identity Warehouse - Oracle Identity Analytics Business Administrator's Guide», 2010. Disponível em: https://docs.oracle.com/cd/E27119_01/doc.11113/e23124/businessadministratorsguideprintable2.html#scrolltoc. [Acedido: 18-Set-2019].

[3] HardSecure, «HardSecure - Soluções - Gestão de Identidades e Acessos», 2008. Disponível em: https://www.hardsecure.com/solution/140/gestao-de-identidades-e-acessos. [Acedido: 19-Set-2019].

[4] Gartner, «About Gartner», *Gartner*, 2019. Disponível em: https://www.gartner.com/en/about. [Acedido: 20-Set-2019].

[5] billmath, «Microsoft Identity Manager». Disponível em: https://docs.microsoft.com/pt-pt/microsoft-identity-manager/microsoft-identity-manager-2016. [Acedido: 22-Set-2019].

[6] billmath, «Guia de conceitos do Microsoft BHOLD Suite». Disponível em: https://docs.microsoft.com/pt-pt/microsoft-identity-manager/bhold/bhold-concepts-guide. [Acedido: 23-Set-2019].

[7] M. billmath, «MIM Deprecated Features And Planning For The Future», 2018. Disponível em: https://docs.microsoft.com/en-us/microsoft-identity-manager/microsoft-identity-manager-2016-deprecated-features. [Acedido: 23-Set-2019].

[8] IBM, «IBM Security Access Manager - Overview - Portugal», 23-Set-2019. Disponível em: https://www.ibm.com/pt-en/marketplace/access-management. [Acedido: 24-Set-2019].

[9] IBM Corporation, «IBM Security Access Manager», p. 6, 2018. Disponível em: https://www.ibm.com/downloads/cas/EGDOEAWM. [Acedido: 24-Set-2019].

[10] Okta, «Lifecycle Management», *Okta*, 26-Jul-2016. Disponível em: https://www.okta.com/products/lifecycle-management/. [Acedido: 31-Out-2019].

[11] «6 Reasons Microsoft Customers Choose Okta for Identity», *Okta*, 24-Abr-2018. Disponível em: https://www.okta.com/resources/whitepaper/six-reasons-microsoft-customers-choose-okta-for-identity/. [Acedido: 15-Nov-2019].

[12] Metodologia, «Metodologia (engenharia de software)», *Wikipédia, a enciclopédia livre*.15-Fev-2017. Disponível em: https://pt.wikipedia.org/w/index.php?title=Metodologia_(engenharia_de_software)&oldid=48020659. [Acedido: 30-Out-2019].

[13] Jeff Sutherland,Ken Schwaber, «What is Scrum?», 2017. Disponível em: https://www.scrum.org/resources/what-is-scrum?gclid=EAIaIQobChMIgLm9oZn-5AIVwsjeCh3rBgQMEAAYAiAAEgKuZ_D_BwE. [Acedido: 04-Out-2019].

[14] Scrum Glossary, «Scrum Glossary | Scrum.org», 2017. Disponível em: https://www.scrum.org/resources/scrum-glossary. [Acedido: 04-Out-2019].

[15] HTML, «Introduction to HTML», 1991. Disponível em: https://www.w3schools.com/html/html_intro.asp. [Acedido: 04-Out-2019].

[16] W3C, «CSS Introduction», 1996. Disponível em: https://www.w3schools.com/css/css_intro.asp. [Acedido: 07-Out-2019].

[17] Angular, «Angular (framework)», *Wikipédia, a enciclopédia livre*. 03-Ago-2019. Disponível em: https://pt.wikipedia.org/w/index.php?title=Angular_(framework)&oldid=55897547. [Acedido: 09-Out-2019].

[18] Scott Chacon, «Git», 2019. Disponível em: https://git-scm.com/. [Acedido: 09-Out-2019].

[19] «About – TortoiseGit – Windows Shell Interface to Git». Disponível em: https://tortoisegit.org/about/. [Acedido: 14-Nov-2019].

[20] Techopedia, «What is SQL Server? - Definition from Techopedia», *Techopedia.com*, 2019. Disponível em: https://www.techopedia.com/definition/1243/sql-server. [Acedido: 09-Out-2019].

[21] Wikipedia, «SQL Server Management Studio», *Wikipedia*. 2019. Disponível em: https://en.wikipedia.org/w/index.php?title=SQL_Server_Management_Studio&oldid=918007599. [Acedido: 09-Out-2019].

[22] «ASP.NET», *Wikipédia, a enciclopédia livre*. 30-Mai-2019. Disponível em: https://pt.wikipedia.org/w/index.php?title=ASP.NET&oldid=55338039. [Acedido: 09-Out-2019]

[23] «O que é o Power BI | Microsoft Power BI». Disponível em: https://powerbi.microsoft.com/pt-pt/what-is-power-bi/. [Acedido: 14-Nov-2019].

Anexos

A 1. Código

A 1.1. Código Structure Controller

O codigo a seguir consiste no codigo desenvolvido no controlador da Estrutura, onde são desenvolvidos os métodos GET, POST, PUT, DELETE

```
[Route("Structure")]
 [EnableCors("AllowAll")]
 [ApiController]
 public class StructureController : Controller
 {
     private readonly RoleMiningContext _context;
     private ApplicationRepository ApplicationRepository;
     private ApplicationRoleRepository ApplicationRoleRepository;
     private RepositoryRepository RepositoryRepository;
     private EntitlementRepository EntitlementRepository;
     private StructureRepository StructureRepository;
     private Structure_UserRepository Structure_UserRepository;
     private UserRepository UserRepository;
     private FunctionalRoleRepository FunctionalRoleRepository;
     private FunctionalRole_UserRepository FunctionalRole_UserRepository;

     /*Construtor*/
     public StructureController(RoleMiningContext context)
     {
         ApplicationRepository = new ApplicationRepository(context);
         ApplicationRoleRepository = new ApplicationRoleRepository(context);
         RepositoryRepository = new RepositoryRepository(context);
         EntitlementRepository = new EntitlementRepository(context);
         StructureRepository = new StructureRepository(context);
         Structure_UserRepository = new Structure_UserRepository(context);
         UserRepository = new UserRepository(context);
         FunctionalRoleRepository = new FunctionalRoleRepository(context);
         FunctionalRole_UserRepository = new     FunctionalRole_UserRepository(context);
         _context = context;
     }
```

```
// GET: Structure
// Devolve lista de 'Structures'
[HttpGet]
[EnableCors("AllowAll")]
public IEnumerable<Structure> GetStructures()
{
    return StructureRepository.FindStructures();
}

[HttpGet("page")]
[EnableCors("AllowAll")]
public IEnumerable<Structure> GetStructuresBySize([FromQuery] string     filter, [FromQuery] int pageIndex, [FromQuery] int
pageSize, [FromQuery] string field, [FromQuery] string order)
{
    return StructureRepository.FindStructuresBySize(filter, pageIndex,     pageSize, field, order);
}

// GET: Structure/5
// Devolve cada 'Structure' individualmente, consoante o id
[HttpGet("{id}")]
[EnableCors("AllowAll")]
public IActionResult GetStructureById([FromRoute] int id)
{
    if (!ModelState.IsValid)
    {
        return BadRequest(ModelState);
    }

    var structure = StructureRepository.FindStructureById(id);

    if (structure == null)
    {
        return NotFound("Structure doesn't exist.");
    }

    var structureDto = new StructureDTO(structure);
```

```
            return Ok(structureDto);
        }

. . .

        // GET: Structure/User/userID
        // Devolve as estruturas em que o respetivo user está inserido
        [HttpGet("User/{userID}")]
        [EnableCors("AllowAll")]
        public IActionResult GetStructuresByUser([FromRoute] int userID)
        {
            if (!ModelState.IsValid)
            {
                return BadRequest(ModelState);
            }

            User user = UserRepository.FindUserByID(userID);

            if (user == null)
            {
                return NotFound("User doesn't exist.");
            }

            var structures_Users = Structure_UserRepository
            .FindStructure_UserByUser(userID);

            if (structures_Users == null)
            {
                return NotFound("Structures_Users error.");
            }

            List<Structure> structures = new List<Structure>();
            var exist = false;

            foreach (var s_u in structures_Users)
            {
                for (int i = 0; i < structures.Count(); i++)
```

```
                {
                    if (structures[i].structureID == s_u.structureID)
                    {
                        exist = true;
                        break;
                    }
                }
                if (!exist)
                {
                    structures.Add(StructureRepository.FindStructureById(s_u.structureID));
                }
                exist = false;
            }

            if (structures.Count() == 0)
            {
                return NoContent();
            }

            return Ok(structures);
        }

. . .

        // DELETE: Structure/id
        [HttpDelete("{id}")]
        [EnableCors("AllowAll")]
        public async Task<IActionResult> DeleteStructure([FromRoute] int id)
        {
            if (!ModelState.IsValid)
            {
                return BadRequest(ModelState);
            }

            var structure = StructureRepository.FindStructureById(id);
            if (structure == null)
            {
```

```
                return NotFound("Structure doesn't exist");
            }

            await StructureRepository.DeleteStructure(structure);
            var structureDto = new StructureDTO(structure);
            return Ok(structureDto);
        }

    }
```

A 1.2. Código Structure Component

O código ilustrado consiste no ficheiro em typeScript da biblioteca do Angular criado através de linha de comandos, com funções e métodos de densenvolvido de forma a facilitar a implementação do ficheiro HTML.

```
export class StructureComponent implements OnInit, OnDestroy {

  private unsubscribe: Subject<void> = new Subject();
  selection = new SelectionModel<Structure>(true, []);

  structure: Structure;
  structures: Structure[];
  totalStructures: Structure[];
  check: boolean;
  isLoading = true;
  filter: string;
  sortActive: string;
  sortDirection: string;
  currentPage: number;
  lastPage: number;
  items: number[] = [];
  pageSize: number;
  showItems = false;
  delete = false;

renderedData: Structure[];
```

```
public displayedColumns = ["select", "structureID", "name", "description"];

public dataSource = new MatTableDataSource<Structure>();


constructor(
  private structureService: StructureService,
  private papa: Papa,
  public dialog: MatDialog,
  private changeDetectorRefs: ChangeDetectorRef,
  private router: Router
) { }

ngOnInit() {
  this.getStructures();
  this.currentPage = 1;
  this.pageSize = 10;
  this.filter = "";
  this.sortActive = "name";
  this.sortDirection = "asc";
  this.getStructuresByPage(this.filter, this.currentPage, this.pageSize, this.sortActive, this.sortDirection);
}

getStructuresByPage(filter, pageIndex, pageSize, field, order): void {
  this.structureService
    .getStructuresByPage(filter, pageIndex, pageSize, field, order)
    .subscribe(
      structures => {
        this.isLoading = false,
          this.dataSource.data = structures,
          this.structures = structures
      }
    );
}

getStructures(): void {
  this.structureService
```

```
.getStructures()
.pipe(takeUntil(this.unsubscribe))
.subscribe(structures => {
  this.totalStructures = structures;
  this.lastPage = Math.ceil(structures.length / this.pageSize);

  if (!this.delete) {
    if (this.lastPage < 5) {
      this.createRange(1, this.lastPage);
    } else {
      this.createRange(1, 5);
    }
  } else {
    if (this.filter != "") {
      this.lastPage = Math.ceil(this.totalStructures.filter(a => new RegExp(this.filter, 'i').test(a.structureID.toString())
      || new RegExp(this.filter, 'i').test(a.name) || new RegExp(this.filter, 'i').test(a.description)).length / this.pageSize);
    }

    if (this.lastPage < 5) {
      this.createRange(1, this.lastPage);
      if (this.currentPage > this.lastPage)
        this.currentPage = this.lastPage;
    } else if (this.currentPage < 5) {
      this.createRange(1, 5);
    } else {
      if (this.currentPage >= this.lastPage) {
        this.createRange(this.lastPage - 4, this.lastPage);
        this.currentPage = this.lastPage;
      } else if (this.lastPage - this.currentPage == 1) {
        this.createRange(this.currentPage - 3, this.currentPage + 1);
      } else {
        this.createRange(this.currentPage - 2, this.currentPage + 2);
      }
    }
    this.getStructuresByPage(this.filter, this.currentPage, this.pageSize, this.sortActive, this.sortDirection);

    this.delete = false;
  }
```

```
      });
  }

//Impot file csv
  importFile(evt) {
    this.isLoading = true;
    this.selection.clear();
    this.currentPage = 1;
    this.filter = "";
    let repetidos = 0;
    let nRepetidos = 0;
    this.dataSource.data = [];
    var files = evt.target.files; // FileList object
    var file = files[0];
    var reader = new FileReader();
    let insertedStructures = [] as Structure[];
    reader.readAsText(file);
    reader.onload = (event: any) => {
      let csv = event.target.result; // Content of CSV file
      this.papa.parse(csv, {
        skipEmptyLines: true,
        header: true,
        complete: results => {
          if (results.meta.fields[0] == "structureID" &&
            results.meta.fields[1] == "name" &&
            results.meta.fields[2] == "description"
          ) {
            for (let i = 0; i < results.data.length; i++) {

              if (this.totalStructures.find(str => (str.structureID == results.data[i].structureID && str.name == results.data[i].name
                && str.description == results.data[i].description))
                || insertedStructures.find((str => (str.structureID == results.data[i].structureID && str.name == results.data[i].name
                && str.description == results.data[i].description)))) {

                repetidos++;
              } else {
                nRepetidos++;
```

```
                insertedStructures.push(results.data[i]);
              }
            }

            if (nRepetidos > 0) {
              this.structureService.postStructures(insertedStructures)
                .pipe(takeUntil(this.unsubscribe))
                .subscribe(result => {
                  this.openDialogWarning(repetidos, nRepetidos, false, '');
                  evt.target.value = '';
                  this.getStructuresByPage(this.filter, this.currentPage, this.pageSize, this.sortActive, this.sortDirection);
                });
            } else {
              this.openDialogWarning(repetidos, nRepetidos, false, '');
              evt.target.value = '';
              this.getStructuresByPage(this.filter, this.currentPage, this.pageSize, this.sortActive, this.sortDirection);
            }
          } else {
            this.openDialogWarning(repetidos, nRepetidos, true, 'The fields of the file do not match');
            evt.target.value = '';
            this.getStructuresByPage(this.filter, this.currentPage, this.pageSize, this.sortActive, this.sortDirection);

          }
        }
      });
    };
  }

filterStructures(evt) {
    this.filter = evt.target.value;
    if (this.filter == '') {
      this.lastPage = Math.ceil(this.totalStructures.length / this.pageSize);
      this.currentPage = 1;

      if (this.lastPage < 5) {
        this.createRange(1, this.lastPage);
      } else {
        this.createRange(1, 5);
```

```
      }
    } else {
      this.lastPage = Math.ceil(this.totalStructures.filter(a => new RegExp(this.filter, 'i').test(a.structureID.toString()) || new
       RegExp(this.filter, 'i').test(a.name) || new RegExp(this.filter, 'i').test(a.description)).length / this.pageSize);
      this.currentPage = 1;

      if (this.lastPage < 5) {
        this.createRange(1, this.lastPage);
      } else {
        this.createRange(1, 5);
      }
    }
    this.getStructuresByPage(this.filter, this.currentPage, this.pageSize, this.sortActive, this.sortDirection);
    this.selection.clear();
  }


  sortData(sort: Sort) {
    if (sort.direction == "") {
      this.sortActive = "name";
      this.sortDirection = "asc";
      this.getStructuresByPage(this.filter, this.currentPage, this.pageSize, this.sortActive, this.sortDirection);
    } else {
      this.sortActive = sort.active;
      this.sortDirection = sort.direction;
      this.getStructuresByPage(this.filter, this.currentPage, this.pageSize, this.sortActive, this.sortDirection);
    }
    this.selection.clear();
  }

  createRange(index, lastItem) {
    this.items = [];
    for (let i = index; i <= lastItem; i++) {
      this.items.push(i);
    }
  }

  onPagination(option: string) {
```

```
if (!((this.currentPage == 1 && (option == 'back' || option == 'first')) || (this.currentPage == this.lastPage && (option ==
'forward' || option == 'last')))) {
  this.selection.clear();
  switch (option) {
    case 'first':
      this.currentPage = 1;
      if (this.lastPage < 5) {
        this.createRange(this.currentPage, this.lastPage);
      } else {
        this.createRange(this.currentPage, 5);
      }
      break;
    case 'back':
      if (this.currentPage > 1) this.currentPage--;
      if (this.lastPage - this.currentPage > 1) {
        if (this.items[0] != 1) {
          this.createRange(this.currentPage - 2, this.currentPage + 2);
        }
      }
      break;
    case 'forward':
      if (this.lastPage > this.currentPage) {
        this.currentPage++;
        if (this.lastPage - this.currentPage > 1) {
          if (this.items.indexOf(this.currentPage) > 2) {
            this.createRange(this.currentPage - 2, this.currentPage + 2);
          }
        }
      }
      break;
    case 'last':
      this.currentPage = this.lastPage;
      if (this.lastPage > 4) {
        this.createRange(this.currentPage - 4, this.currentPage);
      }
      break;
  }
  this.getStructuresByPage(this.filter, this.currentPage, this.pageSize, this.sortActive, this.sortDirection);
```

```
    }
  }

  onPageSelected(item) {
    this.selection.clear();
    if (item > this.currentPage) {
      if (this.lastPage >= item) {
        this.currentPage = item;
        if (this.lastPage - this.currentPage > 1) {
          if (this.items.indexOf(this.currentPage) > 2) {
            this.createRange(this.currentPage - 2, this.currentPage + 2);
          }
        } else if (this.lastPage - this.currentPage == 1) {
          if (this.lastPage > 4)
            this.createRange(this.currentPage - 3, this.currentPage + 1);
        }
      }
    } else if (item < this.currentPage) {
      this.currentPage = item;
      if (this.lastPage - this.currentPage > 1) {
        if (item > 2) {
          this.createRange(this.currentPage - 2, this.currentPage + 2);
        } else if (item == 2) {
          if (this.lastPage > 4) {
            this.createRange(this.currentPage - 1, this.currentPage + 3);
          }
        }
      }
    }

    this.getStructuresByPage(this.filter, this.currentPage, this.pageSize, this.sortActive, this.sortDirection);
  }

  selectPageSize(pageSize) {
    this.selection.clear();
    this.pageSize = pageSize;
    this.showItems = false;
    this.currentPage = 1;
```

```
    this.getStructuresByPage(this.filter, this.currentPage, this.pageSize, this.sortActive, this.sortDirection);

    if (this.filter == '') {
      this.lastPage = Math.ceil(this.totalStructures.length / this.pageSize);
      this.currentPage = 1;

      if (this.lastPage < 5) {
        this.createRange(1, this.lastPage);
      } else {
        this.createRange(1, 5);
      }
    } else {
      this.lastPage = Math.ceil(this.totalStructures.filter(a => new RegExp(this.filter, 'i').test(a.structureID.toString()) || new
RegExp(this.filter, 'i').test(a.name) || new RegExp(this.filter, 'i').test(a.description)).length / this.pageSize);

      if (this.lastPage < 5) {
        this.createRange(1, this.lastPage);
      } else {
        this.createRange(1, 5);
      }
    }
  }

  // HANDLE CLICK OUTSIDE PAGE SIZE SELECTOR

  onClick(event) {
    if (event.target.className != 'selector-item-hide' && event.target.className != 'selector' && event.target.className != 'reverse-
    arrow') {
      this.showItems = false;
    }
  }

  isAllSelected() {
    const numSelected = this.selection.selected.length;
    const numRows = this.structures.length;
    return numSelected === numRows;
  }
```

```
masterToggle() {
  this.isAllSelected() ? this.selection.clear() : this.selectRows();
}

selectRows() {

  for (var i = 0; i < this.structures.length; i++) {
    this.selection.select(this.structures[i]);
  }
}

//Dialog Create

openDialogCreate() {
  const dialog = this.dialog.open(CreateStructureComponent, {
    panelClass: 'costum-dialog',
    width: "323px"
  });
  dialog.afterClosed()
    .pipe(takeUntil(this.unsubscribe))
    .subscribe(result => {
      this.currentPage = 1;
      this.filter = "";
      this.getStructures();
      this.selection.clear();
    });
}

//Dialog warning
openDialogWarning(repetidos, nRepetidos, headerFlag, message) {
  this.dialog.open(WarningModelComponent, {
    panelClass: "costum-dialog",
    height: "220px",
    width: "250px",
    data: {
      dadosNaoInseridos: repetidos,
      dadosInseridos: nRepetidos,
      headerFlag: headerFlag,
```

```
        message: message
      }
    });
  }

  openDialogConfirmDelete() {
    if (this.selection.selected.length > 0) {
      const dialog = this.dialog.open(ConfirmDeleteStructureComponent, {
        height: "200px",
        width: "250px",
        data: { structures: this.selection.selected }
      });
      dialog.afterClosed()
        .pipe(takeUntil(this.unsubscribe))
        .subscribe(result => {
          this.delete = true;
          this.getStructures();
          this.selection.clear();
        });

    }
  }

  openDetails() {
    console.log(this.selection.selected[0].structureID);
    this.router.navigate(['/pm_structure/details/'], { queryParams: { id: this.selection.selected[0].structureID } });
  }

  ngOnDestroy() {
    this.unsubscribe.next();
    this.unsubscribe.complete();
  }
}
```

A 1.3. Código HTML Structure

O código a seguir consiste no ficheiro HTML da interface da classe “Structure”.

```
<div class="structures-content">
    <mat-card>
      <div class="main-container">
        <div class="container">
          <div class="tableContainer">
            <table mat-table [dataSource]="dataSource" matSort *ngIf="!isLoading" (matSortChange)="sortData($event)">

              <ng-container matColumnDef="select">
                <th mat-header-cell *matHeaderCellDef>
                  <mat-checkbox (change)="$event ? masterToggle() : null"
                                [checked]="selection.hasValue() && isAllSelected()"
                                [indeterminate]="selection.hasValue() && !isAllSelected()">

                  </mat-checkbox>
                </th>
                <td mat-cell *matCellDef="let row">
                  <mat-checkbox (click)="$event.stopPropagation()"
                                (change)="$event ? selection.toggle(row) : null"
                                [checked]="selection.isSelected(row)">

                  </mat-checkbox>
                </td>
              </ng-container>

              <ng-container matColumnDef="structureID">
                <th mat-header-cell *matHeaderCellDef mat-sort-header> Structure ID </th>
                <td mat-cell *matCellDef="let element"> {{element.structureID}} </td>
              </ng-container>

              <ng-container matColumnDef="name">
                <th mat-header-cell *matHeaderCellDef mat-sort-header> Name </th>
                <td mat-cell *matCellDef="let element"> {{element.name}} </td>
              </ng-container>
```

```
      <ng-container matColumnDef="description">
        <th mat-header-cell *matHeaderCellDef mat-sort-header> Description </th>
        <td mat-cell *matCellDef="let element"> {{element.description}} </td>
      </ng-container>

      <tr mat-header-row *matHeaderRowDef="displayedColumns; sticky: true"></tr>
      <tr mat-row *matRowDef="let row; columns: displayedColumns;"></tr>
    </table>
  </div>


  <div *ngIf="isLoading" style="display: flex; justify-content: center; align-items: center">
    <mat-progress-spinner mode="indeterminate">
    </mat-progress-spinner>
  </div>

</div>

 <nav>
  <ul class="pagination">
    <li class="page-item-back" [ngClass]="{'page-item-back-disable': currentPage == 1}" (click)="onPagination('first')">
      <svg width="5.692" height="9.229" viewBox="0 0 5.692 9.229">
        <g transform="rotate(180 2.846 4.614)">
          <path id="Path_30" d="M59.727 0L58.65 1.077l3.538 3.538-3.538 3.538 1.077 1.077 4.615-4.615z" class="svg" data-
          name="Path 30" transform="translate(-58.65)" />
        </g>
      </svg>

      <svg width="5.692" height="9.229" viewBox="0 0 5.692 9.229">
        <g transform="rotate(180 2.846 4.614)">
          <path id="Path_30" d="M59.727 0L58.65 1.077l3.538 3.538-3.538 3.538 1.077 1.077 4.615-4.615z" class="svg" data-
          name="Path 30" transform="translate(-58.65)" />
        </g>
      </svg>
    </li>
    <li class="page-item-back" [ngClass]="{'page-item-back-disable': currentPage == 1}" (click)="onPagination('back')">
      <svg width="5.692" height="9.229" viewBox="0 0 5.692 9.229">
```

```
        <g transform="rotate(180 2.846 4.614)">
          <path id="Path_30" d="M59.727 0L58.65 1.077l3.538 3.538-3.538 3.538 1.077 1.077 4.615-4.615z" class="svg" data-
           name="Path 30" transform="translate(-58.65)" />
        </g>
      </svg>
    </li>
    <li class="page-item" [ngClass]="{'page-item-active': currentPage == item}" *ngFor="let item of items; "
    (click)="onPageSelected(item)">{{item}}</li>
    <li class="page-item-forward" [ngClass]="{'page-item-forward-disable': currentPage == lastPage}"
    (click)="onPagination('forward')">
      <svg width="5.692" height="9.229" viewBox="0 0 5.692 9.229">
        <g id="chevron-right">
          <path id="Path_30" d="M59.727 0L58.65 1.077l3.538 3.538-3.538 3.538 1.077 1.077 4.615-4.615z" class="svg" data-
           name="Path 30" transform="translate(-58.65)" />
        </g>
      </svg>
    </li>
    <li class="page-item-forward" [ngClass]="{'page-item-forward-disable': currentPage == lastPage}"
    (click)="onPagination('last')">
      <svg width="5.692" height="9.229" viewBox="0 0 5.692 9.229">
        <g id="chevron-right">
          <path id="Path_30" d="M59.727 0L58.65 1.077l3.538 3.538-3.538 3.538 1.077 1.077 4.615-4.615z" class="svg" data-
           name="Path 30" transform="translate(-58.65)" />
        </g>
      </svg>

      <svg width="5.692" height="9.229" viewBox="0 0 5.692 9.229">
        <g id="chevron-right">
          <path id="Path_30" d="M59.727 0L58.65 1.077l3.538 3.538-3.538 3.538 1.077 1.077 4.615-4.615z" class="svg" data-
           name="Path 30" transform="translate(-58.65)" />
        </g>
      </svg>
    </li>
  </ul>

  <div class="pageSizeSelector">
    <div class="selector-container">
      <div class="selector" (click)="showItems = !showItems">
```

```
                {{pageSize}}
                <svg width="13.07" height="6.535" viewBox="0 0 13.07 6.535" [ngClass]="{'reverse-arrow': showItems}">
                  <g data-name="drop-down-arrow (1)" transform="translate(0 -63.75)">
                    <g transform="translate(0 63.75)">
                      <path id="Path_80" d="M0 63.75l6.535 6.535 6.535-6.535z" class="svg" data-name="Path 80" transform="translate(0
                       -63.75)" />
                    </g>
                  </g>
                </svg>
              </div>
              <ul class="selector-items" [ngClass]="{'selector-items-active': showItems}">
                <li (click)="selectPageSize(10)" [ngClass]="{'selector-item-hide': pageSize == 10}">10</li>
                <li (click)="selectPageSize(20)" [ngClass]="{'selector-item-hide': pageSize == 20}">20</li>
                <li (click)="selectPageSize(50)" [ngClass]="{'selector-item-hide': pageSize == 50}">50</li>
                <li (click)="selectPageSize(100)" [ngClass]="{'selector-item-hide': pageSize == 100}">100</li>
              </ul>
            </div>
            Item per page
          </div>
        </nav>
      </div>
    </mat-card>
  </div>
</div>
```

A 1.4. Código Structure Service

O código seguinte consiste no ficheiro em typeScript da biblioteca Angular criado através de linha de comandos e com implementações de serviços criados da classe “Structure”, consoante a métodos já desenvolvidos no back-end em controlador (Controller).

```
export class StructureUserService {

  private structure_UserUrl = "/Structure_User";

  constructor(private http: HttpClient) { }

  getStructures_Users(): Observable<Structure_User[]> {
    return this.http.get<Structure_User[]>(this.structure_UserUrl);
  }

  getStructures_UsersByPage(filter: string, pageIndex: number, pageSize: number, field: string, order: string): Observable<any> {

    const url =`${this.structure_UserUrl}/page?filter=${filter}
    &pageIndex=${pageIndex}&pageSize=${pageSize}&field=${field}&order=${order}`;
    return this.http.get<Structure_User[]>(url);
  }

  getStructure_UserByUser(userID: number): Observable<any> {
    const url = `${this.structure_UserUrl}/User/${userID}`;
    return this.http.get<Structure_User[]>(url);
  }

  getStructure_UserByStructure(structureID: number): Observable<any> {

    const url = `${this.structure_UserUrl}/Structure/${structureID}`;
    return this.http.get<Structure_User[]>(url);
  }

  postStructure_User(su: Structure_User): Observable<Structure_User> {
    return this.http.post<Structure_User>(this.structure_UserUrl, su, httpOptions);
  }
```

```
  postStructures_Users(structures_Users: Structure_User[]): Observable<Structure_User[]> {
    const url = "/Structures_Users";
    return this.http.post<Structure_User[]>(
      url,
      structures_Users,
      httpOptions
    );
  }

  deleteStructure_User(structure_UserID: number): Observable<{}> {
    const url = `${this.structure_UserUrl}/${structure_UserID}`;
    return this.http.delete(url, httpOptions);
  }

}
```

A 1.5. Codigo User servisse

O código seguinte consiste no ficheiro em typeScript da biblioteca Angular criado através de linha de comandos e com implementações de serviços criados da classe “User”, consoante a métodos já desenvolvidos no back-end em controlador (Controller).

```
export class UserComponent implements OnInit, OnDestroy {

  private unsubscribe: Subject<void> = new Subject();
  selection = new SelectionModel<User>(true, []);

  user: User;
  users: User[];
  totalUsers: User[];
  finalUsers = [] as User[];
  usersToDelete = [] as User[];
  check: boolean;
  isLoading = true;
  renderedData: User[];
```

```
filter: string;
sortActive: string;
sortDirection: string;
currentPage: number;
lastPage: number;
items: number[] = [];
pageSize: number;
showItems = false;
delete = false;

public displayedColumns = ["select", "accountName", "employeeID", "company",
  "department", "costCenter", "employeeType", "employeeStatus", "manager", "jobTitle", "city", "vpn"];

public dataSource = new MatTableDataSource<User>();


@ViewChild(MatPaginator, { static: false }) set matPaginator(paginator: MatPaginator) {
  this.dataSource.paginator = paginator;
}

@ViewChild(MatSort, { static: false }) set matSort(sort: MatSort) {
  this.dataSource.sort = sort;
  this.dataSource.connect().subscribe(d => this.renderedData = d);
}

constructor(
  private userService: UserService,
  public dialog: MatDialog,
  private papa: Papa,
  private router: Router,
  private changeDetectorRefs: ChangeDetectorRef
) { }

ngOnInit() {
  this.getUsers();
  this.currentPage = 1;
  this.pageSize = 10;
  this.filter = "";
```

```
    this.sortActive = "accountName";
    this.sortDirection = "asc";
    this.getUsersByPage(this.filter, this.currentPage, this.pageSize, this.sortActive, this.sortDirection);
  }

  getUsersByPage(filter, pageIndex, pageSize, field, order): void {
    this.userService
      .getUsersByPage(filter, pageIndex, pageSize, field, order)
      .pipe(takeUntil(this.unsubscribe))
      .subscribe(
        users => {
          this.isLoading = false,
            this.dataSource.data = users,
            this.users = users
        }
      );
  }
getUsers(): void {
    this.userService
      .getUsers()
      .pipe(takeUntil(this.unsubscribe))
      .subscribe(users => {
        this.totalUsers = users,
          this.lastPage = Math.ceil(users.length / this.pageSize);


        if (!this.delete) {
          if (this.lastPage < 5) {
            this.createRange(1, this.lastPage);
          } else {
            this.createRange(1, 5);
          }
        } else {
          if (this.filter != "") {
            this.lastPage = Math.ceil(this.totalUsers.filter(u => new RegExp(this.filter, 'i').test(u.accountName)
            || new RegExp(this.filter, 'i').test(u.employeeID)
            || new RegExp(this.filter, 'i').test(u.company) || new RegExp(this.filter, 'i').test(u.department) || new
            RegExp(this.filter, 'i').test(u.costCenter)
```

```
          || new RegExp(this.filter, 'i').test(u.employeeType) || new RegExp(this.filter, 'i').test(u.employeeStatus.toString()) ||
          new RegExp(this.filter, 'i').test(u.manager)
            || new RegExp(this.filter, 'i').test(u.jobTitle) || new RegExp(this.filter, 'i').test(u.city) || new RegExp(this.filter,
           'i').test(u.vpn.toString())).length / this.pageSize);
        }

        if (this.lastPage < 5) {
          this.createRange(1, this.lastPage);
          if (this.currentPage > this.lastPage)
            this.currentPage = this.lastPage;
        } else if (this.currentPage < 5) {
          this.createRange(1, 5);
        } else {
          if (this.currentPage >= this.lastPage) {
            this.createRange(this.lastPage - 4, this.lastPage);
            this.currentPage = this.lastPage;
          } else if (this.lastPage - this.currentPage == 1) {
            this.createRange(this.currentPage - 3, this.currentPage + 1);
          } else {
            this.createRange(this.currentPage - 2, this.currentPage + 2);
          }
        }
        this.getUsersByPage(this.filter, this.currentPage, this.pageSize, this.sortActive, this.sortDirection);
        this.delete = false;
      }
    });
}

checkNullTable(lenght1, length2, attribute) {
  if (lenght1 !== length2) {
    this.displayedColumns.push(attribute);
  }
}

public doFilter = (value: string) => {
  this.selection.clear();
  this.dataSource.filter = value.trim().toLocaleLowerCase();
};
```

```
importFile(evt) {
  this.isLoading = true;
  this.selection.clear();
  this.currentPage = 1;
  this.filter = "";
  let repetidos = 0;
  let nRepetidos = 0;
  let files = evt.target.files; // FileList object
  let file = files[0];
  let reader = new FileReader();
  let insertedUsers = [] as User[];
  reader.readAsText(file);
  reader.onload = (event: any) => {
    let csv = event.target.result; // Content of CSV file
    this.papa.parse(csv, {
      skipEmptyLines: true,
      header: true,
      complete: results => {
        if (results.meta.fields[0] == "accountName" &&
          results.meta.fields[1] == "employeeID" && results.meta.fields[2] == "company" &&
          results.meta.fields[3] == "department" && results.meta.fields[4] == "costCenter" &&
          results.meta.fields[5] == "employeeType" && results.meta.fields[6] == "employeeStatus" &&
          results.meta.fields[7] == "manager" && results.meta.fields[8] == "jobTitle"
          && results.meta.fields[9] == "city" && results.meta.fields[10] == "vpn"
        ) {
          for (let i = 0; i < results.data.length; i++) {
            if (this.totalUsers.find(usr => (usr.accountName == results.data[i].accountName && usr.employeeID ==
              results.data[i].employeeID && usr.company == results.data[i].company
              && usr.department == results.data[i].department && usr.costCenter == results.data[i].costCenter && usr.employeeType ==
              results.data[i].employeeType && usr.employeeStatus == results.data[i].employeeStatus && usr.manager ==
              results.data[i].manager && usr.jobTitle == results.data[i].jobTitle
              && usr.city == results.data[i].city && usr.vpn == results.data[i].vpn))

              || insertedUsers.find((usr => (usr.accountName == results.data[i].accountName && usr.employeeID ==
              results.data[i].employeeID && usr.company == results.data[i].company && usr.department == results.data[i].department
              && usr.costCenter == results.data[i].costCenter && usr.employeeType == results.data[i].employeeType
```

```
              && usr.employeeStatus == results.data[i].employeeStatus && usr.manager == results.data[i].manager && usr.jobTitle ==
              results.data[i].jobTitle && usr.city == results.data[i].city && usr.vpn == results.data[i].vpn))) {
              repetidos++;
            } else {
              nRepetidos++;
              insertedUsers.push(results.data[i]);
            }
          }

          if (nRepetidos > 0) {
            this.userService.postUsers(insertedUsers)
              .pipe(takeUntil(this.unsubscribe))
              .subscribe(result => {
                this.openDialogWarning(repetidos, nRepetidos, false, '');
                evt.target.value = '';
                this.getUsers();
                this.getUsersByPage(this.filter, this.currentPage, this.pageSize, this.sortActive, this.sortDirection);
              });
          } else {
            this.openDialogWarning(repetidos, nRepetidos, false, '');
            evt.target.value = '';
            this.getUsers();
            this.getUsersByPage(this.filter, this.currentPage, this.pageSize, this.sortActive, this.sortDirection);
          }
        } else {
          this.openDialogWarning(repetidos, nRepetidos, true, 'The fields of the file do not match');
          evt.target.value = '';
          this.getUsers();
          this.getUsersByPage(this.filter, this.currentPage, this.pageSize, this.sortActive, this.sortDirection);
        }
      },
    });
  };
}
filterUsers(evt) {
  this.filter = evt.target.value;
  if (this.filter == '') {
    this.lastPage = Math.ceil(this.totalUsers.length / this.pageSize);
```

```
      this.currentPage = 1;

      if (this.lastPage < 5) {
        this.createRange(1, this.lastPage);
      } else {
        this.createRange(1, 5);
      }
    } else {
      this.lastPage = Math.ceil(this.totalUsers.filter(u => new RegExp(this.filter, 'i').test(u.accountName) || new
       RegExp(this.filter, 'i').test(u.employeeID)
        || new RegExp(this.filter, 'i').test(u.company) || new RegExp(this.filter, 'i').test(u.department) || new RegExp(this.filter,
       'i').test(u.costCenter)
        || new RegExp(this.filter, 'i').test(u.employeeType) || new RegExp(this.filter, 'i').test(u.employeeStatus.toString()) || new
        RegExp(this.filter, 'i').test(u.manager)
        || new RegExp(this.filter, 'i').test(u.jobTitle) || new RegExp(this.filter, 'i').test(u.city) || new RegExp(this.filter,
       'i').test(u.vpn.toString())).length / this.pageSize);
      this.currentPage = 1;

      if (this.lastPage < 5) {
        this.createRange(1, this.lastPage);
      } else {
        this.createRange(1, 5);
      }
    }
    this.getUsersByPage(this.filter, this.currentPage, this.pageSize, this.sortActive, this.sortDirection);
    this.selection.clear();
  }

  sortData(sort: Sort) {
    if (sort.direction == "") {
      this.sortActive = "accountName";
      this.sortDirection = "asc";
      this.getUsersByPage(this.filter, this.currentPage, this.pageSize, this.sortActive, this.sortDirection);
    } else {
      this.sortActive = sort.active;
      this.sortDirection = sort.direction;
      this.getUsersByPage(this.filter, this.currentPage, this.pageSize, this.sortActive, this.sortDirection);
    }
```

```
    this.selection.clear();
  }

  createRange(index, lastItem) {
    this.items = [];
    for (let i = index; i <= lastItem; i++) {
      this.items.push(i);
    }
  }

  onPagination(option: string) {
    if (!((this.currentPage == 1 && (option == 'back' || option == 'first')) || (this.currentPage == this.lastPage && (option ==
'forward' || option == 'last')))) {
      this.selection.clear();
      switch (option) {
        case 'first':
          this.currentPage = 1;
          if (this.lastPage < 5) {
            this.createRange(this.currentPage, this.lastPage);
          } else {
            this.createRange(this.currentPage, 5);
          }
          break;
        case 'back':
          if (this.currentPage > 1) this.currentPage--;
          if (this.lastPage - this.currentPage > 1) {
            if (this.items[0] != 1) {
              this.createRange(this.currentPage - 2, this.currentPage + 2);
            }
          }
          break;
        case 'forward':
          if (this.lastPage > this.currentPage) {
            this.currentPage++;
            if (this.lastPage - this.currentPage > 1) {
              if (this.items.indexOf(this.currentPage) > 2) {
                this.createRange(this.currentPage - 2, this.currentPage + 2);
              }
```

```
          }
        }
        break;
      case 'last':
        this.currentPage = this.lastPage;
        if (this.lastPage > 4) {
          this.createRange(this.currentPage - 4, this.currentPage);
        }
        break;
    }
    this.getUsersByPage(this.filter, this.currentPage, this.pageSize, this.sortActive, this.sortDirection);
  }
}


onPageSelected(item) {
  this.selection.clear();
  if (item > this.currentPage) {
    if (this.lastPage >= item) {
      this.currentPage = item;
      if (this.lastPage - this.currentPage > 1) {
        if (this.items.indexOf(this.currentPage) > 2) {
          this.createRange(this.currentPage - 2, this.currentPage + 2);
        }
      } else if (this.lastPage - this.currentPage == 1) {
        if (this.lastPage > 4)
          this.createRange(this.currentPage - 3, this.currentPage + 1);
      }
    }
  } else if (item < this.currentPage) {
    this.currentPage = item;
    if (this.lastPage - this.currentPage > 1) {
      if (item > 2) {
        this.createRange(this.currentPage - 2, this.currentPage + 2);
      } else if (item == 2) {
        if (this.lastPage > 4) {
          this.createRange(this.currentPage - 1, this.currentPage + 3);
        }
```

```
      }
    }
  }
  this.getUsersByPage(this.filter, this.currentPage, this.pageSize, this.sortActive, this.sortDirection);
}

selectPageSize(pageSize) {
  this.selection.clear();
  this.pageSize = pageSize;
  this.showItems = false;
  this.currentPage = 1;
  this.getUsersByPage(this.filter, this.currentPage, this.pageSize, this.sortActive, this.sortDirection);

  if (this.filter == '') {
    this.lastPage = Math.ceil(this.totalUsers.length / this.pageSize);
    this.currentPage = 1;

    if (this.lastPage < 5) {
      this.createRange(1, this.lastPage);
    } else {
      this.createRange(1, 5);
    }
  } else {
    this.lastPage = Math.ceil(this.totalUsers.filter(u => new RegExp(this.filter, 'i').test(u.accountName) || new
   RegExp(this.filter, 'i').test(u.employeeID) || new RegExp(this.filter, 'i').test(u.company)
   || new RegExp(this.filter, 'i').test(u.department) || new RegExp(this.filter, 'i').test(u.costCenter)
   || new RegExp(this.filter, 'i').test(u.employeeType) || new RegExp(this.filter, 'i').test(u.employeeStatus.toString())
   || new RegExp(this.filter, 'i').test(u.manager)|| new RegExp(this.filter, 'i').test(u.jobTitle)
   || new RegExp(this.filter, 'i').test(u.city) || new RegExp(this.filter, 'i').test(u.vpn.toString())).length / this.pageSize);

    if (this.lastPage < 5) {
      this.createRange(1, this.lastPage);
    } else {
      this.createRange(1, 5);
    }
  }
}
```

```
openDialogWarning(repetidos, nRepetidos, headerFlag, message) {
  this.dialog.open(WarningModelComponent, {
    panelClass: "costum-dialog",
    height: "220px",
    width: "250px",
    data: {
      dadosNaoInseridos: repetidos,
      dadosInseridos: nRepetidos,
      headerFlag: headerFlag,
      message: message
    }
  });
}

openDialogCreate() {
  const dialog = this.dialog.open(CreateUserComponent, {
    panelClass: 'costum-dialog',
    width: "700px"
  });
  dialog.afterClosed()
    .pipe(takeUntil(this.unsubscribe))
    .subscribe(result => {
      this.currentPage = 1;
      this.filter = "";
      this.getUsersByPage(this.filter, this.currentPage, this.pageSize, this.sortActive, this.sortDirection);
      this.selection.clear();
    });
}

openDialogConfirmDelete() {
  if (this.selection.selected.length > 0) {
    const dialog = this.dialog.open(ConfirmDeleteUserComponent, {
      disableClose: true,
      height: "200px",
      width: "250px",
      data: { users: this.selection.selected }
    });
    dialog.afterClosed()
```

```
        .pipe(takeUntil(this.unsubscribe))
        .subscribe(result => {
          this.delete = true;
          this.getUsers();
          this.getUsersByPage(this.filter, this.currentPage, this.pageSize, this.sortActive, this.sortDirection);
          this.selection.clear();

        });
    }
  }


  // OPEN DETAILS

  openDetails() {
    this.router.navigate(['/pm_user/details/'], { queryParams: { id: this.selection.selected[0].userID } });
  }


  onClick(event) {
    if (event.target.className != 'selector-item-hide' && event.target.className != 'selector' && event.target.className != 'reverse-
arrow') {
      this.showItems = false;
    }
  }


  isAllSelected() {
    const numSelected = this.selection.selected.length;
    const numRows = this.users.length;
    return numSelected === numRows;
  }

  masterToggle() {
    this.isAllSelected() ? this.selection.clear() : this.selectRows();
  }

  selectRows() {
```

```
    for (var i = 0; i < this.users.length; i++) {
      this.selection.select(this.users[i]);
    }
  }

  ngOnDestroy() {
    this.unsubscribe.next();
    this.unsubscribe.complete();
  }
}
```

A 1.6. Código HTML User

O código a seguir consiste no ficheiro HTML da interface da classe "User".

```
<div class="users-content">
    <mat-card>
      <div class="main-container">
        <div class="container">
          <div class="tableContainer">
            <table mat-table [dataSource]="dataSource" matSort *ngIf="!isLoading" (matSortChange)="sortData($event)">

              <ng-container matColumnDef="select">
                <th mat-header-cell *matHeaderCellDef>
                  <mat-checkbox (change)="$event ? masterToggle() : null"
                                [checked]="selection.hasValue() && isAllSelected()"
                                [indeterminate]="selection.hasValue() && !isAllSelected()">

                  </mat-checkbox>
                </th>
                <td mat-cell *matCellDef="let row">
                  <mat-checkbox (click)="$event.stopPropagation()"
                                (change)="$event ? selection.toggle(row) : null"
                                [checked]="selection.isSelected(row)">

                  </mat-checkbox>
                </td>
```

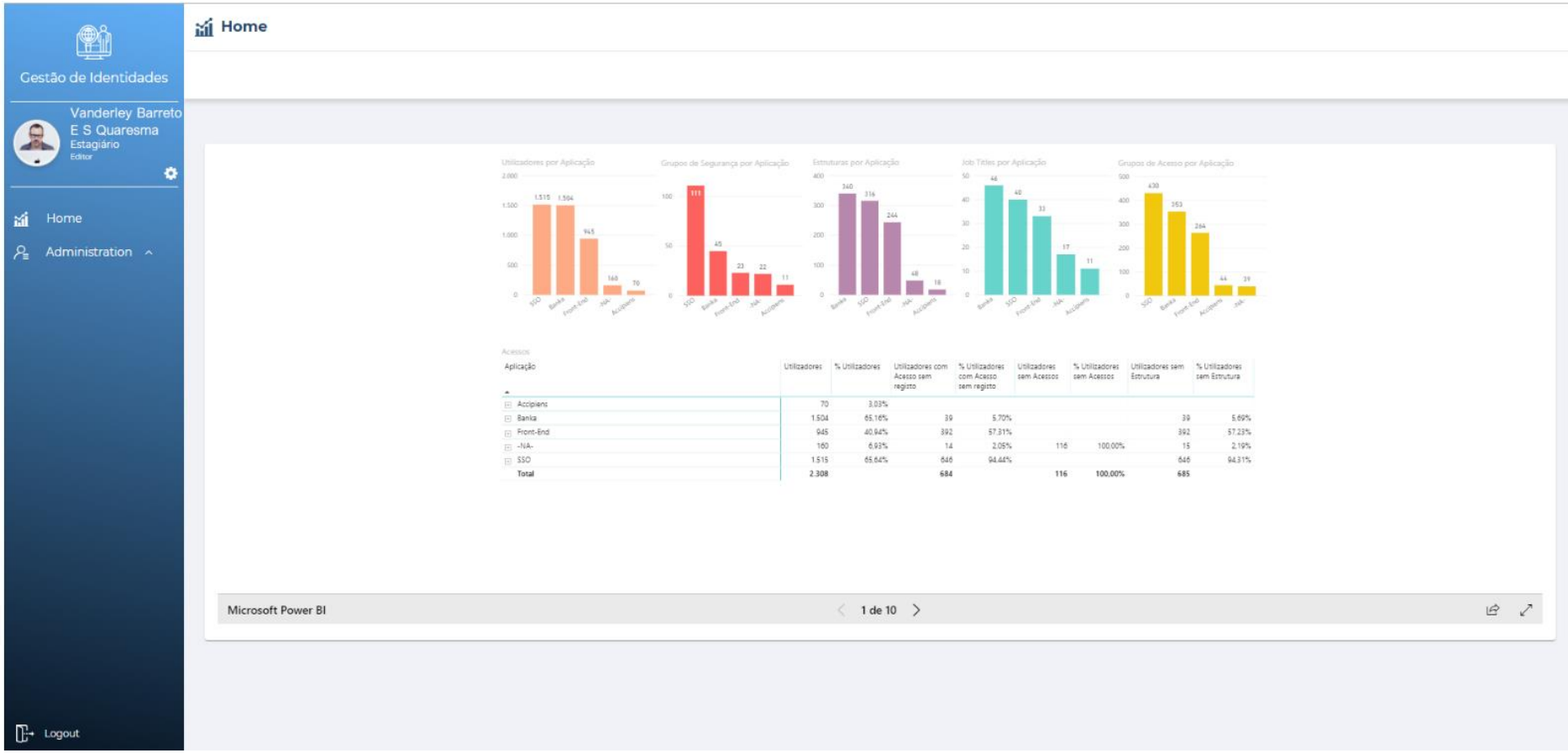
```
</ng-container>

<ng-container matColumnDef="accountName">
  <th mat-header-cell *matHeaderCellDef mat-sort-header> Account Name </th>
  <td mat-cell *matCellDef="let element"> {{element.accountName}} </td>
</ng-container>

<ng-container matColumnDef="employeeID">
  <th mat-header-cell *matHeaderCellDef mat-sort-header> Employee ID </th>
  <td mat-cell *matCellDef="let element"> {{element.employeeID}} </td>
</ng-container>

<ng-container matColumnDef="company">
  <th mat-header-cell *matHeaderCellDef mat-sort-header> Company </th>
  <td mat-cell *matCellDef="let element"> {{element.company}} </td>
</ng-container>

<ng-container matColumnDef="department">
  <th mat-header-cell *matHeaderCellDef mat-sort-header> Department </th>
  <td mat-cell *matCellDef="let element"> {{element.department}} </td>
</ng-container>

<ng-container matColumnDef="costCenter">
  <th mat-header-cell *matHeaderCellDef mat-sort-header> Cost Center </th>
  <td mat-cell *matCellDef="let element"> {{element.costCenter}} </td>
</ng-container>

<ng-container matColumnDef="employeeType">
  <th mat-header-cell *matHeaderCellDef mat-sort-header> Employee Type </th>
  <td mat-cell *matCellDef="let element"> {{element.employeeType}} </td>
</ng-container>

<ng-container matColumnDef="employeeStatus">
  <th mat-header-cell *matHeaderCellDef mat-sort-header> Employee Status </th>
  <td mat-cell *matCellDef="let element"> {{element.employeeStatus}} </td>
</ng-container>

<ng-container matColumnDef="manager">
```

```
        <th mat-header-cell *matHeaderCellDef mat-sort-header> Manager </th>
        <td mat-cell *matCellDef="let element">{{element.manager}} </td>
      </ng-container>

      <ng-container matColumnDef="jobTitle">
        <th mat-header-cell *matHeaderCellDef mat-sort-header> Job Title </th>
        <td mat-cell *matCellDef="let element"> {{element.jobTitle}} </td>
      </ng-container>

      <ng-container matColumnDef="city">
        <th mat-header-cell *matHeaderCellDef mat-sort-header> City </th>
        <td mat-cell *matCellDef="let element"> {{element.city}} </td>
      </ng-container>

      <ng-container matColumnDef="vpn">
        <th mat-header-cell *matHeaderCellDef mat-sort-header> Vpn </th>
        <td mat-cell *matCellDef="let element"> {{element.vpn}} </td>
      </ng-container>

      <tr mat-header-row *matHeaderRowDef="displayedColumns; sticky: true"></tr>
      <tr mat-row *matRowDef="let row; columns: displayedColumns;"></tr>

    </table>
  </div>

  <div *ngIf="isLoading" style="display: flex; justify-content: center; align-items: center">
    <mat-progress-spinner color="primary" mode="indeterminate">
    </mat-progress-spinner>
  </div>
</div>

<nav *ngIf="items.length > 0">
  <ul class="pagination">
    <li class="page-item-back" [ngClass]="{'page-item-back-disable': currentPage == 1}" (click)="onPagination('first')">
      <svg width="5.692" height="9.229" viewBox="0 0 5.692 9.229">
        <g id="chevron-right" transform="rotate(180 2.846 4.614)">
          <path id="Path_30" d="M59.727 0L58.65 1.077l3.538 3.538-3.538 3.538 1.077 1.077 4.615-4.615z" class="svg" data-
           name="Path 30" transform="translate(-58.65)" />
```

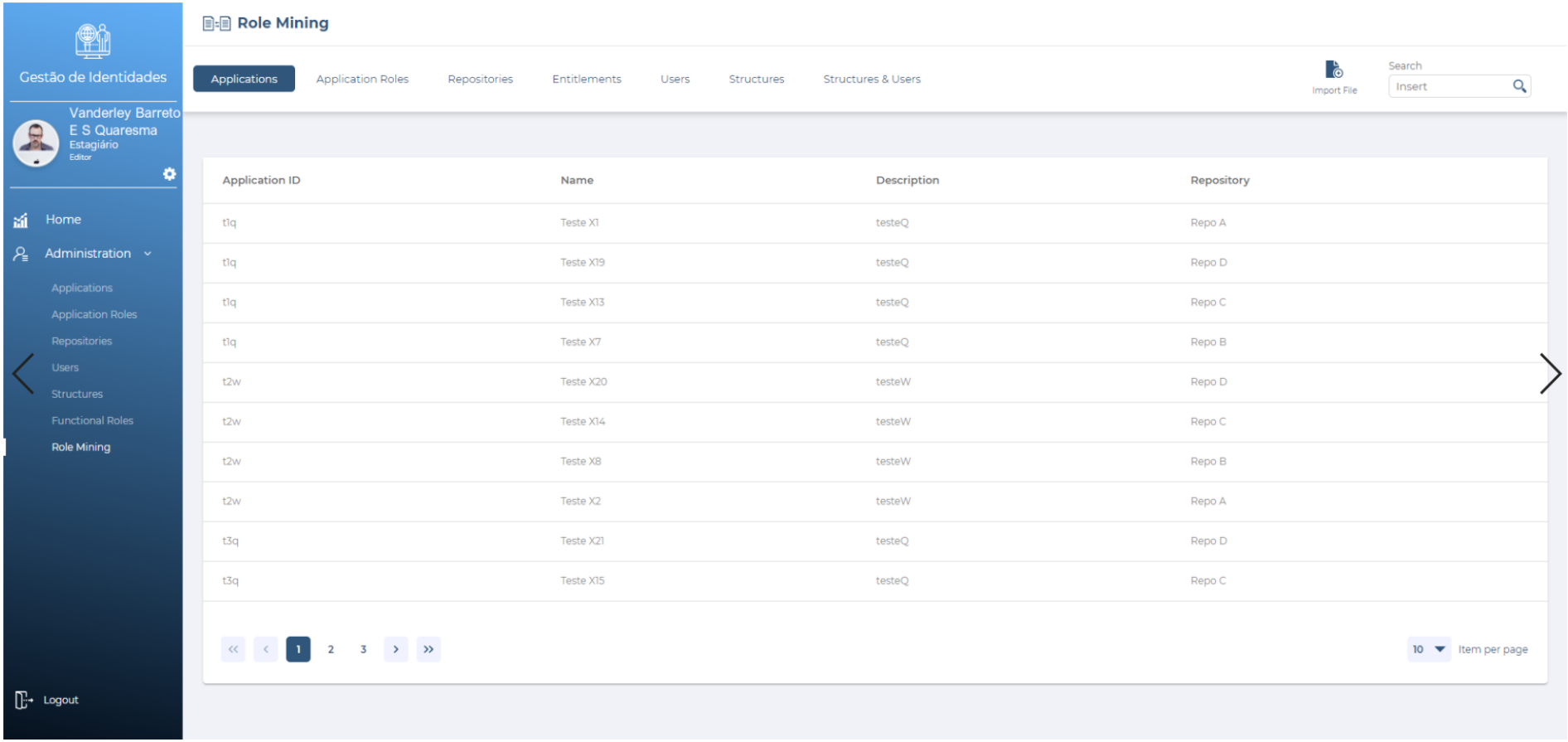
```
    </g>
  </svg>
  <svg width="5.692" height="9.229" viewBox="0 0 5.692 9.229">
    <g id="chevron-right" transform="rotate(180 2.846 4.614)">
      <path id="Path_30" d="M59.727 0L58.65 1.077l3.538 3.538-3.538 3.538 1.077 1.077 4.615-4.615z" class="svg" data-
       name="Path 30" transform="translate(-58.65)" />
    </g>
  </svg>
</li>

<li class="page-item-back" [ngClass]="{'page-item-back-disable': currentPage == 1}" (click)="onPagination('back')">
  <svg width="5.692" height="9.229" viewBox="0 0 5.692 9.229">
    <g id="chevron-right" transform="rotate(180 2.846 4.614)">
      <path id="Path_30" d="M59.727 0L58.65 1.077l3.538 3.538-3.538 3.538 1.077 1.077 4.615-4.615z" class="svg" data-
       name="Path 30" transform="translate(-58.65)" />
    </g>
  </svg>
</li>

<li class="page-item" [ngClass]="{'page-item-active': currentPage == item}" *ngFor="let item of items; "
(click)="onPageSelected(item)">{{item}}</li>

<li class="page-item-forward" [ngClass]="{'page-item-forward-disable': currentPage == lastPage}"
(click)="onPagination('forward')">
  <svg width="5.692" height="9.229" viewBox="0 0 5.692 9.229">
    <g id="chevron-right">
      <path id="Path_30" d="M59.727 0L58.65 1.077l3.538 3.538-3.538 3.538 1.077 1.077 4.615-4.615z" class="svg" data-
       name="Path 30" transform="translate(-58.65)" />
    </g>
  </svg>
</li>

<li class="page-item-forward" [ngClass]="{'page-item-forward-disable': currentPage == lastPage}"
(click)="onPagination('last')">
  <svg width="5.692" height="9.229" viewBox="0 0 5.692 9.229">
    <g id="chevron-right">
      <path id="Path_30" d="M59.727 0L58.65 1.077l3.538 3.538-3.538 3.538 1.077 1.077 4.615-4.615z" class="svg" data-
       name="Path 30" transform="translate(-58.65)" />
```

```
            </g>
          </svg>
          <svg width="5.692" height="9.229" viewBox="0 0 5.692 9.229">
            <g id="chevron-right">
              <path id="Path_30" d="M59.727 0L58.65 1.077l3.538 3.538-3.538 3.538 1.077 1.077 4.615-4.615z" class="svg" data-
               name="Path 30" transform="translate(-58.65)" />
            </g>
          </svg>
        </li>
      </ul>

      <div class="pageSizeSelector">
        <div class="selector-container">
          <div class="selector" (click)="showItems = !showItems">
            {{pageSize}}
            <svg width="13.07" height="6.535" viewBox="0 0 13.07 6.535" [ngClass]="{'reverse-arrow': showItems}">
              <g data-name="drop-down-arrow (1)" transform="translate(0 -63.75)">
                <g transform="translate(0 63.75)">
                  <path id="Path_80" d="M0 63.75l6.535 6.535 6.535-6.535z" class="svg" data-name="Path 80" transform="translate(0
                   -63.75)" />
                </g>
              </g>
            </svg>
          </div>
          <ul class="selector-items" [ngClass]="{'selector-items-active': showItems}">
            <li (click)="selectPageSize(10)" [ngClass]="{'selector-item-hide': pageSize == 10}">10</li>
            <li (click)="selectPageSize(20)" [ngClass]="{'selector-item-hide': pageSize == 20}">20</li>
            <li (click)="selectPageSize(50)" [ngClass]="{'selector-item-hide': pageSize == 50}">50</li>
            <li (click)="selectPageSize(100)" [ngClass]="{'selector-item-hide': pageSize == 100}">100</li>
          </ul>
        </div>
        Item per page
      </div>
    </nav>
  </div>
 </mat-card>
</div>
</div>
```

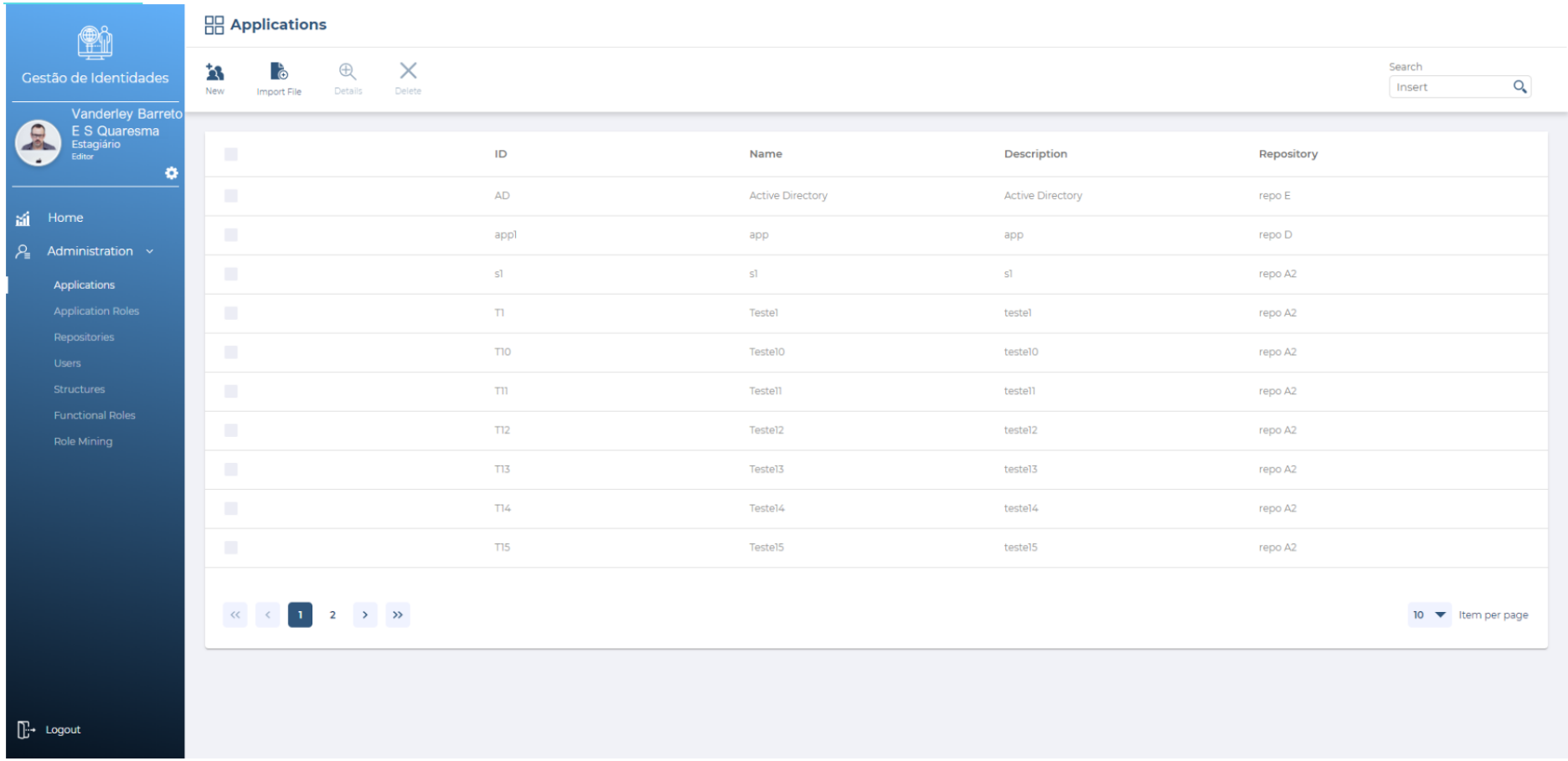
A 1.7. Ilustração da interface Home da PGPN



A 1.8. Ilustração da interface Role Minig da PGPN



A 1.9. Ilustração da interface Gestão Identidades da PGPN



A 2. Descrição de casos de uso

A 2.1. Criar Utilizador

Descrição do caso de uso para criar um utilizador.

Nome	Criar Utilizador
Descrição	O caso de uso consiste na criação de um utilizador
Pré-condição	Login válido
Caminho Principal	1. O ator seleciona o ícone “New”; 2. O sistema apresenta o formulário para criar o utilizador; 3. O ator introduz os campos: Account Name, Employee ID, Company, Department, Cost Center, Employee Type, Employee Status, Manager, Job Title, City, VPN; 4. O ator seleciona a opção “Create”; 5. O sistema cria um novo utilizador e apresenta na aba dos utilizadores.
Caminho Alternativos	4 a) O sistema devolve um alerta indicando que não estão corretos. Campos que só permitem números e estava a introduzir carateres, vice-versa.
Suplementos ou adornos	Verificação se criação é realizada com os campos devidamente preenchidos: no campo Employee Status só pode introduzir números, no Account Name só carateres com tamanho máximo 100 e no Employee ID só carateres com tamanho máximo 50, e de forma não criar campos nulos.

A 2.2. Editar Utilizadores

Descrição do caso de uso para editar um utilizador.

Nome	Editar Utilizadores
Descrição	O caso de uso consiste em editar um utilizador
Pré-condição	Login válido
Caminho Principal	1. O ator seleciona a secção do campo a editar em “Details”, “Modify”; 2. O sistema apresenta o formulário com os campos que permite editar: Account Name, Employee ID, Company, Department, Cost Center, Employee Type, Employee Status, Manager, Job Title, City, VPN;

	3. O ator introduz os dados nos campos respetivos e seleciona a opção “Modify”; 4. O sistema guarda e dá sucesso.
Caminho Alternativos	3 a) O sistema devolve um alerta indicando que não estão corretos. os campos que só permitem números e estava a introduzir carateres, vice-versa.
Suplementos ou adornos	Verificação se ao editar os campos são devidamente preenchidos: no campo Employee Status só pode introduzir números, no Account Name só carateres com tamanho máximo 100 e no Employee ID só carateres com tamanho máximo 50.

A 2.3. Consultar Utilizadores

Descrição do caso de uso para consultar um utilizador.

Nome	Consultar Utilizadores
Descrição	O caso de uso consiste na consulta de um utilizador
Pré-condição	Login válido
Caminho Principal	1. O ator seleciona na aba “User”; 2. O sistema apresenta a lista dos utilizadores com ordenação alfabética e crescente; 3. O ator pesquisa por nome; 4. O sistema apresenta a lista consoante ao nome pesquisado.
Caminho Alternativos	3 a) Não existe o nome pesquisado.

A 2.4. Eliminar Utilizador

Descrição do caso de uso para eliminar um utilizador.

Nome	Eliminar Utilizador
Descrição	O caso de uso consiste na eliminação de um utilizador
Pré-condição	Login válido
Caminho Principal	7. O ator seleciona na aba “User”; 8. O sistema apresenta a lista de utilizadores; 1. O ator seleciona o utilizador e o botão “Delete”; 2. O sistema apresenta a mensagem para confirmar a eliminação;

	3. O ator seleciona a opção “Confirm”; 4. O sistema finaliza a eliminação;
Caminho Alternativos	4 a) O sistema devolve um alerta em que os dados a eliminar têm dependências com a tabela “Entitlement” e tabela “StructureUser” e não pode eliminar.
Suplementos ou adornos	Verificação se a eliminação é realizada com os campos sem dependências.

A 2.5. Importar Utilizadores

Descrição do caso de uso para importar os dados dos utilizadores dum ficheiro CSV.

Nome	Importar Utilizadores
Descrição	O caso de uso consiste na importação dos utilizadores dum ficheiro CSV.
Pré-condição	Login válido
Caminho Principal	1. O ator seleciona a aba “User”; 2. O sistema apresenta vista dos utilizadores; 3. O ator seleciona o ícone “Import File” 4. O sistema apresenta a janela de ambiente de trabalho para seleção do ficheiro; 5. O ator seleciona o ficheiro CSV que pretende importar; 6. O sistema atualiza a vista das estruturas com os dados importados;
Caminho Alternativos	5 a) O sistema devolve um alerta, em que os campos do ficheiro á importar não são compatíveis com os do sistema. Por exemplo o campo “User ID” só permite números e o ficheiro á importar conter no campo carateres.
Suplementos ou adornos	Verificação se a importação é realizada com os campos no ficheiro devidamente corretos, por exemplo na coluna Application ID só deve conter números.

A 2.6. Adicionar Funções a Utilizador

Descrição do caso de uso para adicionar funções a um utilizador.

Nome	Adicionar Funções a utilizador
Descrição	O caso de uso consiste no adicionar funções a um utilizador.
Pré-condição	Login válido
Caminho Principal	1. O ator seleciona a aba dos utilizadores; 2. O sistema apresenta vista dos utilizadores; 3. O ator seleciona o utilizador e o botão “Details” 4. O sistema apresenta a vista de detalhes; 5. O ator seleciona a aba das funções; 6. O sistema apresenta lista das funções; 7. O ator clica no ícone “+”, seleciona a função e clica no botão “Add”; 8. O sistema atualiza secção das funções com as funções adicionadas.
Caminho Alternativos	O ator cancela a operação adiciona.