
# My.IPB

**Jorge Miguel Varandas Torgo - a41113**

Projeto apresentado à Escola Superior de Tecnologia e Gestão de Bragança para obtenção do Grau de Mestre em Informática.

Trabalho orientado por:

Prof. Albano Alves

Bragança

Novembro 2025


# My.IPB

**Jorge Miguel Varandas Torgo - a41113**

Projeto apresentado à Escola Superior de Tecnologia e Gestão de Bragança para obtenção do Grau de Mestre em Informática.

Trabalho orientado por:

Prof. Albano Alves

Bragança

Novembro 2025

# Resumo

Este projeto aborda a fragmentação dos serviços digitais no Instituto Politécnico de Bragança (IPB), onde a comunidade académica necessitava de se autenticar repetidamente em múltiplas plataformas. O principal objetivo foi desenvolver o portal My.IPB, uma plataforma centralizada para unificar o acesso a todos os recursos digitais da Instituição.

Para alcançar este objetivo, foi implementada uma solução de Single Sign-On (SSO) utilizando Keycloak, integrada com um backend desenvolvido em Spring Boot e um frontend em Angular. A metodologia focou-se numa arquitetura desacoplada, onde o portal atua como um agregador, permitindo o acesso direto a módulos específicos das plataformas integradas. A solução foi também estendida à aplicação móvel "On-line.IPB". Adicionalmente, foi criado um backoffice para a gestão centralizada de plataformas e permissões, e um pipeline de CI/CD para automatizar a entrega da aplicação.

Como resultado, o My.IPB tira partido da funcionalidade do Single Sign-On, eliminando a necessidade de múltiplos logins. O sucesso da integração de três plataformas como prova de conceito validou a arquitetura e o processo de integração. O projeto conclui-se com a disponibilização de uma plataforma robusta e escalável que moderniza o acesso aos serviços digitais do IPB.

**Palavras-chave:** Single Sign-On, Portal Institucional, Gestão de Identidade

# Abstract

This project addresses the fragmentation of digital services at the Polytechnic Institute of Bragança (IPB), where the academic community needed to repeatedly authenticate across multiple platforms. The main objective was to develop the My.IPB portal, a centralized platform to unify access to all of the Institution's digital resources.

To achieve this goal, a Single Sign-On (SSO) solution was implemented using Keycloak, integrated with a backend developed in Spring Boot and a frontend in Angular. The methodology focused on a decoupled architecture, where the portal acts as an aggregator, allowing direct access to specific modules of the integrated platforms. The solution was also extended to the "On-line.IPB"mobile application. Additionally, a backoffice was created for centralized management of platforms and permissions, and a CI/CD pipeline was implemented to automate application delivery.

As a result, My.IPB takes advantage of the Single Sign-On functionality, eliminating the need for multiple logins. The successful integration of three platforms as a proof of concept validated the architecture and the integration process. The project concludes with the delivery of a robust and scalable platform that modernizes access to IPB's digital services.

**Keywords:** Single Sign-On, Institutional Portal, Identity Management

# Conteúdo

# Lista de Tabelas

# Lista de Figuras

# Siglas

**API** Application Programming Interface. viii, 10–14, 16, 17, 20, 26, 32

**CI/CD** Continuous integration/Continuous delivery - (Integração contínua). viii, 3, 4, 27

**HTTP** Hypertext Transfer Protocol. 14

**IPB** Instituto Politécnico de Bragança. 1, 5–10, 14, 15, 18, 25, 26, 30–32

**JWT** JSON Web Token. 12–16, 20

**LDAP** Lightweight Directory Access Protocol. 5, 7, 9

**ORM** Object-Relational Mapping. 29

**REST** Representational State Transfer. 10–12, 17, 20

**SSO** Single Sign-On. vii, ix, 1–3, 6, 7, 9, 10, 13–15, 20, 31, 32

**URL** Uniform Resource Locator. 15, 21, 26

# Capítulo 1

# Introdução

O Instituto Instituto Politécnico de Bragança (IPB) disponibiliza um vasto leque de plataformas e serviços online aos seus estudantes, docentes e funcionários. Embora a comunidade académica utilize uma única credencial institucional, a experiência de acesso aos recursos digitais é fragmentada. Os utilizadores necessitam de iniciar sessão individualmente em cada plataforma, um processo repetitivo que consome tempo e não contribui para uma navegação fluida entre os diferentes sistemas.

Este projeto visa dar resposta a este desafio através do desenvolvimento do portal My.IPB, uma plataforma centralizada que atua como um ponto de entrada único para todos os recursos digitais do IPB. A principal inovação do My.IPB é a implementação de uma solução de SSO, que unifica o processo de autenticação. Com o SSO, após um único login no portal, os utilizadores podem aceder a todas as plataformas integradas sem necessidade de se autenticarem novamente, simplificando drasticamente a gestão de sessões.

O sistema de autenticação, baseado em OAuth2/OpenID, implementado pelo Keycloak, permite que, após um único login no portal My.IPB, os utilizadores naveguem para qualquer plataforma ou módulo a que tenham permissão, sendo automaticamente reconhecidos e autenticados. Este mecanismo, para além de ser mais conveniente reforça a segurança, ao centralizar a gestão de identidades e acessos.

## 1.1 Enquadramento

O desenvolvimento do portal My.IPB enquadra-se na estratégia de modernização e transformação digital do Instituto Politécnico de Bragança. A proposta original do projeto, que pode ser consultada na íntegra no apêndice A, estabelece as bases para a criação de uma solução integradora que responda às necessidades da comunidade académica. O projeto foi desenvolvido na Divisão de Informática do IPB, em colaboração com a equipa de desenvolvimento existente, o que permitiu um alinhamento constante com os requisitos e a infraestrutura da Instituição.

## 1.2 Objetivos

Os objetivos deste trabalho, em linha com a proposta original do projeto, são os seguintes:

- **Adoção de uma solução de SSO:** Comparar, escolher e instalar um sistema de autenticação centralizado que unifique a experiência de login na web e no mobile.
- **Criar um portal personalizável:** Desenvolver um dashboard onde os utilizadores possam parametrizar e configurar o layout, adicionando atalhos para as plataformas e módulos que mais utilizam.
- **Garantir acesso direto a módulos específicos:** Permitir, onde possível, que os utilizadores acedam diretamente a funcionalidades específicas dentro de cada plataforma, sem necessidade de navegar por menus intermédios.
- **Estender o SSO à aplicação On-line.IPB:** Assegurar que a aplicação móvel existente do IPB se integra com a solução de SSO, partilhando a sessão do utilizador com o portal web.
- **Desenvolver um módulo de backoffice:** Criar uma interface de administração para a gestão centralizada de plataformas e permissões.

- **Implementar um pipeline de CI/CD:** Garantir a entrega contínua e automatizada da aplicação, desde o controlo de versões até à produção, para assegurar a qualidade e agilidade do processo de desenvolvimento.

Para alcançar estes objetivos, foram utilizadas as seguintes tecnologias:

- **Spring Boot:** Framework para o desenvolvimento do backend da aplicação.
- **Angular:** Framework para o desenvolvimento do frontend do portal.
- **Expo (React Native):** A biblioteca 'expo-auth-session' foi utilizada para implementar a autenticação na aplicação móvel On-line.IPB.
- **Oracle Database:** Sistema de gestão de bases de dados para o armazenamento de dados da aplicação.
- **OAuth2/OpenID e Keycloak:** Protocolo de autorização e plataforma de gestão de identidade e acessos, utilizados para a implementação da solução de SSO.

## 1.3 Estrutura do Documento

Este documento está organizado da seguinte forma:

- **Capítulo 2 - Análise e Desenho da Solução:** Descreve o levantamento de requisitos, a análise de soluções e o desenho da arquitetura do portal My.IPB.
- **Capítulo 3 - Implementação da Solução de SSO:** Detalha a implementação da solução de Single Sign-On para a web e a sua extensão à aplicação móvel On-line.IPB.
- **Capítulo 4 - Desenvolvimento do Portal My.IPB:** Apresenta o desenvolvimento do frontend e do backend do portal.
- **Capítulo 5 - Backoffice e Integração de Plataformas:** Descreve a criação do módulo de administração e o processo de integração de novos sistemas.

- **Capítulo 6 - Infraestrutura e Automação de CI/CD:** Detalha o pipeline de CI/CD, a estratégia de contentorização e a configuração do ambiente de produção.
- **Capítulo 7 - Conclusões e Trabalho Futuro:** Apresenta as conclusões do trabalho realizado, avalia o cumprimento dos objetivos e propõe direções para trabalho futuro.

# Capítulo 2

# Análise e Desenho da Solução

Este capítulo descreve o processo de análise de requisitos que norteou o projeto e apresenta o desenho da arquitetura da solução My.IPB. A arquitetura foi concebida para ser modular, escalável e segura, utilizando tecnologias modernas e alinhadas com o ecossistema tecnológico do IPB.

## 2.1 Levantamento do Portfólio de Aplicações

O primeiro passo do projeto consistiu num levantamento e caracterização do portfólio de aplicações existentes no Instituto Politécnico de Bragança. Esta análise revelou um ecossistema digital rico e diversificado, composto por mais de duas dezenas de sistemas, mas também altamente fragmentado do ponto de vista da autenticação do utilizador.

Apesar de a maioria das aplicações utilizar a base de dados Lightweight Directory Access Protocol (LDAP) central do IPB para validar as credenciais, cada uma implementava o seu próprio mecanismo e interface de login. O portfólio abrangia diversas áreas da vida académica e institucional, incluindo:

- **Plataformas Académicas:** `virtual.ipb.pt`, `online.ipb.pt`, `sumarios.ipb.pt`, `candidaturas.ipb.pt`
- **Sistemas Administrativos:** `gdoc.ipb.pt`, `ponto.ipb.pt`, `concursos.ipb.pt`,

`projetos.ipb.pt`

- **Ferramentas de Apoio e Recursos:** `cloud.ipb.pt`, `rfid.ipb.pt`, `webdocs.ipb.pt`

Esta fragmentação resultava numa experiência de utilização inconsistente e ineficiente, impondo um obstáculo digital às atividades diárias de estudantes, docentes e funcionários, que eram forçados a autenticarem-se repetidamente. A análise deste cenário validou a necessidade crítica de uma solução de SSO que pudesse unificar o acesso e servir de base para um portal agregador, justificando assim os objetivos delineados na proposta de projeto.

## 2.2 Levantamento de Requisitos

O levantamento de requisitos foi baseado na proposta de projeto e nas necessidades da comunidade académica do IPB. Os requisitos foram classificados em funcionais e não-funcionais.

### 2.2.1 Requisitos Funcionais

- **RF01 - Autenticação Única (SSO):** O sistema deve permitir que um utilizador se autentique uma única vez para aceder a todas as plataformas integradas.
- **RF02 - Catálogo de Plataformas:** O portal deve exibir um catálogo de todas as plataformas e serviços disponíveis.
- **RF03 - Dashboard Personalizável:** Utilizadores autenticados devem ter um dashboard pessoal onde podem adicionar, remover e organizar atalhos para as plataformas que mais utilizam.
- **RF04 - Acesso Direto a Módulos:** O utilizador deve ser capaz de clicar num atalho e ser redirecionado diretamente para a funcionalidade específica (módulo) dentro de uma plataforma, já autenticado.

- **RF05 - Backoffice de Gestão:** O sistema deve incluir uma área de administração para gerir as plataformas da própria aplicação.

- **RF06 - Gestão de Permissões de Utilizadores em Plataformas Externas:** O sistema deve fornecer uma interface para gerir as permissões de um utilizador nas plataformas externas integradas.

### 2.2.2 Requisitos Não-Funcionais

- **RNF01 - Segurança:** A comunicação e o armazenamento de dados devem ser seguros. O acesso aos recursos deve ser estritamente controlado por permissões.

- **RNF02 - Usabilidade:** A interface do portal deve ser intuitiva, responsiva e de fácil utilização para os diferentes perfis de utilizador.

- **RNF03 - Manutenibilidade:** O código deve ser bem estruturado e documentado para facilitar futuras manutenções e evoluções.

- **RNF04 - Escalabilidade:** A arquitetura deve ser capaz de suportar um número crescente de utilizadores e a integração de novas plataformas sem degradação de performance.

## 2.3 Análise de Soluções de SSO

A seleção de uma plataforma de Identity and access management (IAM) foi um passo crítico para o sucesso do projeto. A plataforma escolhida deveria não só suportar os protocolos de autenticação modernos, mas também integrar-se de forma transparente com o diretório LDAP existente no IPB e permitir um elevado grau de personalização. Para isso, foi realizado um estudo comparativo das principais soluções do mercado.

Foram analisadas quatro plataformas de referência: duas comerciais, **Okta**[1] e **Microsoft Entra ID**[2], e duas open-source, **WSO2 Identity Server**[3] e **Keycloak**[4].

- **Okta:** É uma plataforma líder no mercado de IAM, oferecida como um serviço (SaaS). A sua principal vantagem é a simplicidade de gestão e uma vasta gama de integrações pré-configuradas. No entanto, o seu modelo comercial, com custos por utilizador, e a sua natureza como solução de "caixa fechada" representavam barreiras significativas em termos de custo a longo prazo e de flexibilidade para uma Instituição pública como o IPB.

- **Microsoft Entra ID:** É a solução de identidade da Microsoft, profundamente integrada com o ecossistema Azure e Microsoft 365. Para uma organização que já utiliza intensivamente os serviços Microsoft, esta seria uma escolha natural. Contudo, para o ambiente heterogéneo do IPB, a sua implementação poderia ser mais complexa e o modelo de licenciamento poderia introduzir custos adicionais.

- **WSO2 Identity Server:** É uma plataforma open-source extremamente poderosa e flexível, com um forte foco na gestão de APIs e integrações empresariais complexas. Embora ofereça um conjunto de funcionalidades muito abrangente, a sua complexidade e curva de aprendizagem são consideravelmente mais acentuadas, exigindo uma equipa com conhecimentos mais especializados para a sua implementação e manutenção.

- **Keycloak:** É uma plataforma open-source, mantida pela Red Hat, que se posiciona como uma solução robusta e, acima de tudo, focada na experiência do developer. Oferece um conjunto de funcionalidades equiparável ao das soluções comerciais para os casos de uso mais comuns, mas com a vantagem de ser auto-hospedada, eliminando custos de licenciamento e garantindo controlo total sobre os dados e a infraestrutura.

A tabela seguinte resume a análise comparativa segundo os critérios definidos como prioritários para o projeto:

Tabela 2.1: Análise comparativa de soluções de SSO.

| **Plataforma** / **Critério** | **Okta** | **Microsoft Entra ID** | **WSO2 IS** | **Keycloak** |
|---|---|---|---|---|
| **Modelo** | SaaS (Comercial) | SaaS (Comercial) | Open-Source (Self-hosted) | Open-Source (Self-hosted) |
| **Custo** | Elevado (por utilizador) | Moderado a Elevado | Inexistente (custo de infra.) | Inexistente (custo de infra.) |
| **Integração LDAP** | Sim | Sim (com conectores) | Sim | Nativa e Extensível |
| **Protocolos** | OIDC, OAuth2, SAML | OIDC, OAuth2, SAML | OIDC, OAuth2, SAML | OIDC, OAuth2, SAML |
| **Personalização** | Limitada (UI) | Limitada | Elevada (complexa) | Elevada (simples) |
| **Curva de Aprendizagem** | Baixa | Moderada | Elevada | Baixa a Moderada |
| **Controlo de Dados** | Fornecedor | Fornecedor | Total (on-premises) | Total (on-premises) |

**Justificação da Escolha:**

A análise tornou clara a superioridade do **Keycloak** para os requisitos específicos do IPB. A sua natureza **open-source** e **auto-hospedada** foi o fator decisivo, pois oferece controlo total sobre a infraestrutura e os dados, um requisito fundamental para uma Instituição pública. A integração **nativa com LDAP** simplifica a ligação ao diretório de utilizadores existente, enquanto o suporte completo aos protocolos **OpenID Connect (OIDC)[5] e OAuth 2.0[6]** garante a compatibilidade com as tecnologias escolhidas para o desenvolvimento (Spring Boot e Angular).

Comparado com o WSO2 IS, o Keycloak apresentou uma **curva de aprendizagem mais suave** e uma **experiência de desenvolvimento mais direta** para os casos de uso de SSO e gestão de identidade necessários ao projeto, sem a complexidade adicional das funcionalidades de gestão de API mais avançadas do WSO2. A sua forte comunidade e documentação clara foram também fatores importantes, tornando-o a escolha mais pragmática e produtiva para a equipa de desenvolvimento do IPB.

## 2.4 Desenho da Arquitetura

A arquitetura da solução My.IPB foi desenhada segundo um modelo cliente-servidor, com uma separação clara de responsabilidades entre o frontend e o backend.

### 2.4.1 Arquitetura Geral

A solução é composta por três grandes blocos:

- **Frontend (Cliente):** Uma Single-Page Application (SPA) desenvolvida em **Angular**, responsável por toda a interface do utilizador. Comunica com o backend através de pedidos a uma API Representational State Transfer (REST).
- **Backend (Servidor):** Uma aplicação desenvolvida em **Spring Boot**, que expõe a API REST. É responsável pela lógica de negócio, persistência de dados e por proteger os recursos.
- **Servidor de Autenticação:** O **Keycloak**, que atua como um fornecedor de identidade centralizado, gerindo os utilizadores e o processo de login.

### 2.4.2 Arquitetura de Gestão de Permissões Distribuída

A gestão de permissões no My.IPB segue um modelo distribuído e federado. Em vez de centralizar o controlo de acesso de todas as plataformas, o My.IPB atua como um

agregador e uma interface de gestão. Cada plataforma integrada é responsável por gerir as suas próprias permissões.

O My.IPB define um contrato de API que as plataformas externas devem implementar. Este contrato especifica endpoints para:

- Obter a lista de todos os módulos disponíveis numa plataforma.
- Obter os módulos a que um utilizador específico tem acesso.
- Atualizar os módulos a que um utilizador tem acesso.

O backoffice do My.IPB, especificamente o componente 'role-manager', utiliza esta API para apresentar uma interface unificada ao administrador. Desta forma, a lógica de negócio e o armazenamento das permissões permanecem na plataforma de origem, garantindo autonomia e desacoplamento, enquanto o My.IPB oferece um ponto central de gestão.

### 2.4.3 Desenho do Backend

O backend segue uma arquitetura em camadas para promover a separação de interesses e a manutenibilidade:

- **Camada de Controlador (Controller):** Implementada no pacote "`pt.ipb.my.server.controller`", utiliza o Spring MVC para expor os endpoints da API REST, como o 'PlatformController' e o 'AuthController'.
- **Camada de Negócio (Service):** Localizada em 'pt.ipb.my.server.services', contém a lógica de negócio principal da aplicação.
- **Camada de Acesso a Dados (Repository):** O pacote "`pt.ipb.my.server.repo`" utiliza o Spring Data JPA para definir as interfaces de acesso à base de dados **Oracle**.

- **Camada de Segurança:** Os pacotes ‘pt.ipb.my.server.security‘ e “`pt.ipb.my.server.keycloak`” utilizam o Spring Security para integrar com o Keycloak, validar os tokens JSON Web Token (JWT) e aplicar as regras de autorização.

### 2.4.4 Desenho do Frontend

O frontend em Angular foi estruturado de forma modular para organizar o código de forma lógica:

- **Módulos Principais:** A aplicação está dividida em módulos como ‘public‘ (área pública), ‘my‘ (dashboard do utilizador), ‘platform-manager‘ (backoffice de gestão de plataformas) e ‘role-manager‘ (backoffice de gestão de permissões), cada um agrupando os seus componentes, rotas e serviços.
- **Componentes:** A interface é construída a partir de componentes reutilizáveis, como listas de plataformas, formulários de edição e o próprio dashboard.
- **Serviços:** A comunicação com a API REST do backend é encapsulada em serviços Angular, que são injetados nos componentes que deles necessitam.
- **Roteamento:** O ficheiro ‘app.routes.ts‘ define as rotas da aplicação, mapeando os URLs para os componentes a serem exibidos.

# Capítulo 3

# Implementação da Solução de SSO

Este capítulo detalha a implementação técnica da solução de SSO, cuja arquitetura foi descrita no capítulo anterior. A integração entre o Keycloak e o backend Spring Boot, realizada através do Spring Security, é o pilar da autenticação e autorização do portal My.IPB.

## 3.1 Configuração do Spring Security

A segurança da API REST do portal My.IPB é gerida pelo Spring Security. O ficheiro principal de configuração, "`SecurityConfigration.java`", define as regras de proteção dos endpoints. A configuração estabelece que, todos os *endpoints* são públicos por omissão e autenticados quando especificado. A especificação das permissões ou roles necessários para cada endpoint é conseguida com o uso de anotações ao nível dos métodos, nomeadamente - "`@PreAuthorize`".

A aplicação foi configurada como um "resource server" no contexto do OAuth2. Isto significa que a sua principal responsabilidade é proteger os recursos (os endpoints da API), validando os tokens de acesso (JWT) que são apresentados pelos clientes (o frontend Angular e a app móvel).

O processo de validação dos tokens JWT emitidos pelo Keycloak é customizado através da classe "`KeycloakJwtAuthenticationConverter.java`". Esta classe é responsável por

extrair as informações do token, como o nome do utilizador e as suas permissões (roles), e convertê-las num objeto de autenticação que o Spring Security consegue consumir.

## 3.2 Fluxo de Autenticação OAuth2 na Web

O fluxo de autenticação para o portal web segue o padrão "Authorization Code Flow" do OAuth2, que é considerado o mais seguro para aplicações web.

1. O utilizador acede ao frontend do My.IPB e clica no botão de login.
2. O frontend redireciona o utilizador para a página de login do Keycloak.
3. O utilizador insere as suas credenciais institucionais do IPB no Keycloak.
4. Após a autenticação bem-sucedida, o Keycloak redireciona o utilizador de volta para o frontend, enviando um código de autorização.
5. O frontend troca este código de autorização por um token de acesso (JWT) e um token de atualização (refresh token), comunicando diretamente com o Keycloak.
6. A partir deste momento, sempre que o frontend faz um pedido à API do backend, inclui o token de acesso (*Bearer*) no cabeçalho 'Authorization'.
7. O backend (Spring Boot), com a ajuda do Spring Security, valida a assinatura e a validade do token. Se o token for válido, o pedido é processado. Caso contrário, é devolvido um erro Hypertext Transfer Protocol (HTTP) (código 403).

## 3.3 Extensão ao Ambiente Móvel: On-line.IPB

A solução de SSO foi estendida à aplicação móvel **On-line.IPB**. A integração foi conseguida utilizando a biblioteca 'expo-auth-session', que facilita a implementação do fluxo de autenticação OAuth2 em aplicações React Native/Expo.

O processo é o seguinte:

1. Na app On-line.IPB, o utilizador prime o botão de login.

2. A biblioteca 'expo-auth-session' abre uma janela do navegador do sistema com a página de login do Keycloak.

3. Se o utilizador já tiver uma sessão SSO ativa no navegador do seu dispositivo (por exemplo, por ter feito login no portal My.IPB anteriormente), o Keycloak reconhece essa sessão e autentica o utilizador automaticamente.

4. Caso o utilizador não tenha uma sessão SSO ativa, é necessário inserir as suas credenciais.

5. Após a autenticação, o Keycloak redireciona para um Uniform Resource Locator (URL) específico da aplicação móvel, que é interceptado pela 'expo-auth-session'. A biblioteca extrai o código de autorização e troca-o por um token de acesso, de forma semelhante ao que acontece na web.

Esta abordagem unifica a experiência de autenticação entre a web e o mobile, permitindo que o utilizador beneficie da sua sessão SSO em todo o ecossistema digital do IPB, independentemente do dispositivo.

## 3.4 O Papel do SSO na Autorização

É fundamental distinguir o papel do SSO no processo de autenticação e no de autorização. A solução de SSO com Keycloak centraliza a **autenticação**, garantindo que o utilizador é quem diz ser. Uma vez autenticado, o Keycloak emite um token JWT que contém a identidade do utilizador e um conjunto de "roles" (papéis) associados a ele no Keycloak.

No entanto, a **autorização** (o que o utilizador pode fazer) é, por desenho da arquitetura, uma responsabilidade descentralizada. O portal My.IPB utiliza as roles do Keycloak para a sua própria gestão de acessos interna. Por exemplo, a classe "`AppPermissionEvaluator.java`" utiliza estas roles para proteger os endpoints do backoffice, garantindo que apenas administradores podem gerir plataformas.

Para as plataformas externas integradas, o comportamento é diferente. Quando um utilizador acede a uma plataforma externa através do My.IPB, o token JWT é usado para o autenticar. A plataforma externa recebe a identidade do utilizador de forma segura, mas, depois de o utilizador ser encaminhado, a decisão sobre as funcionalidades ou dados a que esse utilizador pode aceder é tomada internamente pela própria plataforma, com base nas suas próprias regras de negócio e sistemas de permissões. Algumas aplicações podem, opcionalmente, ser configuradas para consumir as roles do Keycloak, mas a arquitetura não impõe essa obrigatoriedade, promovendo o desacoplamento e a autonomia dos sistemas.

## 3.5 Integração com o Keycloak

A comunicação inicial e a configuração da ligação com o servidor Keycloak são geridas pelas classes no pacote ‘pt.ipb.my.server.keycloak‘. A classe ‘KeycloackInitializer.java‘ é executada no arranque da aplicação para garantir que a integração está corretamente configurada, enquanto a ‘KeycloakHelper.java‘ providencia métodos utilitários para interação com a API do Keycloak e conversão de dados, se necessário.

# Capítulo 4

# Desenvolvimento do Portal My.IPB

Com a arquitetura da solução definida, este capítulo foca-se na implementação das funcionalidades centrais do portal My.IPB. Serão detalhados os componentes do backend e do frontend que, em conjunto, materializam a experiência do utilizador.

## 4.1 Backend: API de Suporte ao Portal

O backend foi desenhado para fornecer os dados necessários ao frontend de forma eficiente e segura. A principal responsabilidade, no que diz respeito ao portal do utilizador, recai sobre o "`PlatformController.java`".

### 4.1.1 PlatformController

Este controlador expõe os endpoints da API REST que permitem ao frontend obter a lista de plataformas para o dashboard do utilizador autenticado. As principais funcionalidades são:

- **Listagem de Plataformas:** Um endpoint que devolve a lista de todas as plataformas configuradas no sistema.
- **Detalhes de uma Plataforma:** Um endpoint que devolve a informação detalhada de uma plataforma, incluindo todos os seus módulos.

- **Dados para o Dashboard:** Endpoint que devolve a configuração do dashboard pessoal, definindo a ordem e a visibilidade das plataformas a serem apresentadas.

Os dados são obtidos da base de dados Oracle através da camada de serviços e repositórios (Spring Data com ajuda de QueryDSL).

## 4.2 Frontend: A Interface do Utilizador

O frontend, desenvolvido em Angular, é o ponto de interação do utilizador com o portal My.IPB. A aplicação está dividida em duas áreas funcionais principais.

### 4.2.1 Área Pública

A área pública, contida no diretório "src/app/public", é a página de entrada do portal, acessível a todos os visitantes. As suas funcionalidades principais visam apresentar a atividade do IPB, funcionando como uma vitrine institucional.

Após uma breve mensagem de boas-vindas, a página apresenta uma **agenda de eventos** numa linha do tempo interativa. Esta funcionalidade, encapsulada no "TimelineComponent", obtém os dados de eventos a partir do portal de notícias do IPB ("ipbnews.ipb.pt"). O componente processa a lista de eventos para exibir os cinco mais relevantes, dando prioridade aos eventos futuros e complementando com os eventos passados mais recentes. O resultado é uma linha do tempo visualmente apelativa que oferece aos visitantes uma visão rápida sobre a atividade presente e futura do instituto.

Logo após a agenda, a página exibe as **notícias institucionais** mais recentes, obtidas da mesma fonte externa. A implementação, visível no ficheiro "`public-index.component.html`", renderiza um cartão ("<news-card>") para cada notícia, apresentando um resumo da atividade do IPB.

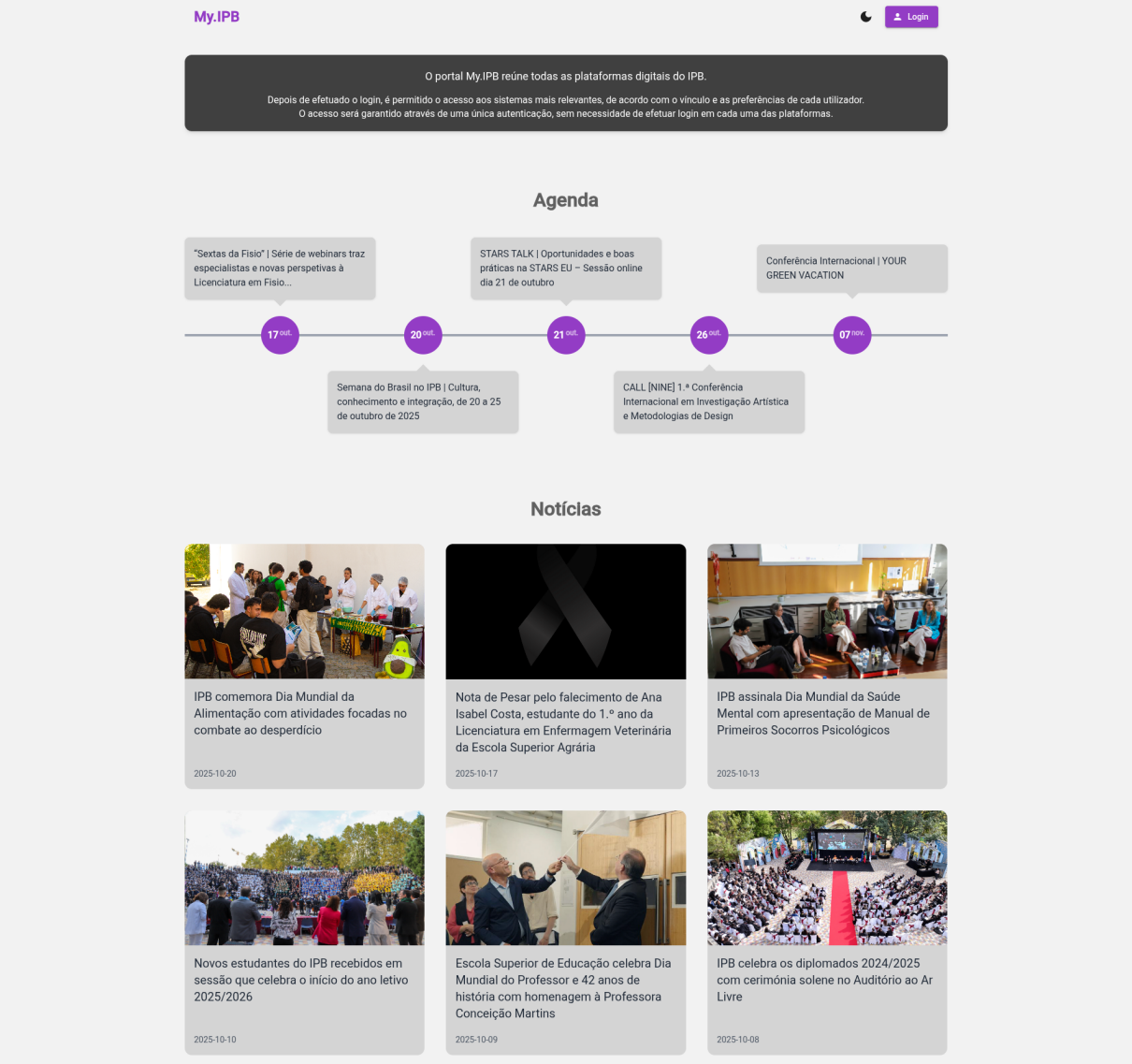


Figura 4.1: Página de entrada do portal My.IPB.

### 4.2.2 Área Privada

Após a autenticação, o utilizador acede à sua área privada do "My.IPB", contida no diretório "src/app/my". É nesta área que o utilizador tem acesso ao catálogo de plataformas e módulos. A área consiste num dashboard personalizável, que serve como centro de comando para a vida académica do utilizador.

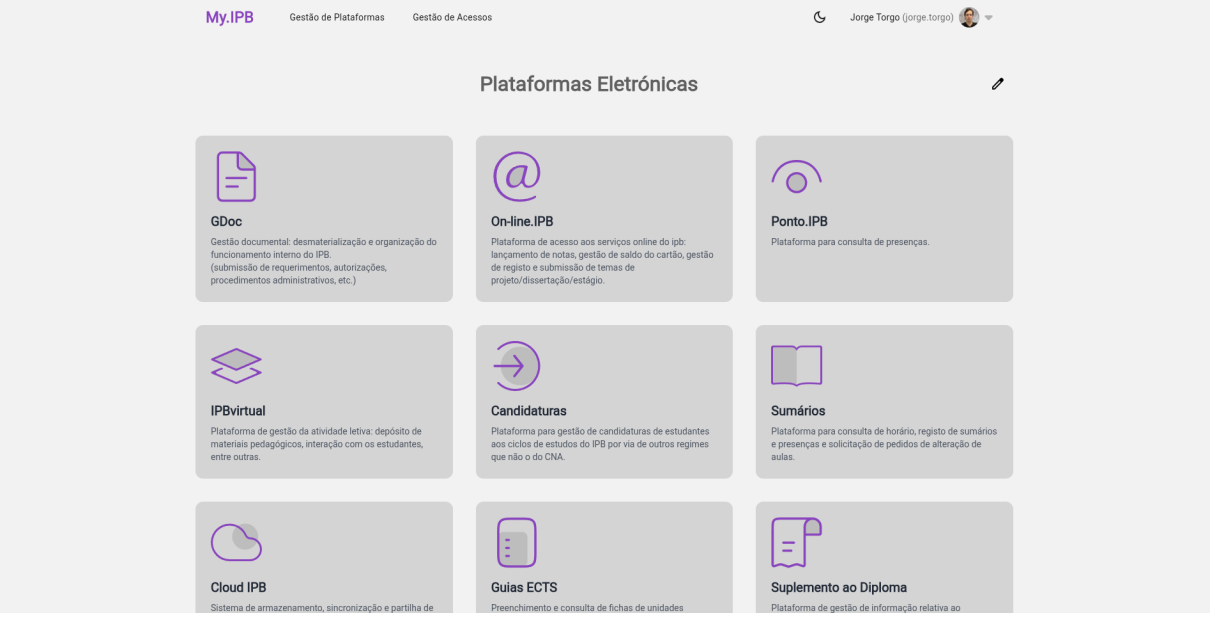


Figura 4.2: Dashboard pessoal do utilizador no portal My.IPB.

As principais funcionalidades do dashboard são:

- **Dashboard Personalizável:** O utilizador pode adicionar, remover e reorganizar atalhos para as plataformas que mais utiliza, criando uma vista adaptada às suas necessidades.
- **Acesso Direto:** Com um único clique, o utilizador é redirecionado para o módulo pretendido, já autenticado, graças à integração com a solução de SSO.
- **Comunicação com o Backend:** O frontend comunica com a API REST do backend para obter os dados das plataformas e para guardar as preferências de personalização do utilizador. Esta comunicação é realizada através de serviços Angular, que encapsulam os pedidos HTTP. Todos os pedidos a endpoints protegidos incluem o token de acesso JWT no cabeçalho, garantindo a segurança da comunicação.

A reatividade da interface, proporcionada pelo Angular, permite uma experiência de utilização fluida e moderna.

# Capítulo 5

# Backoffice e Integração de Plataformas

Um requisito fundamental do projeto My.IPB é a capacidade de gerir e integrar plataformas de forma centralizada. Para este fim, foi desenvolvido um módulo de backoffice que, mais do que uma simples ferramenta de administração, funciona como um centro de orquestração para a integração de novos sistemas no ecossistema My.IPB.

Este capítulo descreve as funcionalidades do backoffice, com especial foco na gestão de permissões, e o processo técnico de integração de uma plataforma.

## 5.1 Funcionalidades do Backoffice

O backoffice do My.IPB, implementado em Angular no diretório "src/app/platform-manager", centraliza a configuração do portal.

### 5.1.1 Gestão de Plataformas

A funcionalidade base do backoffice é o registo das plataformas que compõem o portal. Os administradores podem configurar informações essenciais como o nome, a descrição, o URL base e o caminho para a descoberta de módulos ("modulesPath"). Este registo é

gerido pelo “PlatformManagerController” no backend e permite que o My.IPB saiba como interagir com cada sistema externo.

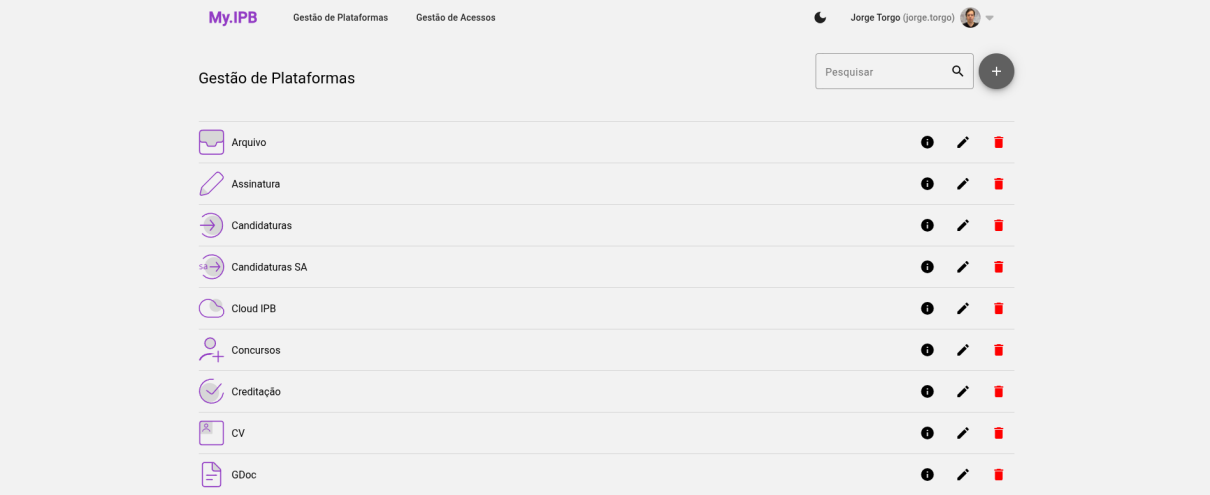


Figura 5.1: Interface de gestão de plataformas no backoffice.

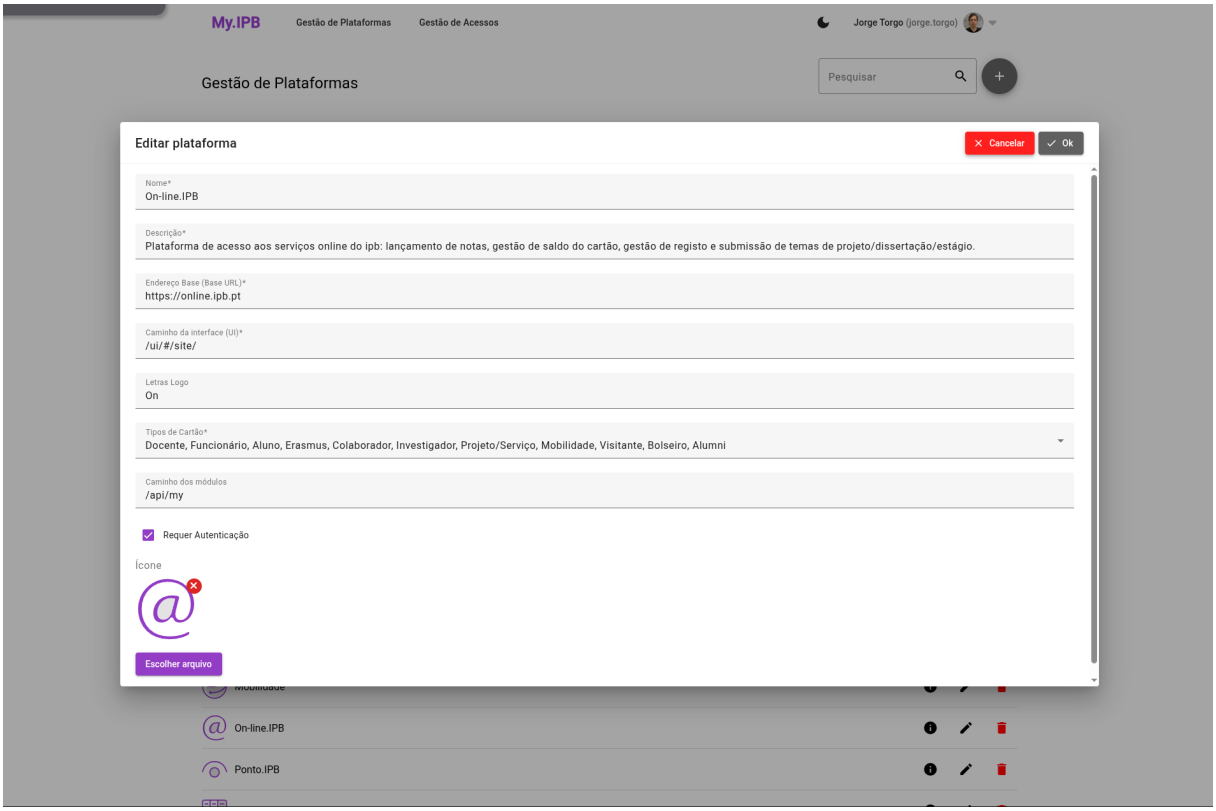


Figura 5.2: Formulário de registo/edição de plataforma.

## 5.2 Interface de Gestão de Permissões

A gestão de permissões para as plataformas externas é realizada através da interface "Role Manager", implementada no componente "`role-manager.component.ts`". Esta interface materializa a arquitetura de gestão distribuída descrita no capítulo anterior, atuando como um cliente central para as APIs de gestão de cada plataforma.

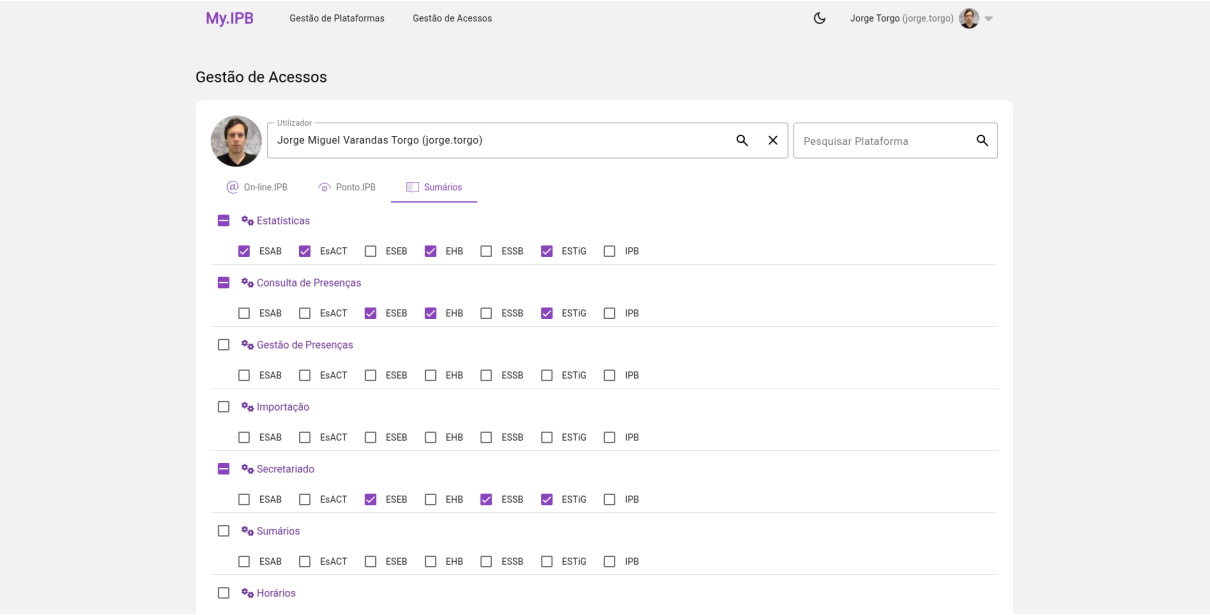


Figura 5.3: Interface de gestão de permissões (Role Manager).

O fluxo de utilização é o seguinte:

1. **Pesquisa de Utilizador:** O administrador começa por pesquisar o utilizador cujas permissões pretende gerir. A interface utiliza o "RfidController" para encontrar o utilizador no sistema.

2. **Descoberta Dinâmica de Módulos e Permissões:** Após a seleção de um utilizador, a interface inicia um processo de descoberta. Para cada plataforma registada que possui um "modulesPath", o frontend realiza dois pedidos:

    - Um pedido para o endpoint "/module-list" da plataforma externa, para obter a lista completa de todos os módulos que essa plataforma oferece.

- Um pedido para o endpoint "/user-modules/{login}" da mesma plataforma, para obter a lista de módulos que o utilizador selecionado tem atualmente atribuídos.

Este processo é realizado em paralelo para todas as plataformas integradas.

3. **Apresentação da Interface de Gestão:** A interface agrega as respostas de todas as plataformas e apresenta ao administrador uma visão consolidada. Para cada plataforma, são listados todos os módulos disponíveis. Os módulos já atribuídos ao utilizador aparecem pré-selecionados. Os módulos podem ainda ser agrupados por "groupId", permitindo uma gestão mais granular (por exemplo, um módulo "Gestão de Presenças" pode ter grupos como "ESTIG" e "ESAB").

4. **Atribuição e Revogação de Permissões:** O administrador pode selecionar ou desmarcar os módulos e os grupos de permissões para o utilizador. Ao submeter as alterações para uma determinada plataforma, o frontend envia um pedido "POST" para o endpoint "/user-modules/{login}" dessa plataforma, contendo a lista atualizada de "ModuleDto" (pares de "moduleId" e "groupId") que o utilizador deverá ter.

Para formalizar este contrato, as plataformas a integrar podem implementar uma interface JAX-RS de referência (Figura 5.4)

```java
package pt.ipb.my.api;

import ...

public interface ModuleEndpointSpecification {  no usages

  @GET
  @Path("/module-list")
  List<ModuleDefinition> getModuleList();

  @GET
  @Path("/user-modules")
  List<ModuleDefinition> getUserModules();

  @GET
  @Path("/user-modules/{login}")
  List<ModuleDefinition> getUserModules(@PathParam("login") String login);

  @POST
  @Path("/user-modules/{login}")
  List<ModuleDefinition> postUserModules(@PathParam("login") String login, List<ModuleDto> moduleList);

}
```

Figura 5.4: Conteúdo de `ModuleEndpointSpecification.java`

Os detalhes das classes usadas na interface de referência são detalhadas no Apêndice B.

Este modelo permite que a gestão de permissões, embora descentralizada na sua implementação, seja centralizada do ponto de vista da experiência do administrador, simplificando drasticamente a gestão de acessos no ecossistema de aplicações do IPB.

## 5.3 O Processo de Integração de Plataformas

Um dos resultados do projeto, conforme planeado, foi a criação de um **Guião de Migração** (Apêndice B), um documento que formaliza o processo técnico para que uma plataforma externa possa ser integrada no ecossistema My.IPB. O processo de **integração técnica** consiste nos seguintes passos:

1. **Registo da Plataforma no Keycloak:** O primeiro passo é configurar a plataforma como um novo "cliente" no Keycloak, para que ela passe a confiar no SSO

para autenticação.

2. **Implementar a API de Gestão de Módulos:** A equipa de desenvolvimento da plataforma externa deve implementar a API de gestão de módulos, que inclui os endpoints para listar todos os módulos, listar os módulos de um utilizador e atualizar os módulos de um utilizador, respeitando o contrato de dados ("ModuleDefinition" e "ModuleDto").

3. **Registo no Backoffice do My.IPB:** Finalmente, um administrador acede ao backoffice do My.IPB e regista a nova plataforma, indicando o seu nome, URL base e o "modulesPath" para a API de gestão. A partir deste momento, a plataforma está integrada e as suas permissões podem ser geridas centralmente.

Para validar a eficácia deste guião e do modelo de integração desacoplado, foram integradas com sucesso três plataformas do IPB como prova de conceito durante o desenvolvimento: **online.ipb.pt**, **sumarios.ipb.pt** e **ponto.ipb.pt**. Esta validação prática demonstrou a viabilidade da arquitetura e do processo de integração definidos.

# Capítulo 6

# Infraestrutura e Automação de CI/CD

Para garantir a qualidade, a consistência e a agilidade no desenvolvimento e na entrega do portal My.IPB, foi implementada uma infraestrutura de integração e entrega contínua (CI/CD). Este capítulo descreve o pipeline de automação, a estratégia de contentorização com Docker, a gestão automatizada da base de dados e a configuração do ambiente de produção.

## 6.1 Pipeline de CI/CD com GitLab

O projeto utiliza o GitLab CI/CD para automatizar todo o ciclo de vida da aplicação. O pipeline, definido nos ficheiros "`.gitlab-ci.yml`", é acionado a cada alteração no repositório e é composto pelos seguintes estágios:

- **Image:** Neste estágio, é construída uma imagem Docker customizada para o ambiente de build do frontend. Esta imagem, definida em "`Dockerfile-build-env`", contém todas as dependências necessárias (Node.js, Angular CLI, etc.), garantindo que o ambiente de compilação seja consistente e reprodutível.
- **Build:** Ocorre a compilação do backend (Spring Boot) e do frontend (Angular). O

backend é compilado com o Gradle, gerando um pacote de instalação. O frontend é compilado com o Angular CLI e a versão da aplicação é atualizada dinamicamente com base na tag do Git. Os artefactos de build de ambos são armazenados para o próximo estágio.

- **Package:** Os artefactos do estágio de build são empacotados numa única imagem Docker de produção, que é publicada no registry interno do IPB (registry.cdev.ipb.pt). O "Dockerfile" define esta imagem, que utiliza uma base OpenJDK 21 e copia a aplicação backend e os ficheiros estáticos do frontend. A imagem resultante é etiquetada com a tag do Git e com a etiqueta "production", e é enviada para o registo de contentores privado do projeto.

- **Deploy:** Este estágio é acionado apenas quando uma nova tag é criada no repositório. O pipeline conecta-se ao servidor de produção, pára a versão antiga da aplicação, descarrega a nova imagem do registo e inicia um novo contentor com o "`docker-compose-production.yml`".

## 6.2 Gestão Automatizada do Ciclo de Vida da Base de Dados

A automação do projeto estende-se para além do código da aplicação, abrangendo todo o ciclo de vida da base de dados. Esta abordagem garante que a estrutura da base de dados e o código que a acede evoluem em perfeita sincronia.

### 6.2.1 Controlo de Versões da Base de Dados com Liquibase

O projeto utiliza o **Liquibase**[7] para gerir as alterações ao esquema da base de dados. No módulo "my-schema", todas as alterações (criação de tabelas, adição de colunas, etc.) são definidas em ficheiros XML, que funcionam como um histórico de versões da base de dados.

Esta prática oferece duas grandes vantagens:

- **Consistência entre Ambientes:** Garante que a estrutura da base de dados é a mesma em todos os ambientes, desde o desenvolvimento até à produção.
- **Implantação Automatizada:** O Spring Boot integra-se com o Liquibase e, no arranque da aplicação, verifica se o schema da base de dados está atualizado. Se existirem alterações pendentes, o Liquibase aplica-as automaticamente antes de a aplicação começar a processar pedidos. Isto elimina a necessidade de executar scripts SQL manualmente durante a implantação.

### 6.2.2 Geração de Código com QueryDSL

Para a camada de acesso a dados, o projeto adota uma abordagem de geração de código através dos módulos "my-dsl-*" que utilizam o **QueryDSL**. Em vez de escrever consultas SQL como strings, o que é propenso a erros, o QueryDSL gera classes Java ("Q-types") que espelham o esquema da base de dados, muito à semelhança de um Object-Relational Mapping (ORM).

Isto permite que as consultas sejam escritas em Java, de forma **type-safe**. Se uma tabela ou coluna for renomeada no esquema do Liquibase, a regeneração das classes do QueryDSL fará com que o código que a referência deixe de compilar, detetando o erro em tempo de desenvolvimento, e não em produção. Esta sinergia entre o Liquibase e o QueryDSL cria um ciclo de desenvolvimento robusto e automatizado, onde o *schema* da base de dados e o código da aplicação estão permanentemente alinhados.

## 6.3 Contentorização com Docker

A aplicação é totalmente containerizada. A configuração do ambiente de produção é definida no ficheiro "`docker-compose-production.yml`":

- **Imagem:** O serviço descarrega do *registry* interno (pull) e utiliza (run) a imagem

"my-ipb-server:production", garantindo que a versão em execução é sempre a mais recentemente etiquetada como tal.

- **Rede:** O contentor é ligado a uma rede docker com o nome backend, permitindo a comunicação com outros serviços da infraestrutura do IPB de forma segura.
- **Volumes:** São utilizados volumes para tornar os dados (ex.: logs) da aplicação persistentes.
- **Reinício Automático:** A política "restart: on-failure" garante que, em caso de erro, o contentor seja reiniciado automaticamente, aumentando a resiliência do serviço.

Esta abordagem de automação e contentorização não só acelera o ciclo de desenvolvimento, como também garante que o processo de entrega de novas versões seja robusto, seguro e livre de intervenção manual, representando uma prática de engenharia de software moderna e essencial para a sustentabilidade do projeto. Adicionalmente, a migração do serviço para uma nova infraestrutura implica apenas a sincronização da pasta contendo a configuração do *docker compose*.

# Capítulo 7

# Conclusões e Trabalho Futuro

Este capítulo final apresenta uma síntese do trabalho desenvolvido, avalia o cumprimento dos objetivos propostos à luz da arquitetura implementada e delineia possíveis direções para a evolução futura do portal My.IPB.

## 7.1 Conclusões

O desenvolvimento do portal My.IPB respondeu com sucesso ao desafio de unificar o acesso ao ecossistema digital do IPB. A criação de um ponto de acesso centralizado, aliada à implementação de uma solução de SSO, caminhou no sentido de eliminar o processo repetitivo de autenticação em múltiplas plataformas, resultando numa experiência de utilização mais fluida e eficiente.

Os principais objetivos do projeto foram alcançados através de uma arquitetura desacoplada e abrangente:

- Foi implementada uma **solução de SSO robusta**, que unifica a experiência de autenticação no ecossistema digital do IPB.
- A solução de SSO foi **estendida com sucesso à aplicação móvel On-line.IPB**, partilhando a sessão do utilizador no dispositivo para criar uma experiência de login transparente entre o portal web e a aplicação móvel.

- Foi desenvolvido um **portal com um dashboard personalizável**, que agrega dinamicamente os módulos a que o utilizador tem acesso em cada plataforma.
- Foi criado um **módulo de backoffice** que funciona como um centro de orquestração para o registo de plataformas e a gestão de permissões de forma federada.
- O processo de **integração de plataformas** foi validado. Em vez de uma migração de dados, foi estabelecido um modelo de integração técnica, baseado em contratos de API ("ModuleDefinition"), que foi comprovado com a integração de três sistemas existentes.

A arquitetura da solução, baseada em Spring Boot e Angular, provou ser uma escolha acertada, resultando num sistema modular e escalável que promove a autonomia das plataformas ao mesmo tempo que centraliza e simplifica o acesso do utilizador. O projeto representa um passo significativo na modernização dos serviços digitais do IPB.

## 7.2 Trabalho Futuro

O portal My.IPB estabelece uma fundação sólida para futuras evoluções. A arquitetura de integração dinâmica abre um leque de possibilidades para além do que foi implementado:

- **Expandir o Ecossistema:** Continuar o processo de integração técnica, trazendo mais plataformas do IPB para o ecossistema SSO.
- **Dashboard Inteligente:** O dashboard pode evoluir para se tornar um centro de notificações, agregando alertas de diferentes plataformas num único local.

O My.IPB não é apenas um portal, mas uma plataforma de integração que pode continuar a crescer, promovendo uma experiência académica cada vez mais conectada e integrada no Instituto Politécnico de Bragança.

# Bibliografia


[1] I. Okta, *Okta Identity Platform*, 2024. acedido em 16 de out. de 2024. URL: https://www.okta.com/.

[2] M. Corporation, *Microsoft Entra ID*, 2024. acedido em 16 de out. de 2024. URL: https://www.microsoft.com/en-us/security/business/identity-access/microsoft-entra-id.

[3] WSO2, *WSO2 Identity Server*, 2024. acedido em 16 de out. de 2024. URL: https://wso2.com/identity-server/.

[4] R. Hat, *Keycloak: Open Source Identity and Access Management*, 2024. acedido em 16 de out. de 2024. URL: https://www.keycloak.org/.

[5] V. Sakimura, J. Bradley, M. Jones, B. de Medeiros e C. Mortimore, "OpenID Connect Core 1.0," OpenID Foundation, rel. téc., nov. de 2014. URL: https://openid.net/specs/openid-connect-core-1_0.html.

[6] D. Hardt, "The OAuth 2.0 Authorization Framework," Internet Engineering Task Force (IETF), RFC 6749, out. de 2012. URL: https://tools.ietf.org/html/rfc6749.

[7] Liquibase, *Liquibase: Database Schema Change Management*, 2024. acedido em 5 de nov. de 2024. URL: https://www.liquibase.org/.

# Apêndice A

# Proposta Original do Projeto

INSTITUTO POLITÉCNICO DE BRAGANÇA
Escola Superior de Tecnologia e Gestão

# MESTRADO EM INFORMÁTICA

*Master of Informatics*

Unidade Curricular de "Dissertação/Projeto/Estágio"

*Course unit "Thesis/Project/Intership"*

***Work proposal***

## Proposta de tema

☐ Dissertação *Thesis* ☑ Projeto *Project* ☐ Estágio *Internship*

Título *Title*

**Conceção e implementação do portal myIPB**

Palavras-chave *Keywords*

SSO, OAuth2, myIPB

| Orientador IPB *IPB Supervisor* | *email* |
|---|---|
| Albano Alves | albano@ipb.pt |

| Co-orientador IPB *IPB Co-supervisor* | *email* |
|---|---|
| | |

| Co-orientador 1 *Co-supervisor 1* | *email* |
|---|---|
| | |

| Instituição do Co-orientador 1 *Institution of co-supervisor 1* | País *Country* |
|---|---|
| | |

| Co-Orientador 2 *Co-supervisor 2* | *email* |
|---|---|
| | |

| Instituição do Co-orientador externo 2 *Institution of co-supervisor 2* | *Country* |
|---|---|
| | |

| Aluno/*Student* nº/ *number* | Nome *name* | email *Student email* |
|---|---|---|
| 41113 | Jorge Miguel Varandas Torgo | a41113@alunos.ipb.pt |

Objetivos *Goals*

Desenvolver um portal para sistematização do acesso às plataformas informáticas do IPB.
A solução deverá contemplar:
- funcionalidade de Single Sign-On (SSO);
- facilidade de parametrização e configuração por cada utilizador;
- acesso direto a módulos específicos das plataformas;
- integração com a app IPBonline;
- módulo de backoffice, para gestão integrada de permissões e acessos.



v2024

## Descrição *Description*

O projeto decorrerá na Divisão de Informática do IPB, com integração na equipa de desenvolvimento.
O portal myIPB será disponibilizado a toda a comunidade académica do IPB em my.ipb.pt
Os utilizadores, de acordo com o seu perfil e depois de se autenticarem, terão acesso às plataformas informáticas do IPB, podendo parametrizar o layout do seu dashboard.
O administrador do portal poderá fazer a gestão de permissões e acessos das várias plataformas do IPB, de forma centralizada.
O interface com as várias plataformas utilizadas no IPB será, numa primeira fase, especificado e documentado num guião; serão depois selecionadas algumas plataformas para prova de conceito e implementação final.
As tecnologias a utilizar no desenvolvimento do projeto serão:
- Spring Boot
- Angular
- Oracle Database
- OAuth2

## Metodologia / Plano de trabalhos *Methodology/ Work plan*

1. Levantamento e caracterização do portfólio de aplicações IPB;
2. Estudo de soluções SSO, com especial ênfase na integração com as soluções e plataformas existentes no IPB;
3. Desenho do portal myIPB;
4. Implementação de uma solução SSO;
5. Criação de um guião para migração das plataformas do IPB para a nova solução de SSO;
6. Concretização da migração de 3 plataformas;
7. Implementação de funcionalidades de parametrização e configuração;
8. Desenvolvimento do módulo de backoffice.

## Pré-requisitos *Prerequisites*

-Conhecimentos de programação em Java (Spring Boot) e Angular.
-Noções de bases de dados relacionais (Oracle ou similares).
-Familiaridade com protocolos de autenticação (OAuth2, OpenID Connect, SAML).
-Experiência em gestão de permissões e acessos.
-Noções de integração de sistemas.

## Recursos necessários *Resources needed*

-Infraestrutura de desenvolvimento e testes disponibilizada pela equipa de desenvolvimento do IPB.
-Acesso controlado às plataformas institucionais do IPB que serão integradas no portal.
-Conta de programador com permissões para configuração de autenticação, perfis de utilizador e gestão de acessos.
-Documentação técnica relativa às plataformas a integrar.
-Apoio técnico e acompanhamento por parte da equipa de desenvolvimento, para validação e suporte nas fases de integração.

## Data *Date*

16/10/2024

# Apêndice B

# Guião de Migração

# Guião de Migração: De uma Aplicação Convencional para SSO com Keycloak

Este guião serve como um manual de migração conceitual para transformar uma aplicação Spring Boot e Angular convencional numa plataforma segura com *Single Sign-On (SSO)*. O objetivo é explicar os passos e os componentes necessários.

O exemplo neste documento foca-se no *backend* Spring Boot com *frontend* Angular, mas é facilmente adaptável para outras arquiteturas e/ou *frameworks*.

**Pré-requisitos:**

- Um projeto Spring Boot existente com uma base de dados.
- Um projeto Angular existente.
- Acesso a uma instância do Keycloak.

## Tabela de Conteúdos

# 1. Visão Geral da Arquitetura Final

A arquitetura de segurança resultante é composta por três componentes principais:

- **Keycloak:** O provedor de identidade (IdP) que centraliza a gestão de utilizadores e autenticação.
- **Frontend (Angular):** Uma SPA que delega o login ao Keycloak. Após a autenticação, gere o estado do utilizador e envia o *access token* em cada pedido à API.
- **Backend (Spring Boot):** Atua como um **Resource Server**. A sua responsabilidade é proteger os endpoints da API validando o *access token* (JWT) e enriquecendo o contexto de segurança com perfis e permissões da sua aplicação.

O pilar desta arquitetura é o **modelo de perfis híbrido**, onde a autorização de um utilizador é decidida por cada plataforma individualmente, enquanto a autenticação está a cargo do Keycloak.

## 2. Preparação e Configuração do Keycloak

Assume-se uma instância do Keycloak em `https://sso.ipb.pt` com um realm `ipb`.

### 2.1. Clientes Necessários

- **Cliente Frontend:**
  - **Tipo:** Público (`Client authentication: Off`).
  - **Configuração:** Requer `Valid redirect URIs` e `Web origins` configurados para os endereços da sua SPA (ex: `http://localhost:4200`).
- **Cliente Backend:**
  - **Tipo:** Confidencial (`Client authentication: On`) ou Público.
  - **Objetivo:** Identifica a sua API (o Resource Server) dentro do realm. O `clientId` será a "audiência" (`aud`) nos tokens JWT.

## 3. Migração do Backend (Spring Boot)

### Passo 3.1: O Objeto Principal do Utilizador

O seu backend precisa de uma classe para representar o utilizador autenticado. É uma boa prática que esta classe implemente a interface `UserDetails` do Spring Security para se integrar perfeitamente com o `SecurityContext`.

- **Ação:** Crie ou adapte uma classe (ex: `YourUserPrincipal`) que contenha os dados do utilizador (ID, nome, email, etc.) e uma lista para os seus perfis de autorização.

```
// Exemplo conceitual de um objeto de utilizador
public class YourUserPrincipal implements UserDetails {
  private String username;
  private String email;
  private List<String> authorities;

  // ... Construtores, getters, setters ...

  @Override
  public Collection<? extends GrantedAuthority> getAuthorities() {
    // Transforma a lista de strings de perfis em objetos GrantedAuthority
    return authorities.stream()
        .map(SimpleGrantedAuthority::new)
        .collect(Collectors.toList());
  }
  // ... Implementar outros métodos de UserDetails ...
}
```

### Passo 3.2: Dependências e Configuração

1. **Adicione a dependência** do Resource Server ao seu `build.gradle` ou `pom.xml`.

   ```
   // Exemplo para Gradle
   implementation 'org.springframework.boot:spring-boot-starter-oauth2-
   resource-server'
   ```

2. **Configure o `application.properties`** para que o Spring possa validar os tokens JWT emitidos pelo seu realm do Keycloak.

   ```
   spring.security.oauth2.resourceserver.jwt.issuer-
   uri=https://sso.ipb.pt/realms/ipb
   spring.security.oauth2.resourceserver.jwt.jwk-set-
   uri=https://sso.ipb.pt/realms/ipb/protocol/openid-connect/certs
   ```

### Passo 3.3: Implementar o Modelo de Perfis Híbrido

Esta é a lógica central que combina os perfis do Keycloak com os perfis da sua base de dados.

1. **Crie um Repositório de Segurança:** Precisará de um método para obter os perfis de um utilizador a partir da sua base de dados.
    - **Ação:** Crie uma interface (ex: `YourSecurityRepository`) e implemente-a para executar uma query na sua base de dados.

```
// Exemplo de interface de repositório
public interface YourSecurityRepository {
    List<String> findRolesByUsername(String username);
}
```

2. **Crie um Serviço de Perfis:** Um serviço simples para expor a lógica do repositório.

```
// Exemplo de serviço de perfis
@Service
public class YourRoleService {
    private final YourSecurityRepository securityRepository;
    // ... construtor ...
    public List<String> getRolesForUser(String username) {
        return securityRepository.findRolesByUsername(username);
    }
}
```

3. **Crie um Conversor JWT Customizado:** Este componente do Spring Security irá interceptar cada token JWT, extrair os perfis do Keycloak e adicionar os perfis da sua aplicação.

```
// Conversor JWT para o modelo híbrido
@Component
public class YourJwtAuthenticationConverter implements Converter<Jwt,
AbstractAuthenticationToken> {
    private final YourRoleService roleService;
    // ... construtor ...

    @Override
    public AbstractAuthenticationToken convert(@NotNull Jwt jwt) {
        // 1. Extrair perfis do token (ex: roles do realm)
        List<String> combinedRoles = new ArrayList<>
(KeycloakUtil.extractRoleClaims(jwt));

        // 2. Obter o username do token
        String username = jwt.getClaimAsString("preferred_username");

        // 3. Adicionar perfis da sua base de dados
        combinedRoles.addAll(roleService.getRolesForUser(username));
```

```
        // 4. Criar o objeto principal com os perfis combinados
        YourUserPrincipal principal = new YourUserPrincipal(username,
jwt.getClaimAsString("email"));
        principal.setAuthorities(combinedRoles);

        return new UsernamePasswordAuthenticationToken(principal,
null, principal.getAuthorities());
    }
}
```

### Passo 3.4: Configuração de Segurança Principal

Finalmente, configure o `SecurityFilterChain` do Spring para usar o seu conversor JWT customizado.

```
@Configuration
@EnableWebSecurity
public class YourSecurityConfiguration {

  @Bean
  public SecurityFilterChain securityFilterChain(HttpSecurity http,
YourJwtAuthenticationConverter converter) throws Exception {
    return http
        .cors(Customizer.withDefaults())
        .csrf(AbstractHttpConfigurer::disable)
        // Proteger todos os endpoints por defeito. Personalize conforme
necessário.
        .authorizeHttpRequests(r -> r.anyRequest().authenticated())
        // Ativar o Resource Server e ligar o nosso conversor
        .oauth2ResourceServer(r -> r.jwt(j ->
j.jwtAuthenticationConverter(converter)))
        // Tornar a sessão stateless, pois cada pedido tem um token
        .sessionManagement(s ->
s.sessionCreationPolicy(SessionCreationPolicy.STATELESS))
        .build();
  }
}
```

## 4. Migração do Frontend (Angular)

### Passo 4.1: Serviços Fundamentais

Para uma arquitetura limpa, comece por criar dois serviços reutilizáveis.

1. **Crie um `EventBusService`:** Um serviço simples baseado em RxJS para publicar e subscrever eventos em toda a aplicação.

```
// event-bus.service.ts
@Injectable({ providedIn: 'root' })
export class EventBusService {
    private subjects: { [key: string]: ReplaySubject<any> } = {};

    publish(topic: string, data: any) {
        this.subject(topic).next(data);
    }

    on(topic: string): Observable<any> {
        return this.subject(topic).asObservable();
    }

    private subject(topic: string): ReplaySubject<any> {
        return this.subjects[topic] || (this.subjects[topic] = new
ReplaySubject<any>(1));
    }
}
```

2. **Crie um `ApiClientService`:** Um serviço para comunicar com o seu backend. Ele encapsula o `HttpClient` do Angular.

```
// api-client.service.ts
@Injectable({ providedIn: 'root' })
export class ApiClientService {
    private http = inject(HttpClient);
    private apiUrl = 'http://localhost:8080/api'; // URL base da sua
API

    getPermissions(): Observable<string[]> {
        return this.http.get<string[]>
(`${this.apiUrl}/auth/permissions`);
    }

    // Adicione outros métodos para chamar o seu backend
}
```

### Passo 4.2: Configuração do Keycloak e do Interceptor

1. **Instale `keycloak-angular`:** `npm install keycloak-angular keycloak-js`
2. **Configure os providers na sua `app.config.ts`:**
    - Use `provideHttpClient` com `withInterceptors` para incluir o `includeBearerTokenInterceptor`. **Este é o passo crítico** que anexa automaticamente o token de acesso a todos os pedidos para a sua API.
    - Use `provideKeycloak` para inicializar o serviço do Keycloak.

```
// app.config.ts
import { ApplicationConfig } from '@angular/core';
import { provideHttpClient, withInterceptors } from
"@angular/common/http";
import { provideKeycloak, includeBearerTokenInterceptor } from
"keycloak-angular";

export const appConfig: ApplicationConfig = {
  providers: [
    provideHttpClient(withInterceptors([
      // Este interceptor anexa o token 'Authorization: Bearer ...'
      includeBearerTokenInterceptor
    ])),
    provideKeycloak({
      config: {
        url: 'https://sso.ipb.pt',
        realm: 'ipb',
        clientId: 'your-frontend-client'
      },
      initOptions: {
        onLoad: "check-sso",
        silentCheckSsoRedirectUri: window.location.origin +
'/assets/silent-check-sso.html'
      }
    })
  ]
};
```

### Passo 4.2.1: O Ficheiro silent-check-sso.html

A opção `onLoad: "check-sso"` instrui o Keycloak a verificar silenciosamente se o utilizador já tem uma sessão ativa no IdP, sem forçar um redirecionamento para a página de login. Este processo ocorre num `iframe` invisível que carrega o URL especificado em `silentCheckSsoRedirectUri`.

É crucial que este ficheiro exista e contenha o seguinte código. Ele atua como uma ponte, comunicando o resultado da verificação de volta para a sua aplicação principal.

- **Ação:** Crie o ficheiro `src/assets/silent-check-sso.html` com o seguinte conteúdo:

```
<html>
  <body>
```

```
    <script>
      parent.postMessage(location.href, location.origin);
    </script>
  </body>
</html>
```

### Passo 4.3: Gestão de Eventos e Sessão

1. **Crie um `KeycloakEventHandler`:** Este serviço ouve os eventos da biblioteca `keycloak-angular` e publica-os no seu `EventBusService`.

```
// keycloak-event-handler.service.ts
@Injectable({ providedIn: 'root' })
export class KeycloakEventHandler {
    #keycloak = inject(KeycloakService);
    #eventBus = inject(EventBusService);

    constructor() {
        this.#keycloak.keycloakEvents$.subscribe(event => {
            if (event.type === KeycloakEventType.OnAuthSuccess) {
                this.#keycloak.loadUserProfile().then(profile => {
                    this.#eventBus.publish('PROFILE_LOADED', profile);
                });
            }
        });
    }
}
```

2. **Crie o `AuthenticationService`:** Este serviço central gere o estado do utilizador. Ele ouve o evento `PROFILE_LOADED` e reage.

```
// authentication.service.ts
@Injectable({ providedIn: 'root' })
export class AuthenticationService {
    #user = new BehaviorSubject<YourUserPrincipal | null>(null);
    public user$ = this.#user.asObservable();

    constructor(private eventBus: EventBusService, private apiClient:
ApiClientService) {
        this.eventBus.on('PROFILE_LOADED').subscribe(profile => {
            this.#user.next(profile as YourUserPrincipal);
            this.apiClient.getPermissions().subscribe(permissions => {
                // ... armazene as permissões onde for necessário ...
            });
        });
    }
}
```

# 5. Integração da Plataforma com o My.IPB

Após migrar a sua aplicação para o SSO do Keycloak, o passo final é integrá-la ao portal My.IPB para permitir a gestão centralizada de permissões. Para isso, a sua aplicação deve expor uma **API de Gestão de Módulos** que o My.IPB possa consumir.

## 5.1. Contrato da API: Modelos de Dados

A comunicação entre o My.IPB e a sua API de módulos baseia-se em dois modelos de dados principais.

**ModuleDefinition**

Representa uma permissão ou funcionalidade (um "módulo") que pode ser atribuída a um utilizador na sua plataforma. **A estrutura deve ser idêntica à classe `ModuleDefinition` do `my-public-api`.**

```
package pt.ipb.my.api;

import org.jetbrains.annotations.Nullable;

public class ModuleDefinition {

  private String moduleId;
  private String name;
  private String groupId;
  private String groupName;
  private String url;
  private String iconStyle;

  // ...

}
```

- **Estrutura (JSON):**

```
{
  "moduleId": "string",
  "name": "string",
  "groupId": "string",
  "groupName": "string",
  "url": "string",
  "iconStyle": "string"
}
```

- **Campos:**
  - `moduleId`: Identificador único do módulo (ex: "inqueritos_admin").
  - `name`: Nome legível para o módulo (ex: "Administração de Inquéritos").

- `groupId`: ID do grupo ao qual o módulo pertence.
- `groupName`: Nome do grupo ao qual o módulo pertence.
- `url`: URL para aceder ao módulo no frontend da sua aplicação.
- `iconStyle`: Classe de CSS para o ícone do módulo (ex: "fas fa-poll").

**ModuleDto**

O ModuleDto define o que deve ser enviado para a atualização dos módulos, onde é enviada apenas a lista dos módulos que devem estar ativos para o utilizador. Os módulos que não forem enviados serão desativados automaticamente.

- **Estrutura (JSON):**

```
{
  "moduleId": "string",
  "groupId": "string"
}
```

- **Campos:**
  - `moduleId`: Identificador único do módulo.
  - `groupId`: ID do grupo ao qual o módulo pertence.(Caso Exista)

**ModuleEndpointSpecification**

A especificação formal dos endpoints, incluindo os seus caminhos, métodos HTTP e assinaturas, é definida na interface ModuleEndpointSpecification.java do my-public-api. Esta interface serve como o contrato programático que a sua aplicação deve implementar.

```
package pt.ipb.my.api;

import jakarta.ws.rs.GET;
import jakarta.ws.rs.POST;
import jakarta.ws.rs.Path;
import jakarta.ws.rs.PathParam;

import java.util.List;

public interface ModuleEndpointSpecification {

  @GET
  @Path("/module-list")
  List<ModuleDefinition> getModuleList();

  @GET
  @Path("/user-modules")
  List<ModuleDefinition> getUserModules();

  @GET
```

```
  @Path("/user-modules/{login}")
  List<ModuleDefinition> getUserModules(@PathParam("login") String login);

  @POST
  @Path("/user-modules/{login}")
  List<ModuleDefinition> postUserModules(@PathParam("login") String login,
List<ModuleDto> moduleList);

}
```

## 5.2. Endpoints da API

A sua aplicação deve implementar os seguintes endpoints num caminho base que será configurado no backoffice do My.IPB ( referido como `modulesPath`).

**1. Listar todos os Módulos da Plataforma**

Este endpoint permite ao My.IPB saber quais são todas as permissões disponíveis na sua aplicação.

- **Endpoint:** `GET /<modulesPath>/module-list`
- **Resposta:** `200 OK`
- **Corpo da Resposta:** `ModuleDefinition[]` - Uma lista com a definição de todos os módulos.

**2. Listar Módulos de um Utilizador (por Administrador)**

Fornece ao My.IPB os módulos que um utilizador específico tem acesso na sua plataforma.

- **Endpoint:** `GET /<modulesPath>/user-modules/{login}`
- **Parâmetro de URL:**
  - `login`: O login do utilizador (ex: `preferred_username` do token JWT).
- **Resposta:** `200 OK`
- **Corpo da Resposta:** `ModuleDefinition[]` - A lista de módulos que o utilizador tem acesso.

**3. Listar Módulos do Utilizador Autenticado**

Fornece os módulos que o próprio utilizador autenticado tem acesso. Este endpoint é tipicamente consumido pelo frontend da sua aplicação.

- **Endpoint:** `GET /<modulesPath>/user-modules`
- **Resposta:** `200 OK`
- **Corpo da Resposta:** `ModuleDefinition[]` - A lista de módulos que o utilizador autenticado tem acesso.

**4. Atualizar Módulos de um Utilizador**

O My.IPB usará este endpoint para atribuir ou remover permissões de um utilizador na sua plataforma.

- **Endpoint:** `POST /<modulesPath>/user-modules/{login}`
- **Parâmetro de URL:**
  - `login`: O login do utilizador a ser atualizado.

- **Corpo do Pedido:** `ModuleDto[]` - Lista de `ModuleDto` de cada módulo que o utilizador deve ter acesso. A sua aplicação deve percorrer esta lista e aplicar as alterações na sua base de dados.
- **Resposta:**
  - `200 OK` se a atualização for bem-sucedida. O corpo da resposta conterá a lista atualizada de módulos do utilizador (`ModuleDefinition[]`).
  - `400 Bad Request` se os dados forem inválidos.
  - `500 Internal Server Error` se ocorrer um erro ao persistir as alterações.

### 5.3. Registo no Backoffice do My.IPB

Com a API implementada e a sua aplicação a correr, o passo final é o registo administrativo:

1. Um administrador acede ao backoffice do My.IPB.
2. Regista a sua aplicação fornecendo:
   - **Nome:** Um nome amigável (ex: "Plataforma de Inquéritos").
   - **URL Base:** O endereço principal da sua aplicação (ex: `https://inqueritos.ipb.pt`).
   - **Caminho dos Módulos (`modulesPath`):** O caminho base onde a sua API de Gestão de Módulos está a ser servida ( ex: `/api/modules`).

A partir deste momento, o My.IPB poderá gerir as permissões da sua plataforma de forma centralizada.

## 6. Conclusão

Ao seguir estes passos conceituais, pode migrar a sua plataforma para um sistema de SSO robusto e completo:

- **Autenticação Centralizada:** O Keycloak trata do login.
- **Comunicação Segura:** O `includeBearerTokenInterceptor` garante que o seu frontend envia tokens de acesso válidos.
- **Backend Seguro:** O seu backend valida os tokens e enriquece a segurança com a sua lógica de negócio.
- **Frontend Reativo:** O seu frontend reage ao estado de autenticação e gere a sessão do utilizador de forma limpa e desacoplada.

Esta abordagem fornece um roteiro claro, permitindo-lhe preencher os detalhes de implementação de acordo com as especificidades da sua plataforma.