Document Workflow Platform

Miguel Henriques Couto de Almeida
David Paulo Santos Robalo Costa
Ricardo Duarte Cardoso Bernardino

Orientadores Pedro Miguel Florindo Miguens Matutino
Artur Jorge Ferreira

Relatório final realizado no âmbito de Projeto e Seminário,
do curso de licenciatura em Engenharia Informática e de Computadores
Semestre de Verão 2022/2023

Julho de 2023

Instituto Superior de Engenharia de Lisboa
Licenciatura em Engenharia Informática e de Computadores

Document Workflow Platform

47249 Miguel Henriques Couto de Almeida
47283 Ricardo Duarte Cardoso Bernardino
45935 David Paulo Santos Robalo Costa

Orientadores: Pedro Miguel Florindo Miguens Matutino
Artur Jorge Ferreira

Relatório realizado no âmbito de Projeto e Seminário,
do curso de licenciatura em Engenharia Informática e de Computadores
Semestre de Verão 2022/2023
Julho de 2023

Resumo

Os processos de geração, verificação, aprovação e publicação de documentos em organizações hierárquicas são vitais para o funcionamento das mesmas. No entanto, estes processos são consumidores de recursos materiais e humanos, ao longo de toda a cadeia, pelos diferentes órgãos da instituição. Torna-se necessário que todas as etapas dos processos sejam bem definidas e bem calendarizadas. Em simultâneo, em todos os momentos do processo os vários intervenientes devem conseguir aferir sobre o estado do mesmo e identificar os restantes intervenientes. Atualmente, em instituições como o ISEL, estes procedimentos são realizados através de e-mails ou até mesmo pela passagem física destes documentos, tornando o procedimento de processos e das suas etapas respetivas moroso, sendo importante automatiza-los e otimizá-los. Com o intuito de sistematizar uma solução para os processos de aprovação de documentos e fornecer uma solução à própria instituição, concebeu-se e desenvolveu-se uma plataforma de *workflow* de documentos, designada por *DWP (Document Workflow Platform)*. A infraestrutura desta plataforma é composta por uma base de dados relacional, acedida e manipulada por uma *Web Application Programming Interface (Web API)*, e por uma aplicação Web que fornece uma interface a utilizadores da própria.

Palavras-chave: Aprovação de documentos; Automatização; Definição de processos; Organizações hierárquicas; Processos de geração de documentos;

Abstract

The processes of document generation, verification, approval, and publication in hierarchical organizations are vital for their functioning. However, these processes consume material and human resources throughout the entire chain, involving different departments within the institution. It is necessary for all stages of the processes to be well-defined and properly scheduled. Simultaneously, at every stage of the process, the various participants must be able to assess its status and identify the other participants. Currently, in institutions like ISEL, these procedures are carried out through email exchanges or even by physically passing these documents, making the process and its respective stages time-consuming. It is important to automate and optimize these procedures.

In order to address this problem and provide a solution to the institution itself, a document *workflow* platform, or *DWP (Document Workflow Platform)*, has been conceived and developed. The infrastructure of this platform consists of a relational database accessed and manipulated through a *Web Application Programming Interface (Web API)*, and a web application that provides an interface to the users themselves.

Keywords: Document Approval; Automation; Process Definition; Hierarchical Organizations; Document Generation;

Índice

Lista de Figuras

Siglas e Acrónimos

ACID Atomicidade, Consistência, Isolamento, Durabilidade. 18

API Application Programming Interface. 5, 6, 9, 13–16, 21, 37

DAO Data Access Object. 15

DWP Document Workflow Platform. 13

EA Modelo Entidade-Associação. 9, 12

HTML Hypertext Markup Language. 21, 27

HTTP Hypertext Transfer Protocol. 14–16

JDBC Java Database Connectivity. 15

JSON JavaScript Object Notation. 10, 12

RBAC Role-Based Access Control. 19, 20

Capítulo 1

Introdução

Todos as instituições, de pequeno, médio ou grande dimensão lidam diariamente com problemas relacionados com a geração, verificação, aprovação e publicação de documentos, sendo todo este processo, na grande maioria dos cenários, uma componente vital para o seu bom funcionamento e sucesso.

Este processo está também sujeito às particularidades de cada instituição como a sua hierarquia institucional, os papéis que os funcionários assumem, a comunicação entre departamentos e a própria forma como são aprovados os documentos e notificados os intervenientes do mesmo processo.

Da necessidade de adaptação destas instituições e destes processos antigos à "Transição Digital", que decorrerá durante esta década, que nasceu a ideia para este projeto.

Foi ao olhar para o problema, que o corpo docente do ISEL estava a enfrentar há alguns anos quando confrontado com todo o sistema de aprovação de processos dentro da instituição, que foram também saindo os principais pontos críticos associados à aprovação de documentos dedicados a processos na mesma instituição que enumeram-se como:

- o longo tempo de demora até aprovar documentos, passando por todas as etapas e intervenientes, é excessivo dentro dos processos do ISEL;
- os problemas de comunicação, entre os diferentes intervenientes, que surgiam no decorrer do próprio processo devido à falta de uma metodologia e métodos sistemáticos de comunicação;
- a demora no envio de comentários ou sugestões sobre um determinado processo como também a não notificação dos docentes e consequente incapacidade de aferirem se fora ou não aprovado um dado processo, tendo este sido terminado;

O grupo consultou a documentação relativa à hierarquia do ISEL e a descrição do funcionamento de alguns dos processos para que fossem analisados os seus detalhes e testados os cenários onde a aplicação realizada neste projeto pudesse ser utilizada para aumentar a sua eficiência.

Analisando a documentação constatou-se que todo o processo de aprovação de documentos, como também os próprios intervenientes dos mais diversos processos estava bem documentada, tendo sido esse o ponto de partida para o do desenho da solução.

O conjugar de toda esta informação, dispersa em ficheiros guardados na instituição, dentro de uma aplicação que facilite a submissão, comunicação e aprovação destes processos, tão bem descritos no papel, mudaria o funcionamento desta instituição podendo impactar não só na economia de diversos recursos, como também a melhor organização e gestão do tempo dos intervenientes.

Na estrada que nos leva à "Transição Digital", a adoção da aplicação preconizada por este projeto, colocaria a instituição numa via de aceleração, na transição digital necessária por todas as instituições que ambicionem manter-se relevantes nesta nova década.

1.1 Objetivos

Como objetivos deste trabalho pretende-se mitigar os problemas enumerados anteriormente, começando por:

- facilitar a migração de todos os processos, etapas, intervenientes e documentos para uma plataforma digital;
- disponibilizar aos utilizadores criação de novos processos, participar, aprovando ou não, os documentos nesses novos processos;
- escrever comentários nas várias etapas de um processo;
- notificar os participantes sobre o estado do processo e quando terminados, aprovados ou rejeitados, os seus intervenientes serem notificados do seu estado final.

1.2 Estado da arte

Existem diversas aplicações de *workflow* disponíveis no mercado, de diferentes tipos. Fornecem uma variedade de recursos e funcionalidades para auxiliar a automação e otimização de fluxos de trabalho, como por exemplo:

- *Modelação de processos* - Ferramentas como o *Microsoft Visio* [1], *Bizagi* [2] e *Lucidchart* [3] permitem a criação de diagramas de processo visualmente apelativos, que representam as etapas, participantes e regras de negócios.

- *Automação de tarefas* - Aplicações como o *Nintex* [4], *Kissflow* [5] e *Zapier* [6] possibilitam a automação de tarefas sequenciais, como encaminhamento automático de documentos, notificações de tarefas pendentes e escalonamento de prazos.

- *Colaboração e comunicação* - Ferramentas como o *Microsoft Teams* [7], *Asana* [8] e *Slack* [9] permitem a colaboração em tempo real, partilha de documentos, troca de comentários e trabalho em equipa dentro de um ambiente colaborativo.

- *Gestão de documentos* - O *SharePoint* da *Microsoft* [10], *Dropbox Business* [11] e *Google Workspace* [12] (anteriormente *G Suite*) fornecem recursos avançados de gestão de documentos, incluindo armazenamento na nuvem, controlo de versão, histórico de revisões e recursos de pesquisa.

- *Monitorização e relatórios* - Aplicações como o *Tallyfy* [13], *KiSSFLOW* [5] e *Process Street* [14] oferecem recursos avançados de monitorização em tempo real, permitindo que os utilizadores acompanhem o estado das tarefas, e gerem relatórios personalizados para análise.

- *Integração com outros sistemas* - Ferramentas como o *Zapier* [6], *Microsoft Power Automate* [15] (anteriormente *Flow*) e *Dell Boomi* [16] fornecem recursos de integração para interligarem diferentes sistemas e aplicações, permitindo que os dados fluam entre estes sem grandes restrições.

- *Segurança e conformidade* - O *IBM BPM* [17] (*Business Process Manager*), *Appian* [18] e *Laserfiche* [19] oferecem recursos avançados de segurança, incluindo controlo de acesso baseado em funções, criptografia de dados, auditoria e conformidade com regulamentos de privacidade e segurança.

Quanto à disponibilidade e preço destas ferramentas de *workflows*, varia de acordo com o fornecedor e a solução escolhida. Algumas ferramentas são gratuitas, oferecendo recursos básicos, enquanto outras, pagas, oferecem funcionalidades mais avançadas. O preço pode ser baseado numa assinatura mensal ou anual, ou em licenças vitalícias.

Apesar da existência deste tipo de aplicação ser muita ampla no mercado, este projeto focou-se em desenvolver algo específico para a instituição, preservando algumas funcionalidades genéricas para possível implementação por outras instituições ou empresas. Além disso, para a correta implementação de um *software* de *workflow* documental, a escolha no mercado seria ou muito ambígua, ou envolveria mais do que um fornecedor, visto que não foi encontrado software de *workflow* de momento no mercado que fornece-se todas as funcionalidades previstas deste projeto numa única aplicação.

1.3 Organização do documento

Este relatório foi organizado para que conste nos próximos capítulos a informação necessária para compreender a arquitetura da solução como a implementação dos seus componentes, está descrita no capítulo 2 , e a sua implementação no capítulo 3. No capítulo 4 apresentamos o caso de estudo usado para demonstrar o funcionamento da solução implementada, e no capítulo 5 terminamos com a conclusão e trabalho futuro.

Capítulo 2

Solução proposta

Neste capítulo é apresentada e discutida a solução proposta e os componentes constituintes desta, bem como são fundamentadas as tecnologias utilizadas, para a respetiva implementação. O sistema proposto considera dois papéis fundamentais: Administrador e Não Administrador. Estes terão acesso ao sistema através de duas aplicações, uma *Web* e outra móvel, cada uma com duas interfaces distintas de acordo com o tipo de papel. Além das aplicações cliente, o sistema será constituído por uma base de dados e uma *Web* Application Programming Interface. Na Secção 2.1 descrevemos os utilizadores da aplicação, na Secção 2.2 a condução de processos e na Secção 2.3, a arquitetura proposta.

2.1 Utilizadores do sistema

O sistema suporta dois tipos de utilizadores, cujas funcionalidades associadas se descrevem em seguida:

- **Administrador -** O administrador é o primeiro utilizador da aplicação. Este é responsável pelo registo dos outros utilizadores na plataforma, pela criação e atribuição de papéis, incluindo a gestão de outros administradores. O administrador também é responsável pela criação de *templates* de processos, definindo todas as etapas que estes devem ter, quais são os responsáveis pela aprovação de cada etapa e quais são os utilizadores que podem usar esse *template* para criar novos processos. Um administrador possui todas as capacidades de um utilizador não administrador.

- **Não Administrador -** O utilizador não administrador terá a capacidade de criar novos processos a partir de *templates* dos quais têm acesso e/ou intervir em processos onde é interveniente.

2.2 Condução de processos

O processo de criação e participação em *workflows* tem o seguinte procedimento:

1. O Administrador deve criar pelo menos um *template* de um processo e definir quais são os utilizadores que têm acesso a este. Ao criar o *template*, o administrador define todos os parâmetros importantes para esse tipo de processo, tais como o número de etapas e os responsáveis pela aprovação em cada etapa.

2. O utilizador acede à interface de novo processo, seleciona o *template* pretendido e define todos os parâmetros e documentos necessários para aprovação.

3. Quando um novo processo é iniciado, os utilizadores responsáveis pela primeira etapa recebem uma notificação por e-mail, a informar que têm uma tarefa pendente. Através desse e-mail podem aceder diretamente à tarefa através de um *link* de redirecionamento ou aceder através da interface de processos, na secção de tarefas pendentes.

4. O utilizador responsável, ao aceder aos seus detalhes da etapa, verifica os documentos submetidos no processo e submete a sua decisão através de uma assinatura simbólica (Aprovado ou Não Aprovado).

5. Após todos os responsáveis pela etapa submeteram a sua aprovação, a etapa é dada como terminada, levando o processo à etapa seguinte, notificando os utilizadores dessa nova etapa que têm uma tarefa pendente.

6. O processo continua a progredir desta forma, sendo finalizado através de uma das seguintes formas:

   - Todas as etapas são aprovadas até ao fim do processo, terminando um como processo aprovado;
   - O processo é interrompido numa das etapas com a não aprovação de um dos responsáveis, terminando como não aprovado.

7. No fim, o autor do processo recebe um e-mail, a informar o estado final (Aprovado ou Não Aprovado) do processo.

2.3 Arquitetura proposta

A arquitetura do sistema proposto, representado na Figura 2.1, tem como elementos essenciais: **Base de dados -** Armazena todos os dados da aplicação em modelo relacional e os documentos submetidos em cada processo. **Servidor Web -** Implementa uma *API RESTful*, responsável por todas as comunicações e lógica do sistema. Este recebe pedidos por parte dos clientes (aplicação *Web* e aplicação móvel) emitindo respostas de acordo com o solicitado. Verifica a autenticação do acesso a dados persistidos na base de dados. **Servidor de autenticação -** Permite a autenticação de utilizadores através de um servidor de autenticação externo, podendo este estar associado aos perfis da instituição. **Aplicação *Web* e Aplicação Móvel -** Serve de interface de interação para os utilizadores.

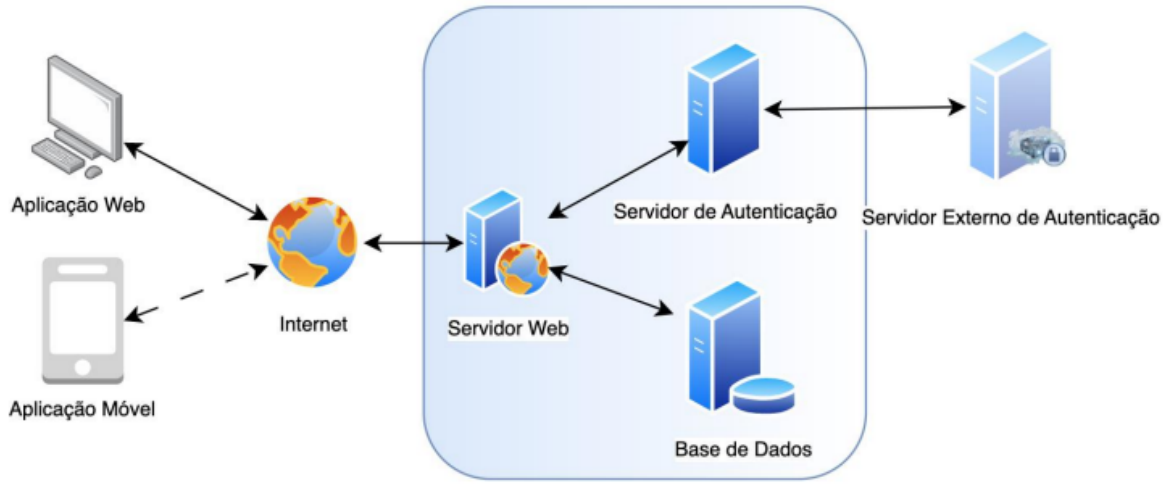


Figura 2.1: Arquitetura do sistema proposto

Capítulo 3

Desenvolvimento da solução

Dados os objetivos definidos para este projeto, neste capítulo são apresentados os detalhes de implementação de cada um dos componentes desta aplicação. Na Secção 3.1 descreve-se a base de dados, na Secção 3.2 aborda-se a Web API e na Secção 3.3 apresenta-se a implementação da aplicação Web.

3.1 Base de Dados

Nesta secção é apresentada a proposta de solução para a base de dados utilizada, que armazena a informação de domínio da Web API. Em primeiro lugar, desenhou-se o modelo de dados, sendo este um Modelo Entidade-Associação (EA). Iniciou-se pela definição das entidades necessárias à existência de um *document workflow*, começando-se pelos utilizadores.

3.1.1 Utilizador

Os utilizadores da aplicação são as entidades envolvidas ativamente nos processos. Contudo, cada utilizador tem permissões distintas consoante os papéis que assumem em cada etapa. Os atributos desta entidade contém informações importantes tais como: o e-mail, nome, *password* (*hash*), e *token* de autenticação. Concebeu-se assim a entidade **Utilizador**, ilustrada na Figura 3.1.

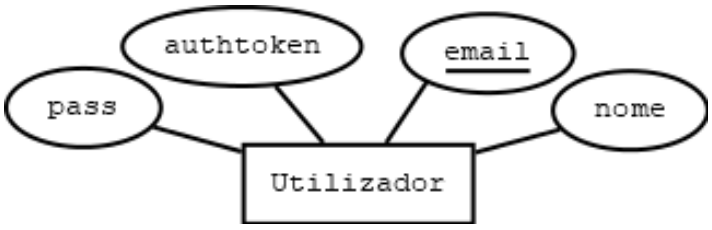


Figura 3.1: Entidade **Utilizador**

3.1.2 Papel

O **Utilizador** tem as suas capacidades limitadas por um conjunto de permissões. Estas permissões são atribuídas como papel, dando acesso ao utilizador para participar em processos.

A entidade **Papel** tem como atributos, o nome e descrição, representado na Figura 3.2.

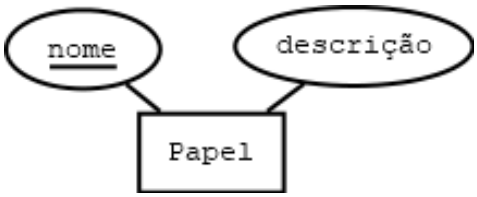


Figura 3.2: Entidade **Papel**

3.1.3 Template

Um **Utilizador** pode criar processos a partir de **Templates**, desde que tenha permissão de acesso aos mesmos. Esta permissão é dada por um **Utilizador** com **Papel** *admin*. Estes **Templates** possuem: nome, descrição e as etapas que descrevem um processo em formato *JavaScript Object Notation (JSON)*. As etapas são caracterizadas por um nome, descrição, prazo, modo de assinatura e os utilizadores responsáveis por estas. O formato escolhido para etapas, JSON, é o mais adequado devido à sua facilidade de leitura, compatibilidade com diferentes linguagens de programação e flexibilidade na estruturação dos diferentes tipos de dados. Os responsáveis por cada **Etapa** podem ser utilizadores individuais ou grupos de utilizadores com um determinado papel. A entidade **Template** também possui o atributo ativo, que define se este pode ser usado para a instanciação de novos processos. Um **Template** não pode ser eliminado após a sua criação, apenas pode ser desativado, de forma a ser possível mantermos o histórico de processos terminados que usaram esse **Template**, garantindo a integridade da base de dados. A entidade **Template** é representada na Figura 3.3.

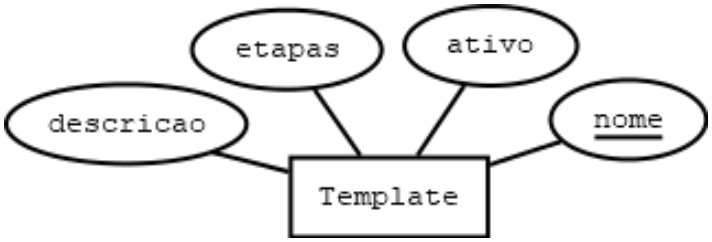


Figura 3.3: Entidade **Template**

3.1.4 Processo

Com os **Templates** descritos previamente, os utilizadores com acesso a estes podem criar uma instância do processo descrito no **Template**. A entidade **Processo** é necessária para armazenar a informação descrita no **Template**, e contém assim os atributos: *id*, *nome*, *descrição*, o estado, data de início e a sua data de fim. O estado pode variar entre *PENDING* (pendente), *APPROVED* (aprovado), ou *DISAPPROVED* (não aprovado). A entidade **Processo** é representada na Figura 3.4.

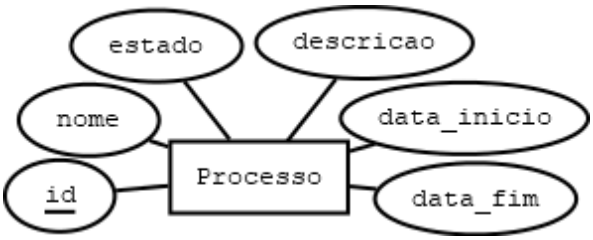


Figura 3.4: Entidade **Processo**

3.1.5 Etapa

Um processo é um procedimento faseado com uma ou mais etapas, que são realizadas pela ordem descrita no **Template**, por meios de assinaturas para a sua aprovação, realizadas pelos utilizadores envolvidos. As entidades **Etapa** contêm informação relativa a essa fase do processo: id, nome, descrição, prazo de conclusão, data de inicio e data de fim e o estado que tal como o processo pode ter os mesmos estados que este. Cada etapa possui um índice respetivo à ordem nesse **Template**. A etapa também possui um modo de assinatura que pode ser Unânime (*Unanimous*) ou Maioritário (*Majority*). O modo de assinatura Unânime, define que uma etapa só é concluída quando todos os responsáveis registaram a sua aprovação através da assinatura, e o modo Maioritário define que uma etapa é concluída quando mais de metade dos responsáveis registam a sua aprovação. A reprovação de uma etapa por parte de um dos responsáveis implica a reprovação de todo o processo. O prazo de uma etapa é dado em dias e é indicativo do tempo previsto para a conclusão desta, servindo apenas para efeitos de notificação dos responsáveis quando esta está perto do término. A entidade **Etapa** é representada na Figura 3.5.

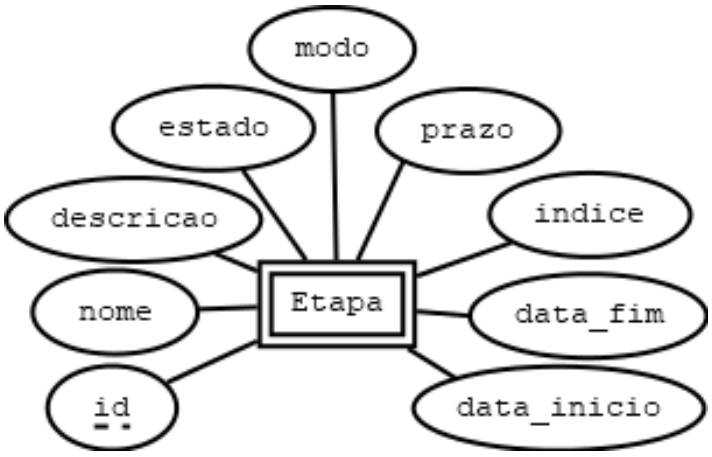


Figura 3.5: Entidade **Etapa**

3.1.6 Documento

Dentro de cada processo, podem ser submetidos vários documentos de diferentes tipos. Estes são armazenados em *FileSystem*, sendo a sua entidade na base de dados uma referência

importante para armazenar o caminho para este ficheiro, bem como o seu identificador, nome, tipo e tamanho. A entidade Documento é representada na Figura 3.6.

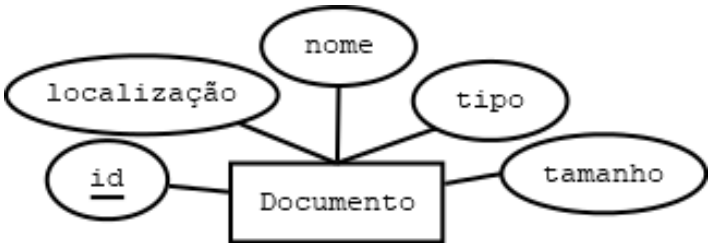


Figura 3.6: Entidade **Documento**

3.1.7 Comentário

Dentro de cada etapa, os utilizadores envolvidos podem realizar vários comentários que visam criticar ou fornecer informação relevante. Esta entidade apresenta como atributos o seu identificador, o texto, a hora e data da sua criação. A entidade **Comentário** é representada na Figura 3.7.

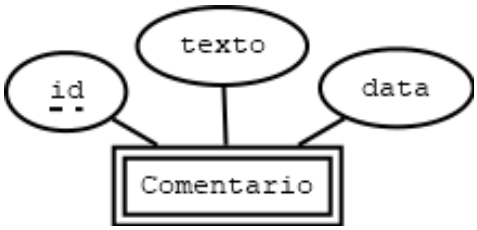


Figura 3.7: Entidade **Comentário**

O projeto usa uma base de dados relacional, implementada utilizando *PostgreSQL*, em conjunto com armazenamento de documentos em *FileSystem*. A seleção do *PostgreSQL* baseia-se nas características comuns das bases de dados relacionais em conjunto com o facto de ser *open source*, e ter capacidade de armazenar atributos no formato *JSON*. Optou-se pelo armazenamento dos documentos em *FileSystem* devido à simplicidade, rapidez na transferência de ficheiros e ao baixo custo, não existindo a necessidade de recorrer armazenamento em servidores externos. O modelo EA desenvolvido está representado na Figura 3.8. É de salientar a relação entre as entidades **Utilizador** e **Etapa**, que representam os responsáveis por cada etapa. Esta relação é caracterizada pelo valor simbólico da assinatura, *true* (aprovado) ou *false* (não aprovado), a data da assinatura, o *id* da notificação gerada e a data de inicio das notificações. O *id* de notificação serve para a sua posterior desativação e a data de inicio das notificações é calculada para um dia antes do término do prazo estabelecido na etapa. Se o prazo só tiver um dia, as notificações só começam no dia seguinte. É enviado um e-mail a cada responsável de etapa no inicio desta, e no caso deste não registar a sua resposta, são enviados lembretes um dia antes do fim do prazo e de dois em dois dias após essa data. A

data de inicio das notificações serve para efeitos de persistência de notificações agendadas, por se tratarem de ações do servidor, é necessário no caso em que este seja reiniciado, que as notificações que estavam ativas não sejam perdidas e sejam ser reagendadas novamente.

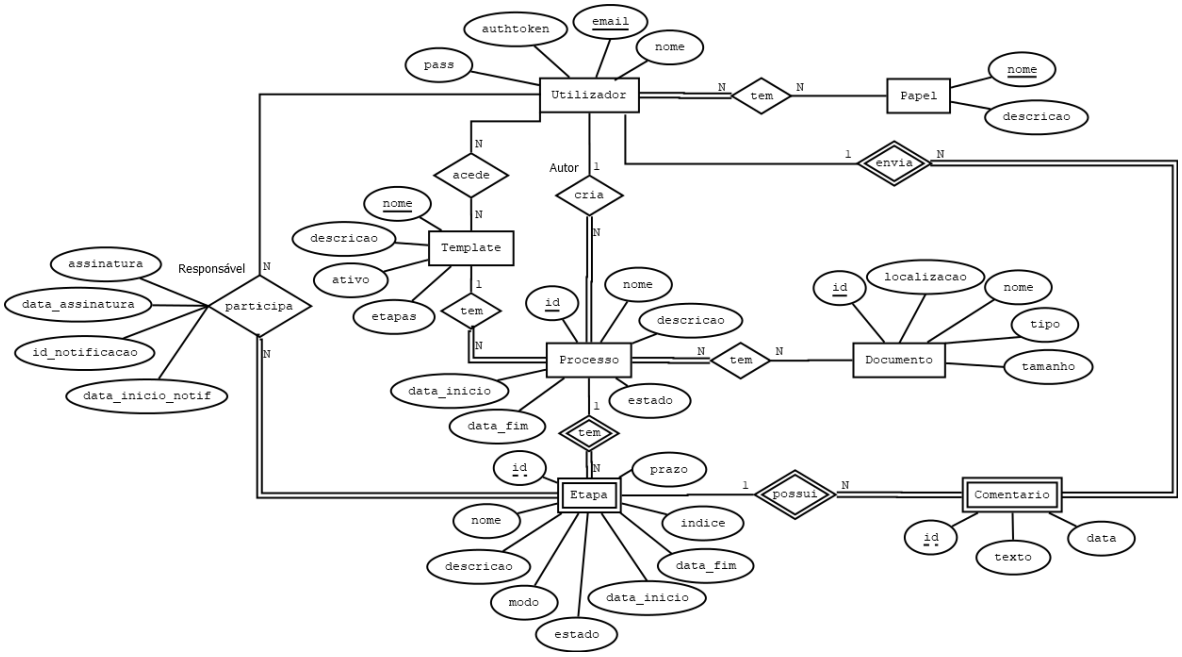


Figura 3.8: Modelo Entidade-Associação da solução proposta.

3.2 Web API

Nesta secção é descrita a proposta para a construção da *Web API* do Document Workflow Platform (DWP).

A Web API disponibiliza um conjunto de *endpoints* que os clientes utilizam para comunicar com o servidor. Estes *endpoints* possibilitam as seguintes operações para diferentes tipos de utilizador:

1. Administrador:
    - Registar novos utilizadores;
    - Obter lista dos utilizadores;
    - Obter informações sobre utilizadores registados;
    - Criar papéis;
    - Eliminar papéis;
    - Obter lista de papéis;
    - Obter detalhes de um papel;
    - Atribuir papéis a utilizadores;

- Remover papéis de utilizadores;
- Obter lista de utilizadores com determinado papel;
- Submeter novos *templates*;
- Obter lista de *templates*;
- Obter detalhes de um *template*;
- Ativar uso de *templates*;
- Desativar uso de *templates*;
- Autorizar acesso de utilização de *templates* para utilizadores;
- Remover acesso de utilização de *templates* para utilizadores;
- Obter lista de utilizadores com acesso a um *template*.

2. Não Administrador:

- Consultar perfil e papéis atribuídos;
- Consultar *templates* aos quais tem acesso.
- Alterar palavra-passe;
- Criar novo processo;
- Consultar processos criados em estado pendente;
- Consultar processos criados em estado terminado;
- Transferir documentos associados a um processo;
- Consultar tarefas em etapas pendentes;
- Consultar tarefas em etapas terminadas;
- Consultar responsáveis de uma etapa;
- Consultar assinaturas de uma etapa;
- Aprovar tarefa em etapa pendente;
- Não aprovar tarefa em etapa pendente;
- Ver comentários numa etapa;
- Adicionar comentários numa etapa.

Para implementar a Web API recorre-se à *framework Spring Web MVC* com recurso à linguagem *Kotlin*. A Web API processa os pedidos *HTTP*, fornecendo recursos baseados na API *Java Servlet*.

A sua construção é baseada:

- num servidor *Tomcat* como *servlet* subjacente;

- Num *dispatch servlet*, registado nesse servidor de *servlet*, para todos os caminhos filhos de um caminho base;
- em *Handlers* definidos pelo utilizador, responsáveis pelo processamento de pedidos de mapeamentos específicos, normalmente definidos por métodos *HTTP* e modelos de caminho. Estes *handlers* são definidos através de métodos de instância em classes *controller* (anotadas com "@Controller"ou "@RestController");
- em injeção de dependências automática, através do seu desenho baseado em inversão de controlo (*IoC*) [20]. O *spring context* é o componente responsável por instanciar e gerir as dependências da aplicação, como *controllers* e *services*. Também são usados *beans*, que representam objetos criados e geridos pelo *context*;
- num *pipeline* entre o *dispatch servlet* e os *handlers* definidos pelo utilizador, que: i) Obtém as informações do pedido e fornece ao *handler*; ii) Anexa o resultado retornado pelo *handler* à resposta; iii) Executa operações antes e após as mensagens do pedido, e do resultado do *handler*.

A *Web API* foi separada em quatro camadas, representadas na Figura 3.9:

- A primeira camada é a do *HandlerInterceptors*, responsável pela interceção dos pedidos, utilizados para verificar o seu conteúdo ou para adicionar informação ao mesmo antes deste chegar ao *handler*; um exemplo de um *HandlerInterceptor* é o *AuthenticationInterceptor*, responsável pelo tratamento de pedidos que requerem autenticação;
- A segunda camada é a dos *controllers*, responsável pela receção e encaminhamento da informação dos pedidos para os *handlers* correspondentes;
- A terceira camada é a dos *services*, responsável por maior parte da lógica da aplicação. Sabe como obter informação ou executar uma dada ação, normalmente recorrendo a operações disponibilizadas pelos diversos repositórios;
- A quarta camada é a de *repositories*, onde são realizadas operações de comunicação com a base de dados. São executadas *queries* de *insert*, *update* ou *delete* que podem retornar valores que são posteriormente mapeados para Data Access Object (DAO) através do *JDBI* e devolvidos às camadas superiores.

Para realizar o acesso à base de dados relacional *PostgreSQL* através dos DAO, foi selecionada a API declarativa da *JDBI* [21]. Tomou-se esta decisão devido a ser: construída sobre a *Java Database Connectivity (JDBC)*, fornecendo uma API com uma abordagem/conceito semelhante, tendo, assim, uma curva de aprendizagem mais curta, fornece uma API mais simples e natural comparativamente com a disponibilizada pela *JDBC*, e tem suporte para a linguagem *Kotlin*.

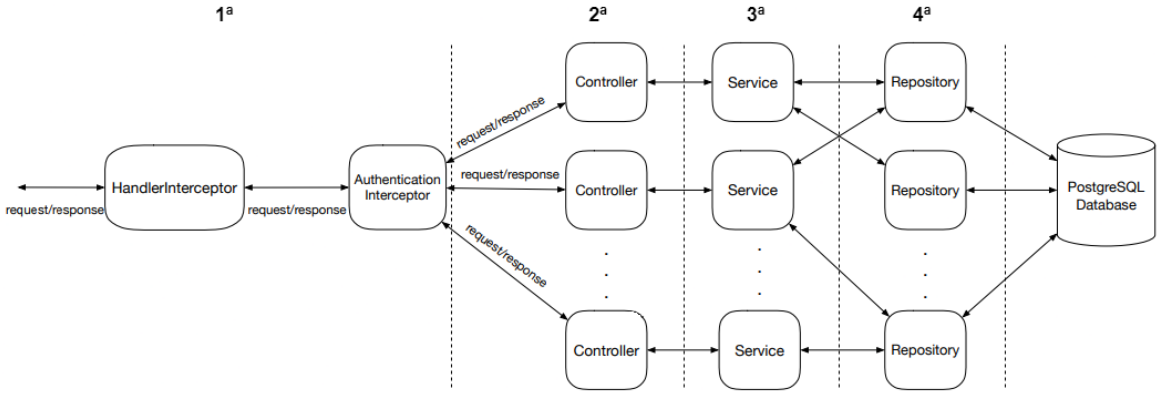


Figura 3.9: Arquitetura da Web API

Na elaboração deste projeto foi restrito o acesso à *Web API* através de um sistema baseado no uso de *Cookies*, controladas e fornecidas pelo servidor e usadas posteriormente pelo cliente para garantir o acesso à maioria dos *endpoints* suportados.

Entende-se por Autenticação todo o processo que envolva verificar e validar uma reivindicação de identidade.

Dos vários métodos que existem para autenticar o cliente com o servidor, foi escolhido o uso de *Cookies* também pelo facto que durante todo o estágio de utilização destas conseguimos, enquanto servidor, aferir que o cliente detém conhecimento único que garanta o seu acesso, pois teve este que previamente passar pelo processo de *login* na aplicação, como também conseguimos aferir que o cliente detém informação somente dele, que no caso da nossa aplicação é o *token* que circula do cliente para o servidor e no sentido inverso.

Para implementar esta peça fundamental do trabalho seguimos a arquitetura normalmente recomendada:

- extraímos a informação de autenticação do pedido;
- validamos o formato da informação;
- passamos a informação pela função de autenticação;
- e por último pela função de validação.

A implementação deste modelo na *framework Spring* passa pelo uso dos *Handler Interceptors* responsáveis por intercetar os pedidos antes de estes passarem aos seus respetivos *Handler Controllers*. Como foi descrito, o primeiro passo no modelo de autenticação é a extração da informação do pedido e esta é realizada através da obtenção do *HTTP Header* que identifica o pedido como um pedido autenticado. Logo após a extração do *Header* recorremos a um Processador de Autenticação implementado neste projeto e responsável por validar o formato, conteúdo e identidade, tendo nesta última a necessidade de aceder à base de dados através de um método fornecido pela camada de Serviço. Se a operação descrita até agora

for concluída com sucesso, o cliente confirmou a sua identidade perante o servidor, estando agora o pedido pronto a passar para a próxima fase, o processo de autorização do cliente.

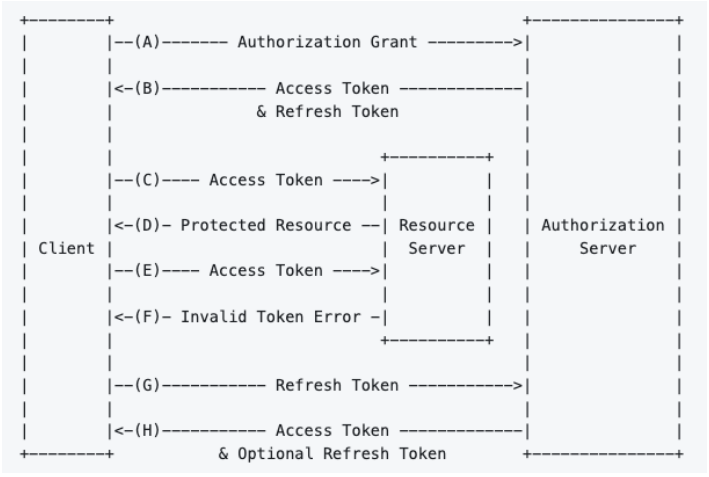
Mas antes de ser descrita esta fase de autorização gostaríamos de deixar umas considerações em relação à metodologia que foi usada, elucidar o leitor de possíveis alternativas e possíveis vetores de ataque à integridade da identidade do cliente e do correto e suposto uso da nossa aplicação.

Dadas as circunstâncias em que se encontra enquadrado este trabalho e tendo como base aquilo que nos propusemos a realizar dentro do tempo que estipulámos, convém estar ciente e conhecer as vulnerabilidades desta solução para enquadrar o leitor das possíveis melhorias a serem feitas numa fase futura deste projeto.

- Uma vez descoberta a *Cookie*, um utilizador mal intencionado pode fazer passar-se por outro utilizador;

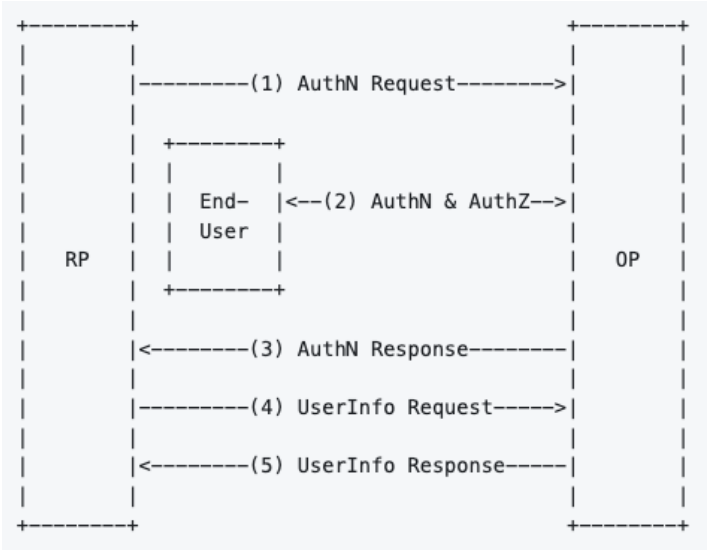
- A *Cookie* não sofre qualquer alteração depois de gerada de forma aleatória.

Este nível de segurança dado pelo uso de *Cookies* pode ser aumentado recorrendo a um servidor externo, representado na Figura 3.2, que isole o estado do cliente e a sua informação confidencial de possíveis atacantes, tendo sido desde o início tal ideia tida em conta quando foi desenhada e planeada a arquitetura da solução, representada na Figura 2.1. Já nessa fase e recorrendo a um outro serviço a ideia passaria por substituir o uso de *Cookies* pelo uso do protocolo *OAuth2.0* 3.10 e da camada *OpenID Connect* 3.11, o que elevaria o nível de segurança da interação do cliente com o servidor e isolava menos uma preocupação no grau de complexidade de implementação do servidor.



1.0

Figura 3.10: Diagrama de Funcionamento do Protocolo *OAuth2.0* [23]



1.0

Figura 3.11: Diagrama de Funcionamento do Protocolo *OpenID Connect* [24]

A aplicação foi desenhada desde início já a pensar neste problema e encontra-se neste momento pronta para a receção desse novo módulo.

O controlo transacional não seria afetado com estas mudanças, pois a introdução da chamada assíncrona ao servidor externo estaria fora do que é o percurso da transação já dentro do respetivo *handler controller* na camada dos serviços, podendo assim o grupo afirmar que os eventuais problemas em garantir as propriedades *ACID* ficariam minimizados.

Tais modificações permitiriam também numa outra fase poder alargar a abrangência da aplicação para uso comercial, podendo a aplicação tirar partido daquilo que são as implementações de autenticação de outras empresas, o que facilitaria o enquadramento da nossa aplicação nas arquiteturas das empresas para mitigar o problema de *Document Workflow*.

Concluindo, a decisão de implementar autenticação recorrendo numa primeira instância ao uso de *Cookies* encontra-se justificada pelos motivos em cima enumerados, estando o grupo de trabalho ciente das limitações da mesma e ciente também do que seriam os próximos passos

a tomar.

3.2.1 Sistema de Autorização

Além de todo o sistema de Autenticação contamos ainda na nossa aplicação com um sistema que controla a Autorização do cliente no acesso aos serviços.

Entende-se como Role-Based Access Control (RBAC) o controlo de acesso baseado em papéis do utilizador no sistema e nas regras que estão definidas para os mesmos.

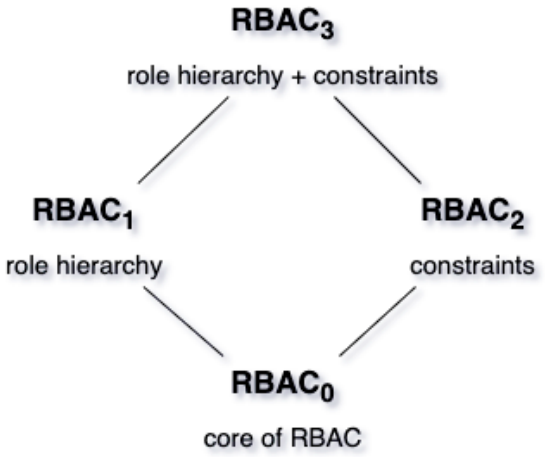


Figura 3.12: Diferentes Modelos *RBAC* [25]

Em relação à arquitetura da solução para este problema foi seguida a mesma lógica do módulo de autenticação, tendo portanto, o pedido, que passar pelo *Handler Interceptor* dedicado ao controlo de autorizações, onde é, novamente, verificada:

- a informação retornada pelo processo de autenticação;
- validado o formato da informação remetente aos papéis do utilizador;
- passados os papéis e o utilizador pela função de autorização;
- E por último pela função de validação.

Porém, desta vez, com a necessidade de associar *metadata* aos *handler controllers* que gostaríamos que somente fossem acedidos por utilizadores Administradores, foram então criadas Classes para Anotações, a primeira de nome "Admin"e associada a funções, servirá para identificar as funcionalidades restritas aos papéis Administradores.

Assim, adotou-se uma implementação baseada no modelo *RBAC0* para implementar o mecanismo dos papéis, podendo um utilizador ter múltiplos papéis durante a sessão, não havendo múltiplas sessões, não existindo hierarquia entre os papéis, sendo deixado, por resolver, todo o controlo de acesso a recursos a nível aplicacional dentro da lógica interna de outros *handler controllers* e dos serviços a que estes recorram.

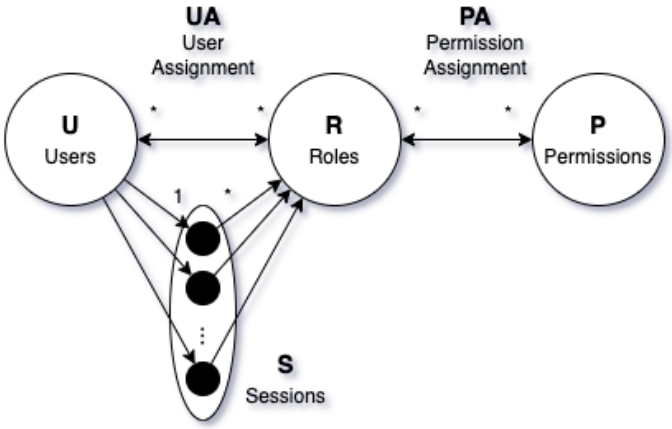


Figura 3.13: Arquitetura do Modelo *RBAC0* [26]

Ainda no âmbito do processo de Autorização o *handler interceptor* de Autorização executa sempre a seguir ao de autenticação, por uma questão lógica, não faz sentido avaliar se um utilizador pode ou não realizar uma dada função na aplicação sem saber antes de que utilizador se tratava quando este realizou o pedido.

Esta solução a que chegámos, serve o propósito do enquadramento do trabalho, pois da mesma forma que foi justificada o uso de *Cookies* na secção 3.2, volta a ser justificada a adoção do modelo simples inspirado no *RBAC0*. A arquitetura da aplicação suporta que as funcionalidades de *RBAC* sejam estendidas para modelos superiores que envolvam hierarquia de papéis como é usado no modelo *RBAC1*, o que liberta e simplifica um pouco a complexidade da lógica domínio aplicacional se todo este processo for delegado para um *RBAC Processor*.

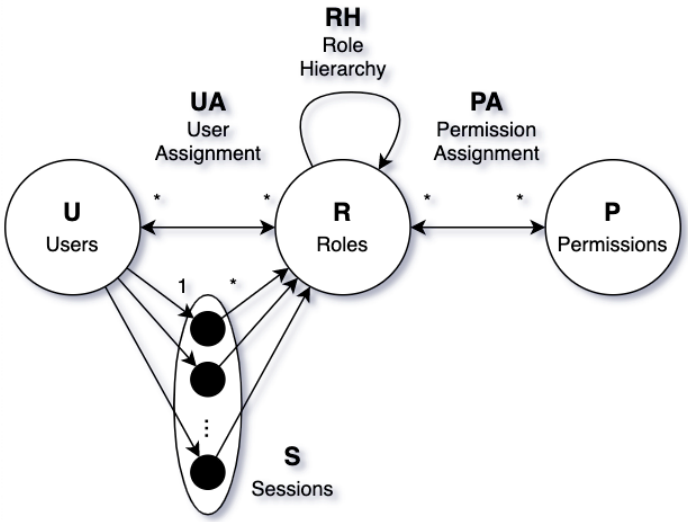


Figura 3.14: Arquitetura do Modelo *RBAC1* [26]

3.3 Aplicação Web

Nesta secção descrevem-se os detalhes de implementação da aplicação Web, referindo-se as funcionalidades de cada uma das páginas que constituem a aplicação e a ferramenta de desenvolvimento selecionada.

Decidiu-se utilizar a linguagem *Typescript* em conjunto com a biblioteca *ReactJS* [27]. Para além de já existir experiência anterior, diminuindo o tempo de aprendizagem, esta faz uso do conceito de virtual DOM (*Document Object Model*) que atualiza apenas as partes da página que sofreram alterações, em vez de realizar uma atualização total da mesma, o código de implementação de componentes é facilmente reutilizado e tem como principal objetivo ser uma biblioteca com rápida resposta, escalável e simples, podendo ser combinada com outras bibliotecas de *JavaScript*. O *ReactJS* também usa *JSX (JavaScript XML)* que permite descrever a *UI* através de uma linguagem semelhante ao *Hypertext Markup Language (HTML)*, tornando a escrita de componentes mais legível e fácil de implementar.

A aplicação Web fornece uma interface de controlo sobre todas as funcionalidades implementadas na Web API. A estrutura da Figura 3.15, representa a interligação entre as vistas da aplicação e a limitação de funcionalidades de acordo com o tipo de utilizador com sessão iniciada.

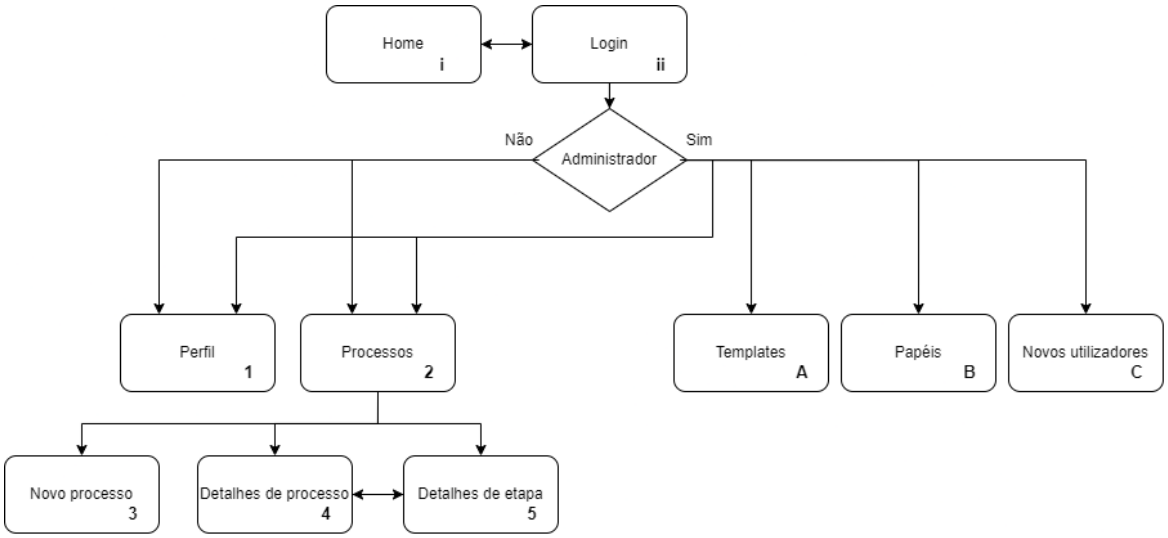


Figura 3.15: Diagrama da aplicação Web

No acesso inicial à aplicação todos os utilizadores dispõem de um acesso à página inicial (i) com uma breve descrição do projeto, na Figura 3.16, e uma interface *Login* (ii) para efeitos de autenticação do utilizador em questão, visível na Figura 3.17.

Após o utilizador introduzir as suas credenciais de acesso no painel de *Login*, a interface obtém duas formas distintas, de acordo com o critério: *i)* o utilizador não tem papel de administrador; *ii)* o utilizador tem papel de administrador.

A interface de utilizador Não Administrador *i)* disponibiliza acesso à painel de perfil, na

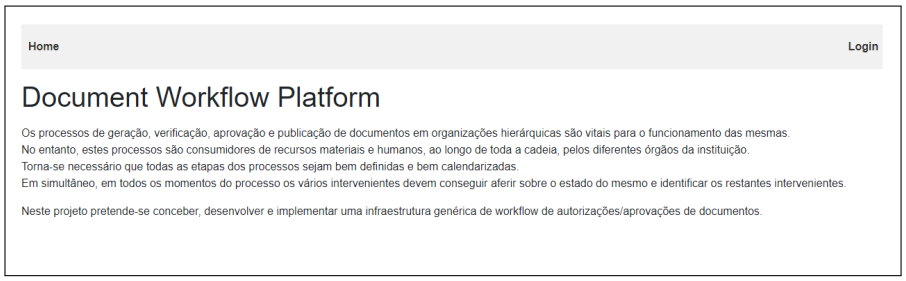


Figura 3.16: Página Inicial (i)

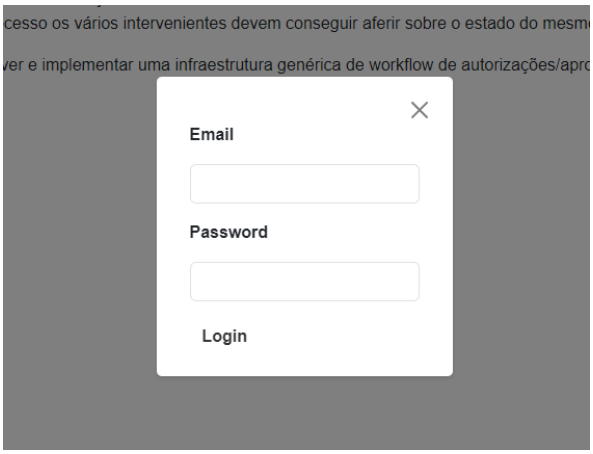


Figura 3.17: Interface de Login (ii)

Figura 3.18, e acesso ao painel de processos, visível na Figura 3.19.

A interface Perfil (1) permite consultar as informações associadas à conta, nomeadamente os papéis atribuídos, visível na Figura 3.18.

A interface Processos (2), visível na Figura 3.19, permite consultar os processos criados e as tarefas em etapas de processos, onde o utilizador se encontra inserido, podendo ser filtrados pelo seu estado, pendente ou terminado. Esta interface também dá ligação ao painel de criação de novos processos.

A interface Novos Processos (3), visível na Figura 3.20, permite ao utilizador consultar os *templates* que tem acesso e os seus detalhes. Com estes *templates* disponíveis o utilizador pode criar um novo processo, preenchendo os campos solicitados e anexando os documentos relevantes para posterior avaliação dos responsáveis em cada etapa.

Quando um processo é criado este fica disponível no painel de processos, onde posteriormente podemos ver mais informações no painel de detalhes de processo (4), visível na Figura 3.21. Na interface de detalhes de processo (4) podemos transferir os documentos carregados pelo autor, ver o estado atual e aceder aos detalhes de todas as etapas.

Na interface de detalhes de etapa (5) podemos consultar os utilizadores responsáveis pela etapa, assim como verificar as assinaturas submetidas por cada, visível na Figura 3.23. Se o utilizador estiver inserido numa etapa que necessita nesse momento da sua assinatura, são

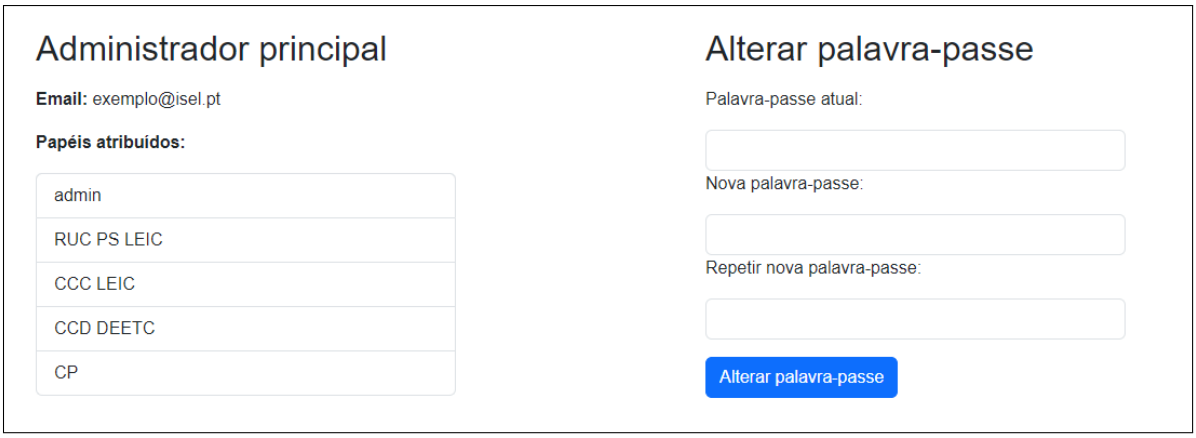


Figura 3.18: Interface de Perfil (1)

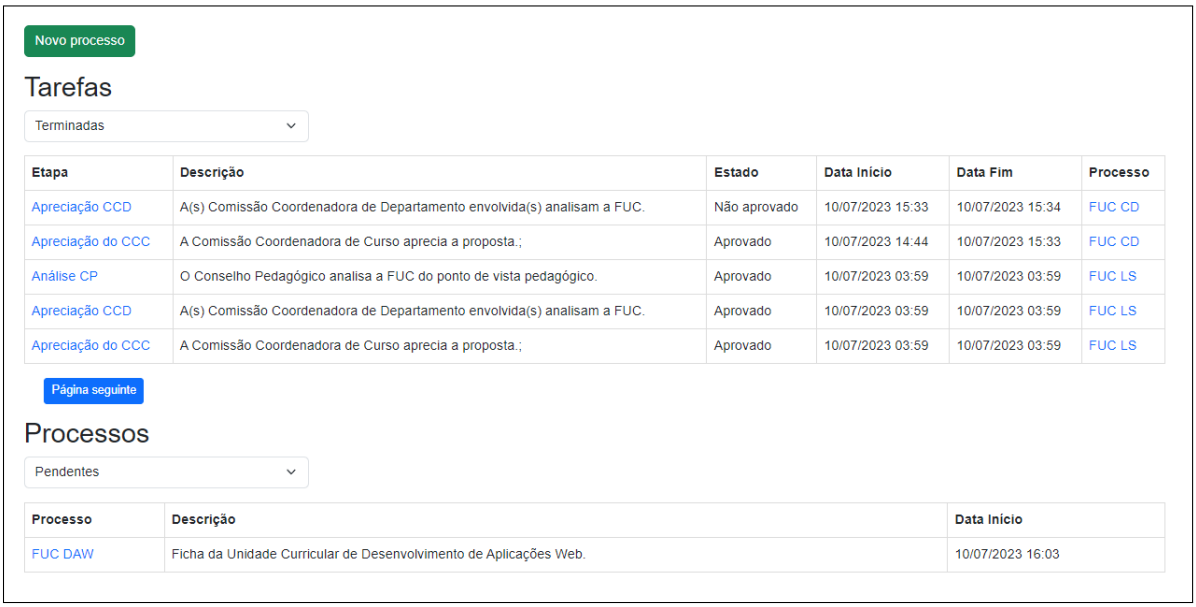


Figura 3.19: Interface de Processos (2)

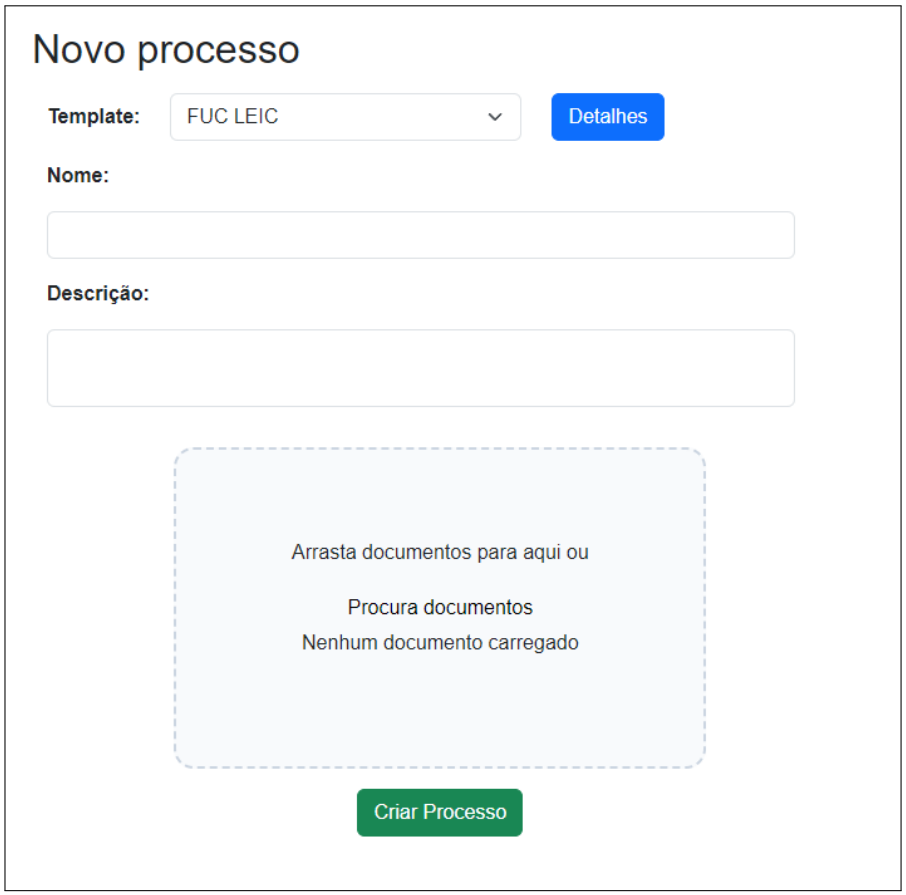


Figura 3.20: Interface de Novos Processos (3)

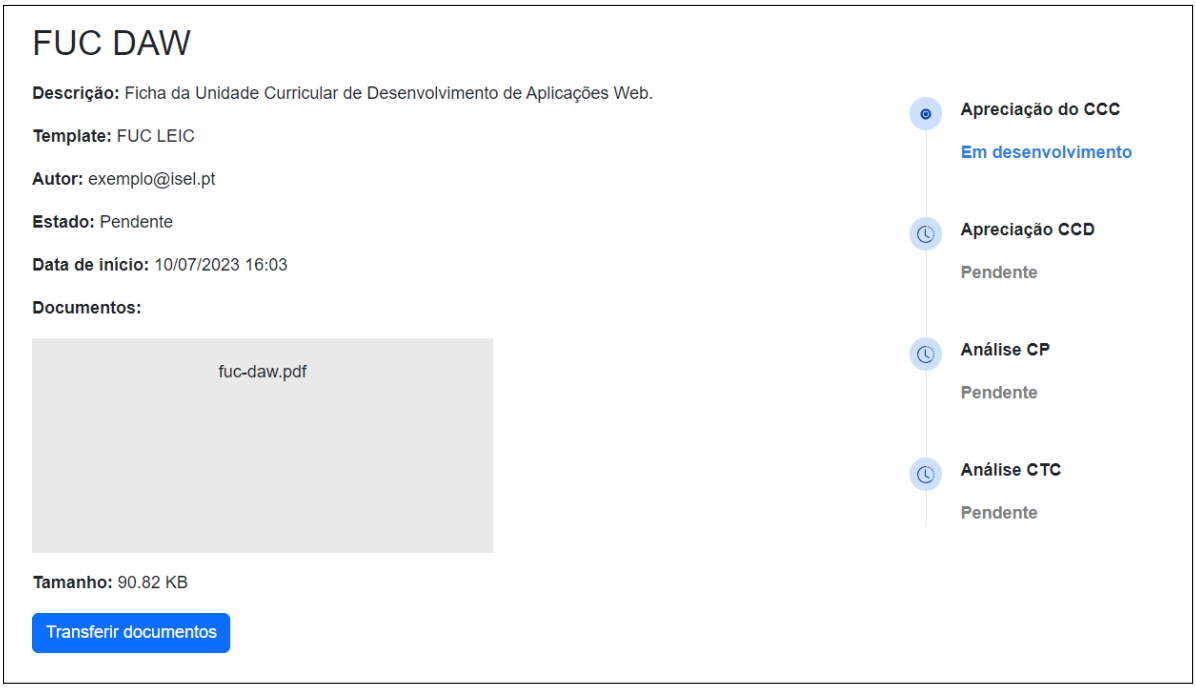


Figura 3.21: Interface Detalhes de Processo (4)

Figura 3.22: Interface Detalhes de Etapa (5)

visíveis botões que permitem ao utilizador registar a sua decisão. Neste painel também se encontram comentários que os intervenientes do processo podem consultar ou publicar novos, visível na Figura 3.22.

A interface de utilizador Administrador *ii)* possui as mesmas características da interface *i)*, acrescentando um novo painel com funções para a gestão de *templates*, papéis e utilizadores, podemos observar as diferenças entre *Navbars* de cada tipo na Figura 3.24.

Na interface *Templates* (A), visível na Figura 3.25, um administrador, pode consultar os detalhes ou desativar os *templates* existentes. À semelhança dos papéis, o administrador também pode gerir, para cada *template*, os utilizadores que o podem usar na criação de novos processos. Quando um administrador cria um novo *template*, deve preencher uma etapa de cada vez, podendo posteriormente apagar ou alterar a sua ordem usando *drag and drop*. Em cada etapa devem ser definidos os utilizadores responsáveis pela sua assinatura, podendo estes ser um grupo de utilizadores com um determinado papel ou utilizadores individuais registados na aplicação.

Na interface Papéis (B), visível na Figura 3.26, um utilizador administrador, pode consultar ou eliminar os papéis existentes, assim como criar novos. Para cada papel, o administrador

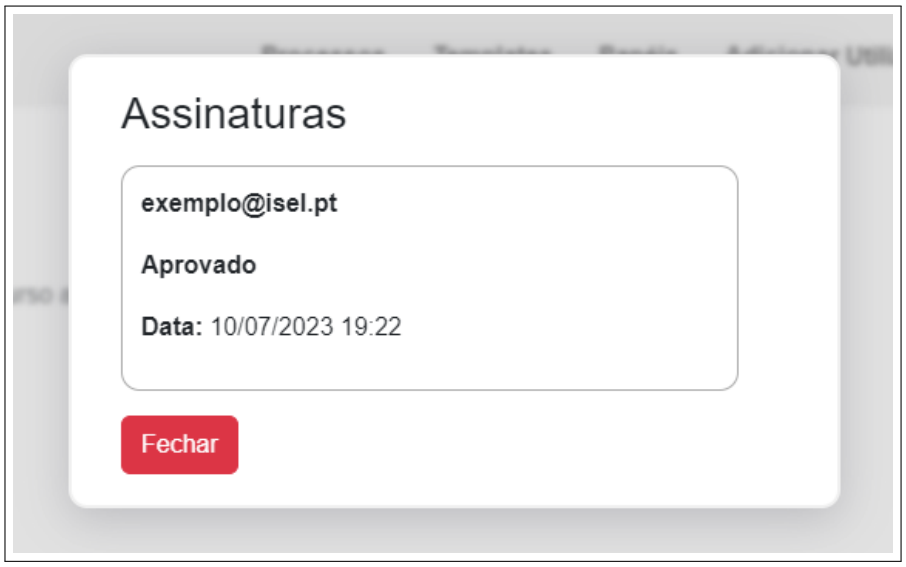


Figura 3.23: Painel de assinaturas dos responsáveis de etapa

Figura 3.24: *Navbar* de acordo com o tipo de Utilizador.

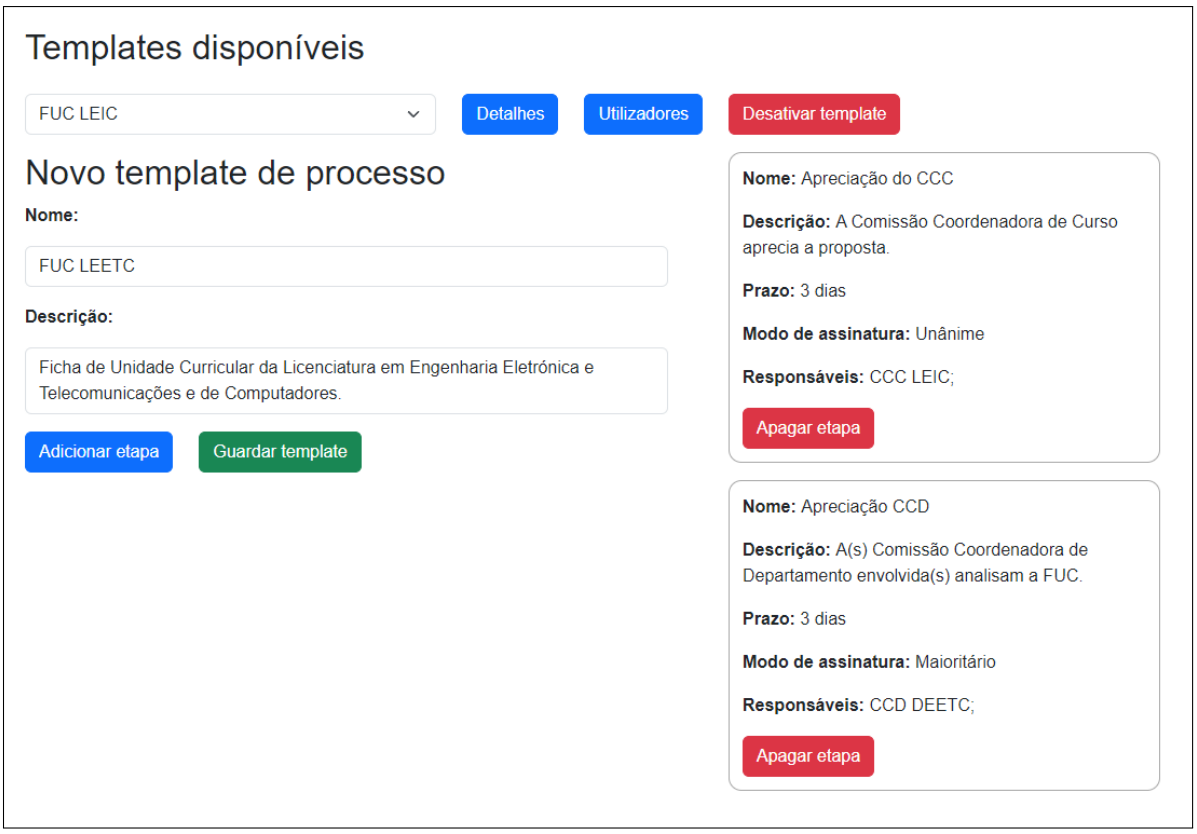


Figura 3.25: Interface de Templates (A)

Figura 3.26: Interface de Papéis (B)

pode gerir os utilizadores da aplicação que devem possuir estes papéis, tornando mais fácil a sua integração nas etapas durante a criação de novos *templates* de processos.

Nos componentes que possuem gestão de utilizadores, de forma a facilitar a pesquisa de utilizadores, estes estão agrupados pela lista de papéis ou pela lista de todos os utilizadores registados na aplicação, podendo ser posteriormente filtrados através da barra de pesquisa, estas funcionalidades são visíveis na Figura 3.27.

Na interface Novos Utilizadores (C), visível na Figura 3.28, um administrador, pode preencher os campos solicitados para adicionar novos membros à aplicação. Após premir o botão submeter é enviada uma mensagem para o e-mail indicado, que contém as credenciais de acesso para este novo utilizador.

Para o *front-end* da aplicação, é usada a linguagem *CSS* [28] e a biblioteca *Bootstrap* [29]. O *Bootstrap* é uma biblioteca de *HTML* [30], *CSS* e *Javascript open source* que oferece diversas classes para estruturar de forma simples a interface da aplicação.

A solidificação da estrutura *Backend* e *Frontend* é crítica para o correto funcionamento da aplicação. Apesar do aspeto visual da aplicação ser importante para os seus utilizadores, o desenvolvimento da estrutura base da aplicação é o foco principal do projeto.

A aplicação *Web* contém várias dependências necessárias para o correto funcionamento desta. Como exemplo, durante o processo de *Login*, no componente respetivo, não só é usado um elemento *Form* da biblitoteca *React* para o preenchimento dos campos necessários (*E-mail* e *Password*), como também funções da biblioteca *Validator*, como a *isEmail* para averiguar se os campos preenchidos pelo utilizador estão no formato correto e se contém todos os caracteres obrigatórios. Estas bibliotecas, antes de serem importadas, precisam primeiro de ser instaladas, sendo automaticamente configurada a sua presença como dependência no

Figura 3.27: Gestão de utilizadores

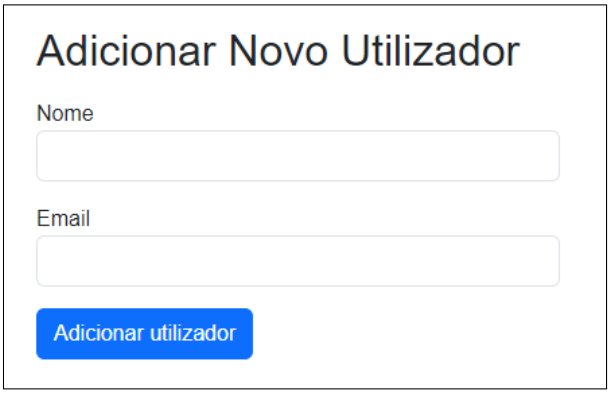


Figura 3.28: Interface de Novos Utilizadores (C)

ficheiro *package.json*. A presença de múltiplas dependências no projeto, implica o uso de uma ferramenta de instalação rápida para prevenir que manualmente seja configurada dependência a dependência, consecutivamente e por hábito de uso, foi utilizada a ferramenta *NPM (Node Package Manager)*, um gestor de pacotes para o *Node.JS*.

Além de dependências, a aplicação contém vários ficheiros de imagens e ficheiros *CSS*, que precisam de ser corretamente carregados e transformados em código ou *links*. A presença de muitas dependências, implica que estes tenham de ser empacotados para que o desempenho do *Browser* onde é executada a aplicação seja otimizado. Para este efeito, recorre-se ao empacotador de módulos *Webpack*. O *Webpack* junta e organiza, de forma otimizada, todos os módulos e dependências da aplicação, num só ficheiro, e utiliza *Loaders* para o seu correto carregamento.

Capítulo 4

Caso de estudo - ISEL

Neste capítulo é apresentado o caso de estudo do sistema desenvolvido, aplicando-o à estrutura hierárquica do ISEL, com o objetivo de verificar e aplicar o sistema num contexto real, com dados verídicos, para que as funcionalidades e objetivos propostos possam ser validados. São referidos os detalhes do que foi realizado para concluir o caso de estudo, em conjunto com as figuras relativas aos processos mais relevantes para demonstrar o correto funcionamento do sistema.

4.1 Dados utilizados

Desde o início do projeto, os orientadores mencionaram as FUC (Fichas de Unidade Curricular) dos cursos do ISEL, como um exemplo de documento que necessita de ser elaborado e revisto por vários docentes e orgãos da instituição. O documento que descreve todo o processo de aprovação de uma FUC, foi aqui considerado como um exemplo a ser implementado no sistema proposto, tendo as seguintes etapas:

- o Responsável de Unidade Curricular RUC de uma UC envia proposta de FUC para a Comissão Coordenadora de Curso (CCC) para aprovação;
- a CCC aprecia a proposta e envia a proposta aprovada para a(s) Comissão(ões) Coordenadora(s) do(s) Departamento(s) (CCD) envolvido(s) nessa UC;
- a(s) CCD envolvida(s) analisam a FUC e após a aprovação de todas as CCD, a FUC é enviada ao Conselho Pedagógico CP;
- o CP analisa a FUC do ponto de vista pedagógico e aprova, enviando de seguida para o Conselho Técnico-Científico CTC;
- o CTC analisa a FUC do ponto de vista científico e aprova, enviando para o secretariado da presidência do ISEL, o qual envia para os Serviços Académicos do ISEL e para o Serviço de Comunicação para publicação na página de internet do ISEL.

Se nalgum destes passos existirem alterações a sugerir, o processo é não aprovado e os comentários são devolvidos ao RUC e volta a repetir-se todo o processo até que seja aprovado. Todas as aprovações devem ser devidamente assinadas e datadas.

4.2 Análise de resultados

Com os dados fornecidos, iniciamos a aplicação pela primeira vez com o e-mail do primeiro utilizador, o administrador. Este administrador começa por registar os restantes utilizadores e papéis da aplicação. Os novos utilizadores recebem as credenciais por e-mail, como representado na Figura 4.1.

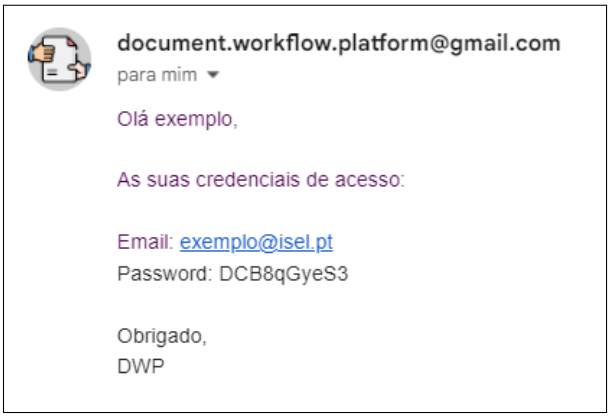


Figura 4.1: E-mail com credenciais de acesso

De seguida, o administrador atribui os papéis criados aos utilizadores registados, cria o template da FUC e define os utilizadores que o podem usar para criar novos processos. Na Figura 4.2 está representado o processo FUC e as várias etapas que o constituem.

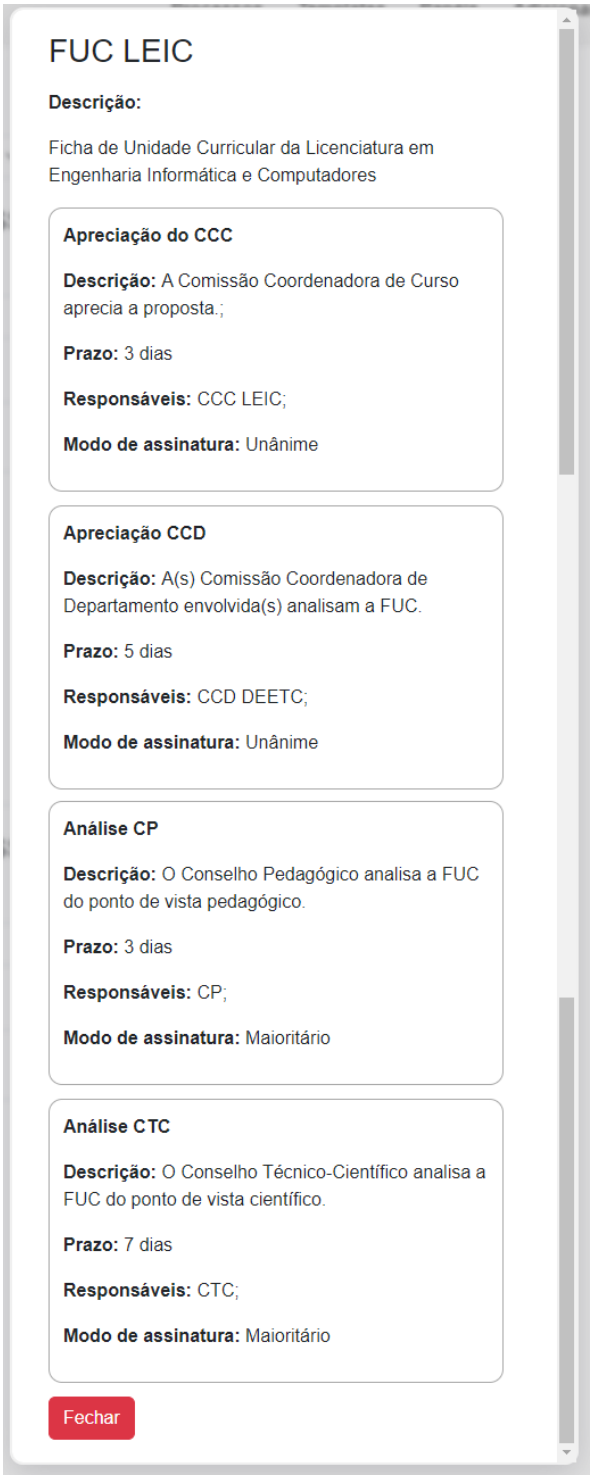


Figura 4.2: Detalhes do Template FUC

O RUC de uma unidade curricular que tiver permissões de acesso a este novo template pode instanciar um novo processo, para o exemplo recorremos à UC de Projeto e Seminário.

Quando uma nova etapa é iniciada, todos os responsáveis inseridos nesta recebem um e-mail como o representado na Figura 4.3. Se um responsável não registar a sua assinatura no prazo previsto de conclusão da etapa, este irá receber mais e-mails idênticos.

Figura 4.3: E-mail de notificação sobre tarefa pendente

No fim do processo o autor é notificado com um e-mail informativo sobre o estado com que o processo terminou, observável na Figura 4.4.

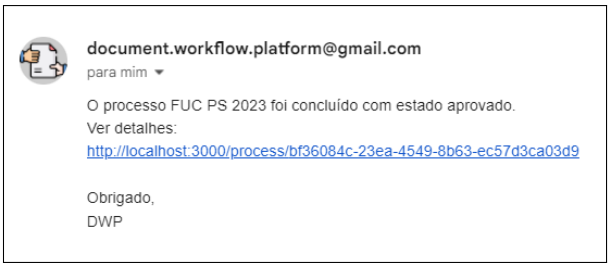
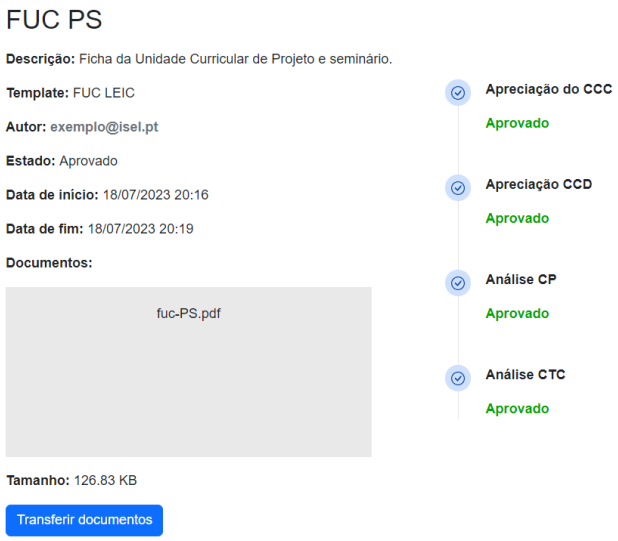


Figura 4.4: E-mail de notificação de processo finalizado

Com este processo exemplo, todos os intervenientes puderam consultar os seus detalhes e participar ativamente nas suas etapas, registando as suas decisões, comentários e notificados pela aplicação quando necessário.

Outro exemplo seria a não aprovação de um processo, que termina numa etapa em que um dos responsáveis regista essa decisão. Podemos ver a diferença entre um processo terminado com aprovação e um terminado com a não aprovação na Figura **??**

(a)



(b)

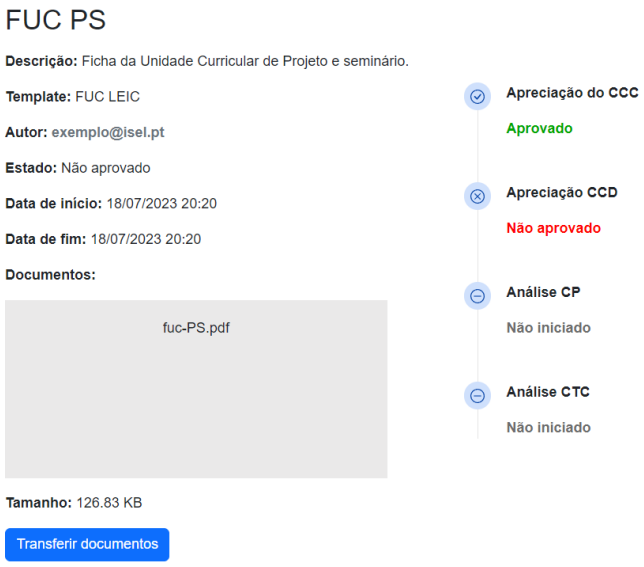


Figura 4.5: (a) Processo terminado com sucesso (aprovado em todas as suas etapas) (b) Processo terminado sem sucesso (não aprovado numa das suas etapas)

Podemos concluir, com este exemplo real de utilização, que a aplicação conseguiu registar os dados fornecidos e demonstrou o funcionamento esperado de acordo com as funcionalidades propostas.

Capítulo 5

Conclusão e trabalho futuro

5.1 Conclusão

O processo atual de aprovação de documentos no ISEL e outras instituições é feito maioritariamente através da troca de documentos físicos e/ou e-mails, sendo pouco eficiente e mais demorado que a solução que o *Document Workflow Platform* propõe. A aplicação desenvolvida pelo grupo teve como objetivo automatizar todo este processo através de um sistema constituído por: base de dados relacional em *PostgreSQL*; *Web API* em *Kotlin* com recurso à *framework Spring*; aplicação *Web* em *Typescript* com recurso à *library React*. Através desta solução desenvolvida, os processos mencionados previamente são agora facilmente realizados através da plataforma da aplicação, apresentando uma interface visualmente amigável ao utilizador, permitindo que este crie, ou participe, em processos com diversos números de etapas, criados através de *templates*. O acesso, tanto para os *templates* como para a participação em processos, é dada através de Papéis, atribuídos diretamente aos utilizadores. O utilizador participante num determinado processo pode observar o estado atual deste e das suas etapas, e também assinar na própria plataforma da aplicação, removendo a necessidade do uso de qualquer outra plataforma, como a troca de *e-mails* mencionada anteriormente, facilitando o trabalho dos envolvidos. Em suma, sem qualquer troca física ou digital (correio eletrónico por exemplo) de documentos entre utilizadores, os processos de aprovação destes facilmente poderão ser realizados numa única plataforma aplicacional.

5.2 Trabalho futuro

Finalizando o projeto, o grupo conseguiu implementar todas as funcionalidades obrigatórias propostas, propondo como trabalho futuro a implementação dos requisitos opcionais sugeridos e melhoria do que foi submetido, de forma a tornar a aplicação mais robusta e com uma experiência de utilização melhorada. Nos requisitos opcionais salientamos a integração dos meios *Autenticação.gov* (assinatura digital), como a maior prioridade, visto ser uma funcionalidade importante que viria a substituir a assinatura simbólica por uma assinatura mais segura e utilizada em diversas aplicações portuguesas inseridas neste contexto. Também seria

implementado um servidor de autenticação externo, que permitiria a autenticação com contas institucionais ou outras, deixando de existir necessidade de gerar novas credenciais de acesso. As restantes funcionalidades opcionais, seriam posteriormente implementadas, sendo estas: a criação de documentos genéricos no inicio de um processo com parâmetros fixos, para um determinado *template*; a versão móvel da aplicação.

Referências

[1] Software de criação de fluxogramas E diagramas: Microsoft visio. Microsoft. https://www.microsoft.com/pt-pt/microsoft-365/visio/flowchart-software

[2] Modern appspowered by process. Bizagi. http://www.bizagi.com/

[3] Intelligent diagramming. Lucidchart. https://www.lucidchart.com/pages/

[4] Process Management and Workflow Automation Software. Nintex. https://www.nintex.com/

[5] Kissflow. Kissflow workflow: Workflow software to automate and increase productivity 2023. Kissflow's Workflow Software. https://kissflow.com/workflow/

[6] Automation that moves you forward. Zapier. https://zapier.com/

[7] Microsoft teams. Video Conferencing, Meetings, Calling. https://www.microsoft.com/en-us/microsoft-teams/group-chat-software

[8] Asana. Manage your team's work, projects, tasks online • asana. Asana. https://asana.com/

[9] Slack. Slack is your productivity platform. Slack. https://slack.com/

[10] SharePoint, Team Collaboration Software Tools. https://www.microsoft.com/en-ww/microsoft-365/sharepoint/collaboration

[11] Secure team collaboration - dropbox business. Dropbox. https://www.dropbox.com/business

[12] Google. Google Workspace — Business Apps; Collaboration tools. https://workspace.google.com/

[13] Process Automation Software. Tallyfy. https://tallyfy.com/

[14] Checklist, Workflow and SOP software. Process Street. https://www.process.st/

[15] Take care of what's important. automate the rest. Power Automate — Microsoft Power Platform. https://powerautomate.microsoft.com/en-us/

[16] Boomi iPaaS Solutions; Tools for Cloud Connected Business. Boomi. https://boomi.com/

[17] IBM Business Process Manager Overview. IBM. https://www.ibm.com/docs/en/bpm/8.5.5?topic=mana business-process-overview

[18] Appian platform for process automation -low-code - process mining. Appian. https://appian.com/

[19] Enterprise content management, process automation. Laserfiche. https://www.laserfiche.com/

[20] the IOC Container. https://docs.spring.io/spring-framework/docs/3.2.x/spring-framework-reference/html/beans.html

[21] Jdbi 3 Developer Guide. http://jdbi.org/

[22] Java JDBC API. https://docs.oracle.com/javase/8/docs/technotes/guides/jdbc/

[23] OAuth 2.0. OAuth. https://oauth.net/2/

[24] Openid Connect. Okta. https://www.okta.com/openid-connect/

[25] Ennahbaoui, Mohammed ELHAJJI, Said. (2014). SWOT analysis of access control models. International Journal of Security and Its Applications (IJSIA),. 8. 407-424. 10.14257/ijsia.2014.8.3.39.

[26] Xue, Nian Liang, Lulu Zhang, Jie Huang, Xin. (2016). An Access Control System for Intelligent Buildings. 10.4108/eai.18-6-2016.2264493.

[27] React. https://react.dev/

[28] MozDevNet. CSS: Cascading style sheets. MDN. https://developer.mozilla.org/en-US/docs/Web/CSS

[29] Bootstrap. The most popular HTML, CSS, and JS library in the world. https://getbootstrap.com/

[30] MozDevNet. HTML: Hypertext markup language. MDN. https://developer.mozilla.org/en-US/docs/Web/HTML