Instituto Superior de Engenharia de Lisboa
Área Departamental de Engenharia de Eletrónica e Telecomunicações e de Computadores

Projeto i-on Web

Relatório Final da Unidade Curricular de Projeto

Licenciatura de Engenharia Informática, Redes e Telecomunicações

Projeto 24

Alunos

Ricardo Severino nº 45245

Catarina Palma nº 45241

Orientadores

Professor João Trindade

Professor Luís Falcão

Julho de 2021

Resumo

A iniciativa i-on surge da necessidade de melhorar a forma como é divulgada informação académica, desenvolvendo um conjunto de projetos para apoiar as atividades académicas, estando atualmente com o foco na agregação de informações de calendário e de estrutura curricular, com vista à sua redistribuição em formatos computacionalmente convenientes.

O projeto i-on Web está integrado na iniciativa i-on a partir deste ano letivo e consiste numa aplicação web para disponibilizar uma interface para *web browser* para apresentação de informação académica. A aplicação foi concebida sobre a plataforma Node.js, recorrendo à tecnologia Docker para *deployment*, aos serviços GitHub para controlo de versões e metodologia de trabalho, à Elasticsearch para armazenamento de dados da aplicação e à *framework* Bootstrap para desenvolver a *user interface*. A aplicação permite, entre outras funcionalidades, a disponibilização de informação académica e a subscrição do utilizador a turmas de unidades curriculares, assim como a obtenção do horário e calendário personalizados em conformidade com as subscrições.

O i-on Web apresenta dois modos de operação: o modo integrado, em que consome os dados disponibilizados pelo i-on Core, através da respetiva API; e um modo '*standalone'*, para testes, em que funciona somente com dados fictícios internos. A aplicação encontra-se disponível para execução local (modo de desenvolvimento), está em demonstração (modo *standalone* de teste) na plataforma Heroku e está pronta para ser colocada em produção.

Agradecimentos

Em primeiro lugar, gostaria de agradecer aos professores João Trindade e Luís Falcão por toda a disponibilidade e apoio demonstrados ao longo do nosso projeto. Um especial agradecimento ao professor João Trindade, por nos ter dado a oportunidade de realizar este projeto e de fazer parte da iniciativa i-on.

A todos os membros da iniciativa, docentes e alunos, obrigado pelo seu acolhimento e espírito de trabalho em equipa.

Agradeço ao meu colega Ricardo pelo seu companheirismo, trabalho em equipa, otimismo, empenho e pela disponibilidade para auxiliar e partilhar o seu conhecimento comigo ao longo de todos estes semestres, muito obrigada por todas as memórias.

Os meus agradecimentos também a todos os colegas e professores com quem interagi no decorrer do curso, que me acolheram demonstrando sempre um espírito de entreajuda e cooperação.

Por fim, agradeço especialmente à minha família sempre presente e fornecendo um apoio e compreensão incondicionais mesmo nos momentos mais difíceis e por me darem as condições necessárias para a realização desta licenciatura.

A todos aqueles que foram mencionados, muito obrigado.

Julho 2021, Catarina Palma

Os meus primeiros agradecimentos são direcionados aos orientadores João Trindade e Luís Falcão, agradeço imenso pelo apoio e ajuda ao longo desta jornada tão importante que é a realização do projeto final do curso. Ao professor João Trindade dou um especial obrigado por nos ter dado a oportunidade de desenvolver o i-on Web, algo que me deixou, e ainda deixa muito feliz e lisonjeado.

Aproveito também para agradecer a todos os membros da iniciativa i-on, por nos terem recebido e dado o apoio que precisámos ao longo do desenvolvimento do projeto, assim como agradecer pelas memórias e experiências que tive tanto com colegas como com docentes ao longo da licenciatura.

Um grande obrigado à minha amiga e colega de grupo, Catarina. Por ter desenvolvido o projeto i-on Web comigo, como pelos últimos anos de trabalho em conjunto. A sua companhia, apoio e paciência para comigo ao longo deste tempo, têm um valor que não consigo estimar, o meu mais sincero obrigado.

Por último, agradeço muitíssimo à minha família por, desde o primeiro dia, acreditarem em mim e me apoiarem, incondicionalmente, em todos os momentos e ocasiões ao longo desta longa e árdua jornada. Sem dúvida alguma, um apoio imprescindível para a conclusão da licenciatura.

Pelas memórias, muito obrigado a todos!

Julho 2021, Ricardo Severino

Índice

Índice de Ilustrações

Capítulo 1

Introdução

No decorrer deste capítulo introdutório é realizada uma breve descrição da iniciativa i-on bem como do projeto i-on Web. Será também apresentada a estrutura do relatório relativamente ao conteúdo e ordem dos capítulos que o constituem.

1.1 O que é a iniciativa i-on?

A iniciativa i-on surge da necessidade de melhorar e facilitar a forma em que é divulgada informação académica – por exemplo, no ISEL – sendo que, atualmente, esta não é difundida de modo a permitir processamento complementar que possibilite a construção de ferramentas adicionais às disponibilizadas pelas instituições.

Deste modo, a iniciativa i-on desenvolve um conjunto de projetos para apoiar as atividades académicas, estando atualmente com o foco na agregação de informações de calendário e de estrutura curricular, com vista à sua redistribuição em formatos computacionalmente convenientes.

Esta iniciativa, introduzida no ano letivo 2019/2020, visa aprimorar cada um dos projetos que a constituem a cada ano letivo. É ainda de referir que o i-on é *open-source*, permitindo a qualquer um estudar e contribuir para o *software* desenvolvido.

1.2 O projeto i-on Web

O i-on Web, a desenvolver de raiz a partir deste ano letivo, consiste numa aplicação web para disponibilizar uma interface para web browser presente em qualquer dispositivo (smartphones, tablets, portáteis, entre outros) para apresentação de recursos académicos.

1.2.1 Requisitos funcionais

- Exposição de informação geral sobre múltiplos cursos, nomeadamente:
    - Uma breve introdução sobre os mesmos;
    - O plano curricular, com informação genérica relativamente a cada uma das unidades curriculares;

- Autenticação dos utilizadores através do email institucional;
- Possibilidade de o utilizador obter a oferta letiva de determinado curso num dado semestre, e por consequência a possibilidade de subscrição às turmas disponíveis numa dada unidade curricular;
- Apresentar ao utilizador a lista de turmas às quais se encontra subscrito, bem como, a opção de editar a mesma (subscrevendo ou anulando a subscrição a determinada turma);
- Exposição do horário do utilizador em conformidade com as turmas a que este se encontra subscrito assim como a opção de transferir o mesmo (em formato *pdf*);
- Apresentar na página inicial lembretes relativos aos próximos eventos (exames, trabalhos, entre outros) das turmas a que este se encontra subscrito;
- Exposição de um calendário com os eventos das disciplinas a que o utilizador se encontra subscrito assim como informação relativa ao calendário escolar;
- Exposição de um perfil do utilizador permitindo ao mesmo editá-lo.

1.2.2 Requisitos não funcionais

- Testes unitários, existindo um processo de *delivery* automatizado;
- Testes de integração;
- *Deploy* automatizado da aplicação para o ambiente de *staging* e produção (*Continuous Deployment).*

1.3 Motivação

Como alunos da Licenciatura em Engenharia Informática, Redes e Telecomunicações, o teor e objetivos de cada projeto presente na lista de opções para projeto final de curso pode variar imenso como consequência das diferentes áreas estudadas ao longo do curso, tais como informática, redes de computadores e telecomunicações. No nosso caso em particular, ao longo do decorrer do curso sempre existiu uma maior inclinação para a área de informática. Sendo esta a nossa preferência, vimos no projeto i-on Web um projeto candidato a satisfazer o nosso gosto pessoal numa etapa tão importante como o término da licenciatura.

Adicionalmente, consideramos que a finalidade do projeto e o que este propõe seja de grande utilidade para a vida académica e algo que gostaríamos de ter tido disponível no decorrer do curso.

Por último, outro fator que pesou na escolha pelo projeto i-on Web foi o cariz “profissional” que apresenta, no sentido, de fazer parte de uma iniciativa que engloba outros projetos sendo um contexto semelhante ao que se poderá ver num ambiente empresarial.

1.4 Estrutura do relatório

Ao longo do relatório final tem-se como objetivo primordial descrever o desenvolvimento do projeto i-on Web até ao seu estado atual. Entre outros aspetos, serão referidas e justificadas as decisões de implementação e as tecnologias utilizadas. O trabalho desenvolvido ao longo deste semestre letivo será então abordado na seguinte ordem:

- No capítulo 2 será dada uma introdução às tecnologias utilizadas, salientando as características das mesmas e o porquê de se ter optado por estas;
- No capítulo 3 é descrita a arquitetura da solução, nomeadamente, a arquitetura a nível da iniciativa i-on, assim como a arquitetura interna do i-on Web e as estratégias de *deploy*;
- No capítulo 4 apresenta-se em maior detalhe a lógica de implementação do servidor i-on Web;
- O capítulo 5 descreve em maior detalhe as estratégias de deploy assim como assuntos relacionados (por exemplo, *dockerfiles* e metodologia de trabalho);
- No capítulo 6 é descrito o processo de autenticação e a lógica associada ao mesmo;
- O capítulo 7 é dedicado à explicação da implementação de cache no projeto;
- O capítulo 8 aborda o fluxo de navegação da *user interface*, complementado com alguns exemplos das páginas desenvolvidas. Por último, também são referidos alguns aspetos de implementação relevantes;
- Finalmente, no capítulo 9 serão expostas as conclusões, adversidades encontradas e trabalho futuro.

Nota: O link para o repositório GitHub onde se encontra o *dossier* de projeto [1] encontra-se no capítulo de referências.

Capítulo 2

Tecnologias utilizadas

Este capítulo irá abordar as tecnologias utilizadas ao longo da realização do projeto referindo, algumas das suas características e detalhes mais relevantes de funcionamento assim como a sua relevância no desenvolvimento do projeto.

2.1 GitHub

Assim como todos os projetos que a iniciativa i-on integra, o projeto i-on Web utiliza o sistema de controlo de versões **Git** e repositórios disponibilizados pelo serviço GitHub proporcionando a capacidade de preservar o histórico das alterações efetuadas, assim como, a possibilidade de reverter para uma versão prévia.

O GitHub proporciona mecanismos para controlo e *merge* (fusão) de versões no desenvolvimento de *software.* Esta característica é fundamental uma vez que o intuito da iniciativa i-on é o continuo aprimoramento de cada um dos projetos que a constituem, dito isto, e sendo esta uma iniciativa *open-source* (permitindo qualquer um contribuir) é evidente a importância de uma ferramenta como o GitHub. Mais especificamente, usufruiu-se das seguintes *features*.

- ***Issues* –** A abertura de um *issue* permite a recolha de *feedback*, reportar a existência de *bugs*, assim como organizar tarefas que se pretende realizar. Desta maneira, é possível que todos os problemas e avanços da aplicação fiquem documentados. A um *issue* atribui-se um nome e é possível escrever comentários sobre o próprio, adicionalmente, é possível associar-lhe várias *labels*, um *milestone* e um *state*, de forma que, qualquer pessoa, mesmo que não esteja familiarizada com o contexto da tarefa, consiga perceber rapidamente o problema em questão, a data que se espera que esteja resolvido e o estado em que o mesmo se encontra.

- ***Branches* e *Pull Requests* –** *Branches* é um mecanismo usado para isolar/desviar o trabalho a desenvolver dos restantes *branches* (ramos) existentes no repositório, isto é, sempre que se pretenda desenvolver novas *features*, corrigir *bugs* ou testar novas ideias, as *branches* garantem uma área dentro do repositório, porém, contida e controlada para o efeito.

  Em concreto, no projeto i-on Web, criou-se um *branch* que se nomeou “*staging*” e que é utilizado como ambiente de ensaio antes da aplicação ir para o sistema em produção, ou seja, qualquer alteração passará sempre primeiro pelo *branch* de *staging* antes de ir para o *main branch* (*branch* principal coincidente com a versão

da aplicação que se encontra em produção). Esta separação é importante para a correta validação do código ainda na fase de *staging*, permitindo assim que a versão em produção fique sempre consistente e disponível.

Quando se considerar que o código desenvolvido num determinado *branch* já cumpre o propósito para o qual o mesmo foi criado, poder-se-á contribuir com as nossas alterações fazendo um *pull request*. Se este for aceite, as alterações realizadas serão integradas no ramo base.

- **GitHub *actions* –** A mecanismos como *pull requests* e *pushs* pode-se associar "*actions*" que executam código sobre o trabalho desenvolvido. Esta ferramenta é utilizada para a execução de testes unitários para que o código realizado seja validado, mantendo a integridade do projeto intacta. É ainda utilizado para a realização de *deploy* para a disponibilização do projeto online.

  As ações podem ser configuradas em ficheiros dentro da pasta "**.github/workflows**".

A partir deste conjunto de ferramentas acredita-se que o desenvolvimento da aplicação se encontra bem documentado, de maneira a ser mais conveniente a alguém que queira estudar a evolução desta ao longo do tempo. Adicionalmente, qualquer um pode facilmente contribuir através de mecanismos como *issues* e *pull requests*.

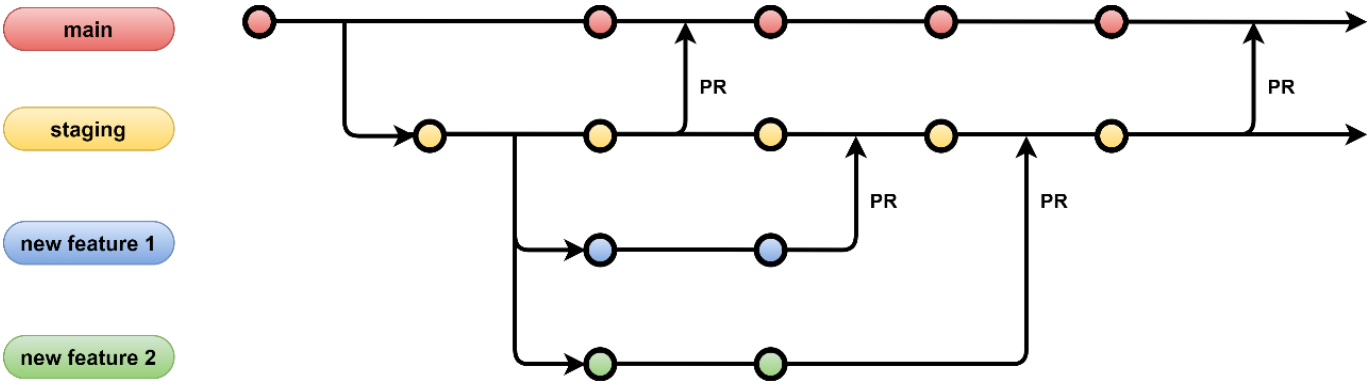

Figura 1 - Esquema referente ao mecanismo de branches e pull requests adotado no i-on Web.

2.2 Containers e anfitrião Docker

Utilizou-se *containers* em conjunto com a tecnologia *docker* com o objetivo de contornar dificuldades que se tem num modelo típico de desenvolvimento de *software*, tais como:

- ➔ Repetitiva instalação dos vários componentes de *software* para executar o código fonte;
- ➔ Necessidade de controlo de versões em todos os componentes instalados;
- ➔ Possível interferência entre processos da mesma máquina.

No decorrer do desenvolvimento do projeto utilizou-se ***containers*** como técnica de criação de isolamento (apesar de não ser completo) entre aplicações dentro da mesma máquina, permitindo, criar e executar ambientes de execução reproduzíveis que são definidos por **imagens** (garantem a capacidade de empacotar tudo o que é necessário para a execução da aplicação).

Foi utilizado o **Docker** como concretização destes conceitos, por exemplo, utilizou-se *dockerfiles* como mecanismo para criação de imagens para, a partir destas, instanciar contentores. Por outras palavras, o Docker age como um 'anfitrião' de contentores usando o *kernel* do Linux para criar um nível de encapsulamento forte permitindo a segregação de processos, podendo estes ser executados de forma independente, garantindo então a capacidade de empacotar e executar uma aplicação num ambiente que aparenta ser um sistema completamente isolado (contentor) apesar de usar o mesmo *kernel* de sistema operativo e os mesmos dispositivos de *hardware*.

Em complemento aos ficheiros Docker já mencionados (dockerfiles), também se utilizou a ferramenta **Docker Compose** para definir e executar múltiplos *containers*.

Concluído, qualquer sistema que suporte Docker pode construir e executar a aplicação i-on Web, obtendo-se a vantagem de a versão do *software* estar sob nosso controlo. Adicionalmente, considera-se, que as características de um container e da tecnologia Docker foram essenciais para o *deploy* da aplicação.

WSL2 (Windows Subsystem for Linux 2)

De notar que foi possível utilizar o Docker em máquinas Windows uma vez que estas oferecem suporte para Linux nativo a partir do WSL2, que utiliza um *kernel* Linux, permitindo desenvolver e ter um ambiente de execução para aplicações Linux numa máquina Windows.

2.3 Elasticsearch

O projeto i-on Web apesar de não necessitar de armazenamento de recursos uma vez que os consome das APIs do i-on Core, optou-se por utilizar uma base de dados para armazenar informações de sessão.

Optou-se pela ElasticSearch uma vez que já se havia tirado partido da mesma na unidade curricular de 'Programação na Internet', existindo já uma familiarização com a mesma por parte dos alunos. Adicionalmente, esta indexa documentos (base de dados noSQL) no formato JSON via HTTP, disponibilizando uma API HTTP que segue os princípios REST o que vai de encontro com as necessidades da aplicação.

Em síntese, irá utilizar-se a ElasticSearch para armazenar dados que são próprios da aplicação e cujo armazenamento em memória persistente é necessário, sendo que, de momento, tal só se aplica a dados de sessão. Esta escolha abre portas para que, futuramente, caso seja benéfico, se possa armazenar persistentemente outros dados que sejam necessários para a aplicação.

2.4 Heroku

Com o objetivo primário de ter o sistema em produção numa máquina disponibilizada pelo ISEL, optou-se pela plataforma Heroku para disponibilizar online a versão de *staging* do i-on Web. É de referir que se está a usufruir de uma versão gratuita existindo algumas limitações, como por exemplo, no número de acessos diários.

Um ponto de grande importância e que contribuiu para a escolha desta plataforma é facto do Heroku ser um ambiente/serviço de execução em *cloud* que suporta o *deploy* de Dockerfile e/ou Docker Compose o que é ideal para o caso particular da aplicação i-on Web.

Outro fator de certa relevância na escolha pela plataforma Heroku foi o facto de esta disponibilizar *add-ons (*ferramentas e serviços de desenvolvimento) o que possibilitou a utilização do *add-on* Bonsai ElasticSearch para servir, como já referido, como base de dados da aplicação para o armazenamento de dados de sessão. De notar que para o uso desta tira-se partido da versão gratuita (*sandbox*) do Bonsai Elasticsearch [2] o que por consequência apresenta algumas limitações (número de pedidos diários, capacidade, entre outros).

Nota: Como será detalhado mais à frente, tirou-se partido do GitHub *actions* para automatizar a tarefa de realizar o *deploy* da aplicação para o Heroku, tirando-se partido do Docker Compose e Dockerfile.

2.5 Bootstrap

Utilizou-se a *framework* Bootstrap de modo a desenvolver o *design* da aplicação, de forma apelativa e responsiva, dos vários ecrãs da aplicação, isto é, a aparência da aplicação web é alterada dinamicamente consoante a orientação e tamanho do ecrã do utilizador. Optou-se por este tipo de desenho uma vez que irá possibilitar um utilizador, em qualquer tipo de dispositivo, aceder ao i-on Web e consultar as suas informações académicas, quer esteja num portátil ou num *smartphone* (procurou-se não condicionar a experiência de utilizador nesse aspeto).

Capítulo 3

Arquitetura

Ao longo deste capítulo apresentar-se-á a estrutura do sistema i-on assim como indicar o enquadramento do projeto i-on Web no mesmo, mais especificamente, irão ser descritas as ligações e dependências existentes assim como abordar de forma genérica a estrutura interna do projeto i-on Web e as suas estratégias de *deploy*.

3.1 Estrutura do sistema i-on e enquadramento do i-on Web

A estrutura do sistema i-on assim como as respetivas interações entre os projetos envolvidos pode ser representada através do seguinte esquema.

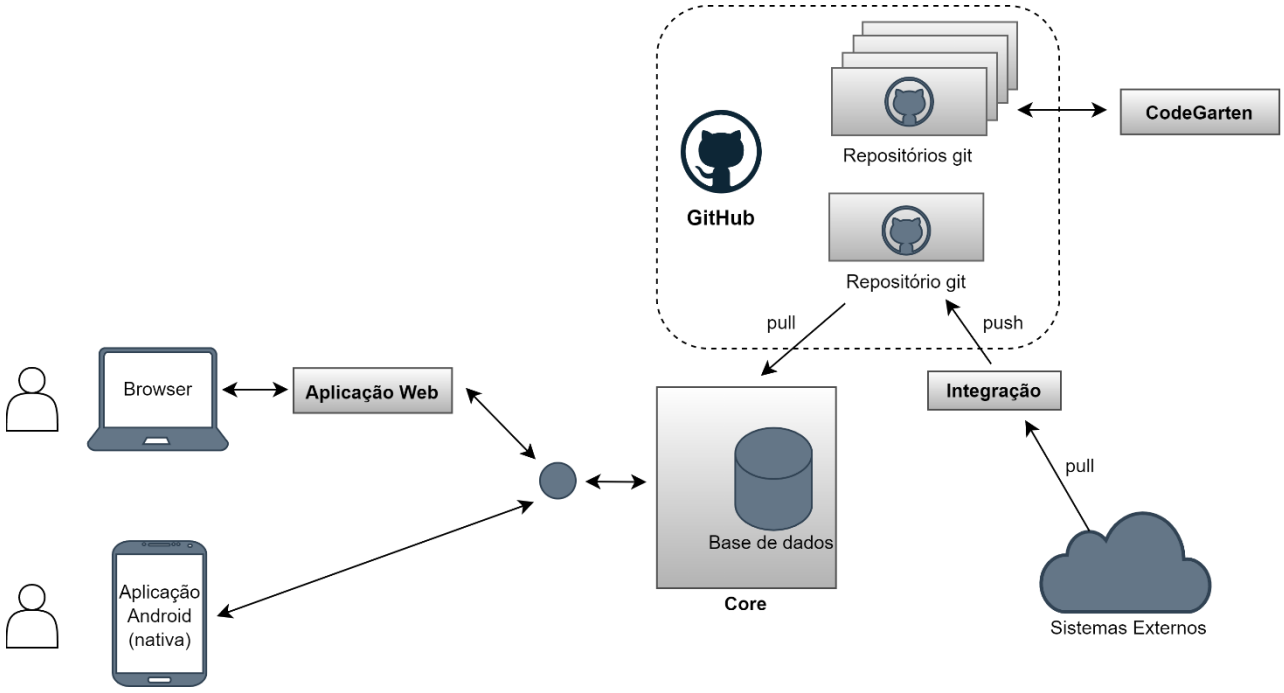

Figura 2 - Arquitetura da iniciativa i-on.

De um modo geral, todos os projetos constituintes da iniciativa encontram-se interligados. Particularizando para o caso das aplicações clientes do i-on (**Web** e **Android**), estas dependem do projeto i-on **Core**. Este fornece uma interface

programática que permite as aplicações cliente obterem recursos académicos e autenticarem os utilizadores da instituição (*read* e *user* API, respetivamente). O **i-on Integration**, por sua vez, obtém informação académica através de sistemas externos e distribui essa mesma informação para o i-on Core por meio de repositórios git.

Por último, o projeto **Codegarten** ambiciona permitir a gestão de repositórios de alunos assim como prover uma interface para entrega de trabalhos, de momento, não está integrado com os restantes projetos da iniciativa i-on.

3.2 Arquitetura interna do i-on web

Tendo sido fornecida uma visão geral dos projetos da iniciativa i-on e como estes estão relacionados, irá agora ser mencionado a arquitetura interna básica do projeto i-on Web.

Com o objetivo de organizar os módulos que constituem a aplicação e a distribuí-los em conformidade com a sua função na mesma, a aplicação encontra-se subdivida nas seguintes camadas lógicas:

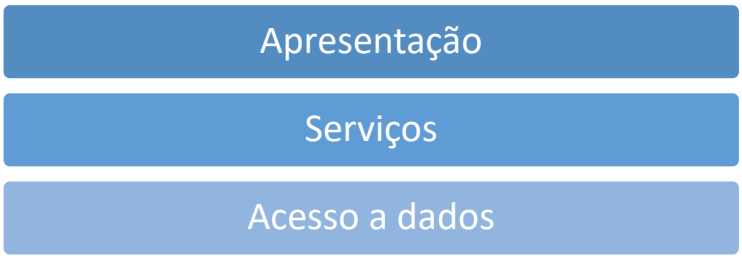

Figura 3 - Camadas arquiteturais.

Apresentação
Responsável por atender e responder aos vários pedidos que chegam à aplicação, conhece e sabe lidar com o protocolo de comunicação (HTTP), abstraindo os restantes módulos deste. Isto, pois, extrai a informação e argumentos necessários dos pedidos e, consoante os *endpoints* dos mesmos, encaminha-os para os respetivos serviços, obtendo o resultado de processamento.

Serviços
Incorpora a lógica da aplicação necessária para os serviços disponibilizados. Nesta camada as informações obtidas na camada de acesso a dados e de entrada são processadas.

Acesso a dados
Camada composta pelos elementos que realizam o acesso aos dados, podendo estes, particularizando para o projeto i-on Web, ser requisitados através de uma API, encontrarem-se armazenados numa base de dados ou ser lidos de ficheiros.

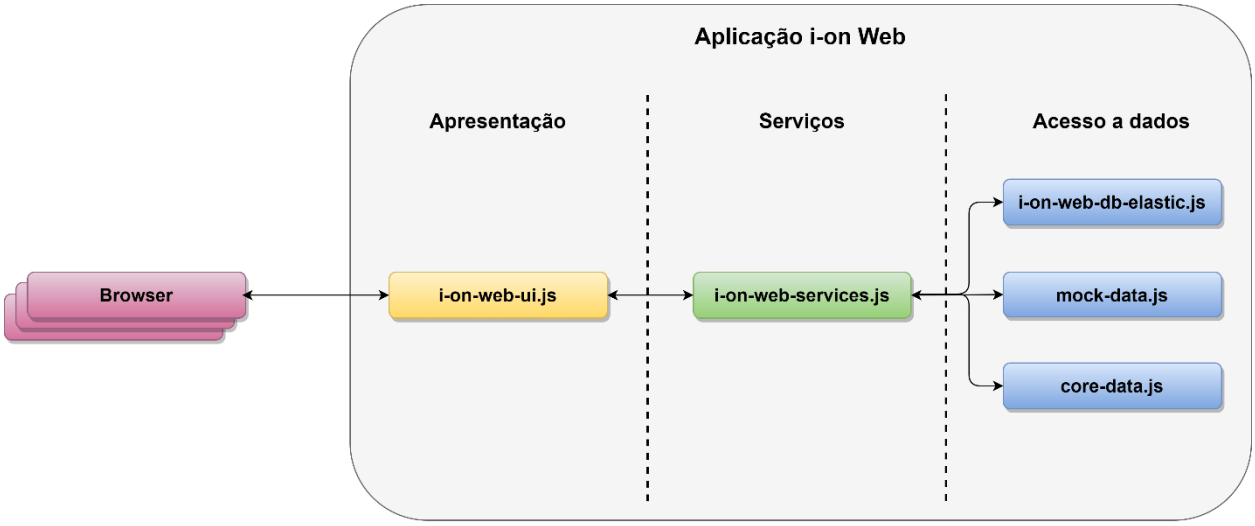

Figura 4 - Arquitetura interna base do i-on Web.

A arquitetura acima (fig.4) demonstra a lógica base da aplicação, mais em concreto, como alguns dos módulos interagem entre si, baseando-se no princípio de três camadas anteriormente descrito.

A aplicação demonstra o seguinte comportamento:

1. Após o envio de um pedido (por exemplo, através de um *browser*) este é atendido por um módulo da camada de apresentação e.g., i-on-web-ui.js;
2. O módulo da camada de apresentação, consoante o *endpoint*, pode extrair informação do mesmo (e.g., identificador do curso que o utilizador deseja que lhe seja apresentado) e chama o serviço responsável por processar o pedido;
3. O serviço processa o pedido, podendo, ou não efetuar validações (por exemplo, verificar se um identificador de curso cumpre certos requisitos mínimos como ser um número inteiro positivo).

   Para obtenção de dados, consoante o pedido, o serviço realiza pedidos a um dos três módulos da camada de acesso a dados, sendo estes:

   - core-data.js – Obtém dados do i-on Core;
   - mock-data.js – Obtém dados fictícios;
   - i-on-web-db-elastic.js – Obtém dados de sessão da *Elasticsearch*;
4. Após obtenção dos recursos, os módulos de acesso a dados retornam os mesmos. O módulo de serviço, após processamento dos dados obtidos, responde ao módulo de apresentação. Este, por sua vez, responde ao pedido HTTP feito inicialmente.

Nota: É de referir que a versão da arquitetura da figura 4 está simplificada, uma versão mais fiel e completa será ilustrada no próximo capítulo. Dito isto, a lógica acima apresentada (baseada no conceito de 3 camadas lógicas) estará sempre presente no fluxo arquitetural da aplicação.

3.3 Estratégias de deployment

Ao longo desta secção irá ser dada uma introdução às estratégias de *deployment* existentes, nomeadamente, quais os ambientes e tecnologias envolvidos no processo.

O projeto, no seu estado atual, encontra-se apto para ser *deployed* de três formas.

- Localmente;
- Para a plataforma Heroku (*staging*);
- Para a máquina hospedeira (produção).

Todas estas tiram partido de ficheiros *dockerfile* e *docker-compose* (abordados mais à frente) no decorrer do seu processo.

Deploy local

De modo a executar a aplicação localmente é necessária a execução de duas aplicações *multi-container*.

- Uma referente ao projeto i-on Web, sendo constituída por dois serviços, sendo estes a aplicação web e a Elastisearch.
- Outra referente ao projeto i-on Core.

Para a execução de ambas as aplicações é possível consultar a documentação referente no repositório GitHub de cada um dos projetos, respetivamente, no repositório GitHub do i-on Web [1] e do i-on Core [3].

Deploy para a plataforma Heroku (*staging*)

Para demonstração e disponibilização do projeto i-on Web online tirou-se partido da plataforma Heroku assim como do GitHub *actions* com o intuito de possuir um *deploy* automatizado a cada *push* / *pull request* para os *branches* especificadas.

Deploy para a máquina hospedeira (produção)

De momento a aplicação encontra-se com todas as parametrizações necessárias para ir para um ambiente de produção (máquina hospedeira), por exemplo, através da publicação da sua imagem no GitHub *container registry*, que permite armazenar e gerir imagens Docker. Esta publicação, à semelhança do *deploy* para o Heroku, também pode ser automatizada com o uso de GitHub *actions*.

Capítulo 4

Servidor i-on Web

Este capítulo complementa a introdução realizada no capítulo prévio, relativamente à arquitetura do projeto, pretendendo-se agora apresentar em maior detalhe a lógica de implementação da aplicação.

O projeto i-on Web encontra-se dividido em módulos, tendo cada um a sua respetiva função. Tendo-se optado por seguir sempre que possível o princípio de responsabilidade única (*single principle responsability principle*), atribuiu-se a cada módulo uma única tarefa. Com o cumprimento deste princípio os módulos da aplicação web:

- Tornaram-se mais fáceis de interpretar, uma vez que cada um tem apenas uma finalidade;
- Foi mais fácil manter eventuais alterações isoladas (manutenção dos módulos foi facilitada).

O esquema presente na figura 5 demonstra a arquitetura da aplicação em maior detalhe relativamente à figura 4.

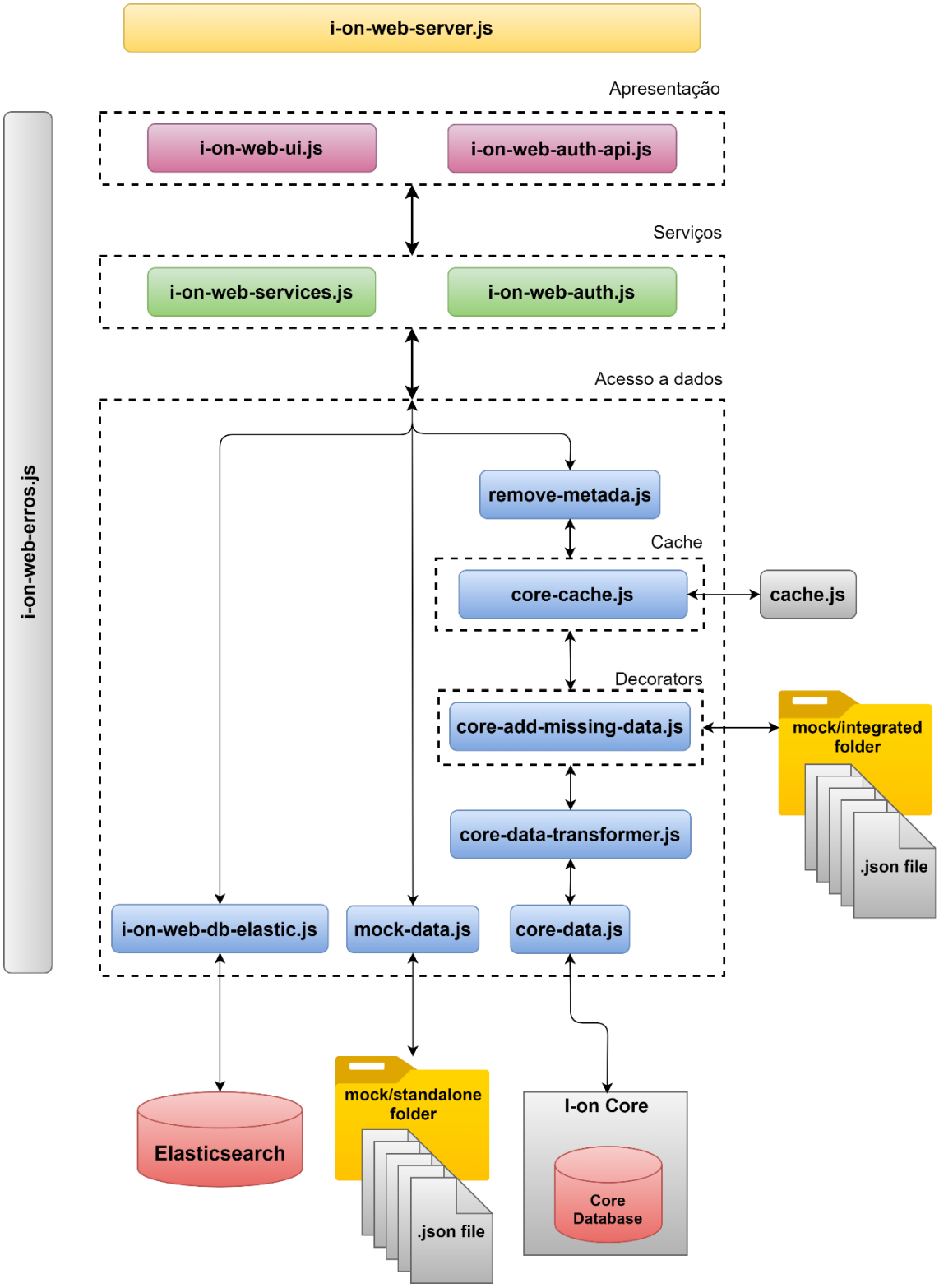

Figura 5 - Arquitetura da aplicação i-on Web.

Em seguida será pormenorizada a função atribuída a cada um dos módulos acima representados.

- **i-on-web-server**

  Ficheiro que constitui o ponto de entrada na aplicação servidora, contendo as configurações necessárias ao funcionamento da aplicação, nomeadamente, fornecendo as devidas dependências (injeção de dependências) aos módulos constituintes da aplicação de modo a diminuir o nível de acoplamento entre os mesmos.

  Mais especificamente, é utilizada a funcionalidade '*require*' sobre ficheiros/módulos da aplicação, e à função retornada é-lhe fornecida os argumentos necessários para o correto funcionamento da aplicação. É ainda neste módulo que a partir da *framework* Express se definem *middlewares* utilizando, por exemplo, objetos de roteamento (*Router*). Essencialmente, é responsável por inicializar a aplicação colocando a mesma á escuta.

- **i-on-web-errors**

  Ficheiro que contém a listagem dos possíveis erros da aplicação e são estes que circulam na mesma para que esta não esteja dependente do protocolo de comunicação. Por exemplo, no acesso a dados existe uma tradução dos erros para os erros internos da aplicação que por sua vez serão novamente traduzidos nos módulos de apresentação para os erros do protocolo de comunicação (no caso HTTP).

4.1 Módulos de acesso a dados

Acesso ao Core

- **core-data**

Módulo que realiza os pedidos às APIs do i-on Core, traduzindo os possíveis erros HTTP recebidos para erros da aplicação. A natureza destes pedidos pode ser colocada em três categorias:

  - Obtenção de recursos académicos, e.g., informação sobre o calendário escolar, cursos, entre outros;
  - Autenticação dos utilizadores da aplicação;
  - Gestão das subscrições e perfil dos utilizadores.

O *media type* retornado nas respostas do Core é application/vnd.siren+json enquanto o *media type* para o envio de informação é *application/json*.

- **core-data-transformer**

Este módulo é responsável por transformar os recursos recebidos do Core para o formato esperado (modelos de dados do domínio) pelos módulos de serviços, de modo a estes serem mais facilmente processados. Este aspeto de transformação foi ainda de maior relevância visto que o *media type* retornado pelo Core não nos é o mais conveniente.

Em termos de estrutura optou-se pela colocação deste módulo o mais próximo possível da fonte de dados (‘core-data’) para que caso no futuro ocorra alguma mudança no formato dos dados recebidos da fonte, o número de módulos da aplicação a serem alterados, seguindo esta arquitetura, seja minimizado.

- **core-add-missing-data**

Muita informação necessária para o funcionamento da aplicação web (como planos curriculares, informação sobre os cursos, entre outros) ainda não se encontra disponível ou encontra-se incompleta. Portanto, o módulo ‘core-add-missing-data’ colmata (com informação proveniente de ficheiros json) a ausência ou inconsistência de alguns dados do i-on Core.

Este módulo trata-se de um *decorator*, sendo que quando a informação for obtida como pretendido, este poderá ser retirado sem qualquer efeito para os restantes módulos da aplicação.

- **core-cache**

Implementa a lógica de cache da aplicação, recebendo os metadados e dados provenientes dos módulos de acesso a dados anteriores. Irá realizar o cache de apenas alguns dados (os que fazem sentido no contexto da aplicação) sendo transparente para os restantes pedidos que não usufruam da mesma.

- **cache**

Classe utilizada no módulo anterior, apresenta uma interface com os métodos que se considerou relevantes para a implementação do sistema de cache, nestes são utilizados os métodos disponibilizados pelo node-cache.

- **remove-metadata**

Realiza a remoção dos metadados, de forma que apenas sejam enviados dados para os módulos de serviços, isolando-o assim de toda a lógica associada aos mesmos, nomeadamente, a lógica de cache.

Nota: A implementação da cache e a lógica associada aos metadados serão descritas em maior detalhe no capítulo 7.

Acesso à Elasticsearch

- **i-on-web-db-elastic**

Responsável pelo armazenamento e acesso a dados na Elasticsearch que irá conter dados de sessão. Os dados são armazenados num índice denominado 'sessions' em que cada documento é referente a uma sessão. Neste (exemplo na figura 6) encontram-se os campos referentes aos *tokens* de sessão recebidos do Core, um campo referente ao *email* do utilizador e um campo 'lastRefreshed' onde se indica em *epoch miliseconds* a data de criação/atualização do documento. Tanto no envio como na receção de dados é utilizado o *media type application/json*.


```
{
    "_index" : "sessions",
    "_type" :  "_doc",
    "_id": "X2RdynoBo7E3yFlgBxZ3",
    "_score": 1.0,
    "_source": {
        "email": "ANNNNN@alunos.isel.pt",
        "lastRefresh": 1626892732254,
        "access_token": "TvXMOR3LqRPcFW1Ujt2hTzph9NdGNSiHvIuSnaIMHzeaSepItmrY7Ls-h6am1EnHCCn3Z_R2Ubgus1UdaUqOvg",
        "token_type": "Bearer",
        "refresh_token": "j8uidn-1Ze9POKXPnggRjfBHixD4V9p39YlUpQ-yR3lzxpsLCM496d7t62pWo-5Frou7fqUZU6-u3mooGWMb8A",
        "expires_in": 3598,
        "id_token": "eyJhbGciOiJIUzI1NiJ9.eyJleHAiOjE2MjY4OTYzMzEsImub3MuaXNlbC5wdCJ9.Vhdm1ChPUJCxZNs3_jtgIEhhpv9no_6Ao59RAarU31g"
    }
}
```


Figura 6 - Exemplo da estrutura de um documento no índice sessions.

Acesso a dados mock

- **mock-data**

Utilizado no modo de operação '*standalone'* em alternativa ao acesso ao i-on Core (os modos de operação serão referidos mais à frente), este realiza o acesso a ficheiros JSON de modo a obter dados fictícios (*mock*).

Este também permite uma lógica para a existência de contas de utilizador (em memória) somente para fins de demonstração. Todos utilizadores são armazenados num objeto que contém como propriedade o email de cada utilizador. Por sua vez, para cada propriedade (associada a um utilizador) existe um objeto no qual é guardado o nome do utilizador e as turmas das unidades curriculares às quais subscreveu.

4.2 Módulos de serviços

- **i-on-web-auth**

Implementa a lógica relativa ao processo de autenticação e à gestão de sessões, sendo estes descritos em maior detalhe no capítulo 6.

- **i-on-web-services**

Módulo que implementa a lógica de cada uma das funcionalidades da aplicação. Adicionalmente é responsável pela validação de erros e tratamento da informação recebida pelo utilizador e enviada para o mesmo.

4.3 Módulos de apresentação

- **i-on-web-ui**

Define, através de um objeto ***Router***, as rotas e respetivos métodos HTTP pelos quais é responsável por receber e responder aos pedidos. Cada *handler* definido neste *router* é responsável por receber um pedido, extrair o corpo e/ou parâmetros do pedido enviando-os de seguida para os serviços de forma que estes pedidos ser processados. Na construção de respostas utiliza-se *templates* Handlebars para gerar HTML para a apresentação da interface do utilizador. Neste módulo é ainda realizada a tradução dos erros da aplicação para erros HTTP.

- **i-on-web-auth-api**

Este módulo é responsável por receber pedidos HTTP relacionados com o processo de autenticação. Em contraste com o módulo anterior, este não renderiza páginas HTML, mas sim envia respostas no formato JSON, uma vez que para este módulo apenas são realizados pedidos por parte de *scripts* que correm no lado do cliente.

Em resumo, a aplicação é constituída pelos módulos descritos anteriormente, sendo que a implementação de alguns destes (e.g., cache e autenticação) será detalhada posteriormente. Adicionalmente, é possível notar que este tipo de arquitetura permitiu uma maior flexibilidade para divisão de tarefas, permitindo, no futuro, adicionar ou alterar pontos de entrada e de acesso a dados da aplicação sem afetação nos serviços suportados.

4.4 Modos de Operação

Com a finalidade de demonstrar o i-on Web sem que esta dependente do i-on Core, procedeu-se à criação de alguns ficheiros json com os dados já no formato pretendido pelos módulos de serviço. Com a existência destes ficheiros *mock* torna-se possível executar a aplicação em dois modos de operação distintos, sendo estes apresentados em seguida.

4.4.1 Modo *Standalone*

Neste modo, o i-on Web não obtém os dados do i-on Core, mas sim dos **ficheiros com dados fictícios (*mock*)** referidos anteriormente, isto é, todos os dados expostos na aplicação quando esta é executada serão dados provenientes de ficheiros json.

Uma vez que não se tira partido do i-on Core, simulou-se uma lógica que permite autenticação de utilizadores na aplicação. As informações relativas a estes como subscrições e informação de perfil encontram-se apenas em memória volátil, não existindo qualquer desafio para a autenticação (como confirmação de email ou password) foi necessário impôr um limite no número de contas existentes de modo a evitar ataques de *denial of service* à aplicação.

Por omissão, a aplicação não se encontra neste modo, assim sendo, deve-se colocar a variável de ambiente OPERATION_MODE com o valor '*standalone*'.

4.4.2 Modo Integrado

Neste modo de operação, a aplicação **obtém informação** e **comunica** com o **i-on Core**. Através deste irá obter-se informação relativa aos recursos académicos assim como guardar/obter informação de utilizador, tirando partido, respetivamente, da *read* API e *user* API.

- A *read* API disponibiliza uma interface programática para a obtenção de informação relativa aos recursos académicos;
- A *user* API expõe uma interface que permite às aplicações clientes autenticarem os seus utilizadores, assim como operações relativas aos mesmos, como a subscrição a turmas de determinadas unidades curriculares, anulação de subscrições, edição do perfil de utilizador, entre outros.

O procedimento para a execução de cada um destes modos assim como outra informação genérica relativa ao projeto, encontram-se documentados no ficheiro README.md do repositório do projeto [1].

A implementação da aplicação não se encontra condicionada/limitada aos modos de operação, sendo que para alternar entre um modo e o outro é somente necessário a alteração de uma variável de ambiente. Tal é possível graças à injeção de dependências realizada pelo módulo 'i-on-web-server' (módulo de entrada da aplicação servidora) que ao diminuir o nível de acoplamento entre os diferentes módulos da aplicação, garante uma maior transparência relativamente aos modos de operação.

Capítulo 5

Deployment

Ao longo desta secção irão ser descritas em maior detalhe as estratégias de *deployment*, já mencionadas anteriormente no capítulo referente à arquitetura. Tal como, a criação dos ficheiros *dockerfile* e *docker-compose* e qual a metodologia de trabalho, sendo estes essenciais para o processo de *deploy*.

5.1 *Dockerfile* e *docker-compose*

Primeiramente, é fundamental a utilização de *containers* para o processo de *deployment*. Para tal, foi necessário criar uma imagem da aplicação que possa ser executada em qualquer máquina/ambiente. Com este objetivo foi criado um ficheiro *dockerfile* a partir do qual é possível construir a imagem, encontrando-se definidas no mesmo as dependências e variáveis de ambiente. As variáveis de ambiente da aplicação encontram-se listadas em seguida:

- **OPERATION_MODE –** Indica qual o modo de operação (tema abordado mais à frente);
- **CORE_READ_TOKEN –** *Token* necessário para a obtenção de informação por parte do Core, nomeadamente, para a *read* API;
- **CORE_URL –** A localização do Core;
- **CORE_CLIENT_ID –** O id de cliente da aplicação;
- **CORE_CLIENT_SECRET –** O *secret* (segredo) de cliente da aplicação;
- **DB_ELASTIC_URL –** Localização da Elasticsearch;

Foram ainda criados dois ficheiros *docker-compose* para criação de aplicações *multi-container*.

1. Um dos ficheiros é utilizado no *deploy* para a plataforma Heroku não contendo por isso nenhuma configuração relativa à Elasticsearch (utiliza-se o *add-on*).
2. O outro ficheiro, deverá ser utilizado nas restantes situações (e.g., para produção e/ou correr localmente) dando origem a uma aplicação *multi-container* com a base de dados Elasticsearch num dos *containers* e a aplicação web noutro.

5.2 Metodologia de trabalho

Como referido anteriormente irá ser utilizada a plataforma GitHub para controlo de versões, nomeadamente, a criação de *branches* para tarefas especificas como uma nova funcionalidade ou resolução de um erro da aplicação. Dos *branches* criados ao longo do semestre, dois sempre persistiram e são de salientar, o *branch* '*staging'* e o *branch* '*main'*. Estes foram utilizadas para manter versões mais completas e estáveis, sendo que o código existente na *branch* de *staging* só seria *merged* com o *branch main* se o código desenvolvido no *branch staging* fosse validado pelos testes existentes. Adicionalmente, como já referido, ao longo do desenvolvimento do projeto outras ferramentas, tais como, *issues* e *pull requests* foram utilizadas para uma melhor documentação e organização de trabalho.

5.3 Teste unitários e de integração

Com o intuito de descrever o processo de *deploy* é importante referir os testes que antecedem o mesmo assim como a sua importância no desenvolvimento da aplicação, sendo estes inerentes à metodologia de trabalho.

5.3.1 Testes unitários

Para validação da lógica da aplicação, mais especificamente, para validação do comportamento das funções no módulo 'i-on-web-services', realizaram-se testes unitários tirando partido da biblioteca Chai que permite efetuar asserções para a validação das respostas obtidas.

Os testes unitários foram de grande relevância permitindo-nos rápida e facilmente encontrar erros, prevenindo que versões da aplicação com erros graves fossem mantidas durante muito tempo, por exemplo, na plataforma Heroku, dado que através do GitHub *actions* além do *deploy* também eram executados os testes unitários permitindo logo a perceção da existência de algum erro relativo à lógica da aplicação.

5.3.2 Testes de integração

Para validação das respostas obtidas pelo cliente, nomeadamente, das páginas HTML assim como do conteúdo das mesmas realizaram-se testes de integração de modo a validar as respostas enviadas pela aplicação i-on web. Para tal, utilizaram-se as seguintes bibliotecas:

- **Frisby** – É uma ferramenta de teste de API que permite realizar pedidos à aplicação assim como realizar asserções sobre as respostas obtidas;
- **Cheerio** – É uma biblioteca de *web scraping* (ou *web data extraction*) baseada em jQuery e permite, a partir de seletores CSS *standard*, a procura de elementos e obtenção do conteúdo dos mesmos, permitindo assim, testar as páginas HTML fornecidas pela aplicação web.

Os testes de integração, apesar de não terem uma execução automatizada como os testes unitários, não deixam de ter a sua importância visto que providenciam uma

maior cobertura de testes ao código realizado (testam a aplicação como um todo), ao invés dos testes unitários que apenas testam a lógica da aplicação. É de salientar que ao contrário dos testes unitários, os testes de integração atualmente encontram-se numa versão muito básica não realizando validações às componentes que seriam de facto interessantes.

5.4 *Deploy* local

O projeto i-on Web deve funcionar tendo como base o i-on Core, através do qual se irá obter informação e para o qual se irá enviar informação de utilizador. Assim sendo, a conectividade entre ambos e o seu correto funcionamento foi um objetivo de extrema importância para o progresso do projeto, de modo que, este foi um dos primeiros a ter em atenção.

Seguindo a documentação já existente no repositório GitHub do i-on Core [3], foi possível colocar a aplicação *multi-container* do Core em execução.

De seguida, utilizou-se o ficheiro *docker-compose* já descrito, de forma a criar uma aplicação *multi-container* constituída pelos serviços referentes à aplicação web e à Elasticsearch.

Nota: O i-on Web e o i-on Core não pertencem à mesma aplicação, logo, não se encontram na mesma rede. Deste modo, utilizou-se o *default gateway* dos contentores Docker (172.17.0.1.) para obter conectividade entre as duas aplicações, tirando partido do *host* como intermediário. É de referir que, tipicamente, o *host* atribui aos contentores Docker endereços pertencentes à gama 172.17.0.0/16.

É ainda de referir que, por omissão, os valores das variáveis de ambiente presentes no ficheiro *docker-compose* são os corretos para a conexão ao Core, no entanto, caso o utilizador (como já referido) o pretenda, no momento da construção da imagem poderá inserir outros valores.

5.5 *Deploy* para a plataforma Heroku (*staging*)

Para demonstração e disponibilização do projeto i-on Web colocou-se o mesmo online tirando partido do Docker Compose, GitHub *actions* e Heroku.

De modo a efetuar o *deploy* da aplicação i-on Web para o Heroku foi necessário incluir no repositório GitHub uma *action* (num ficheiro .yml) para automatizar o processo de *deploy* de modo a manter a versão mais atual online.

Nesta ação é então indicado o ficheiro *docker-compose*, quais os eventos (*push*, *pull request*) e *branches* (*staging*) que desencadeiam a execução da mesma por parte do GitHub Actions. Adicionalmente, este ficheiro contém informações associadas à conta no Heroku (como email e API *key*), estando estas guardadas em GitHub *secrets* ao invés de escritas no próprio ficheiro por motivos de segurança e privacidade.

É ainda de relembrar a utilização do *add-on Bonsai* Elasticsearch na plataforma Heroku de modo a armazenar dados de sessão.

Nota: Uma vez que o projeto i-on Core ainda não se encontra disponível online, o modo de operação utilizado pela aplicação (na plataforma Heroku) é o modo *standalone* com dados fictícios.

O esquema apresentado na figura 7 sintetiza a lógica existente no processo de desenvolvimento do i-on Web. É ainda de referir a existência de *actions* referentes à execução de testes que validam o código desenvolvido.

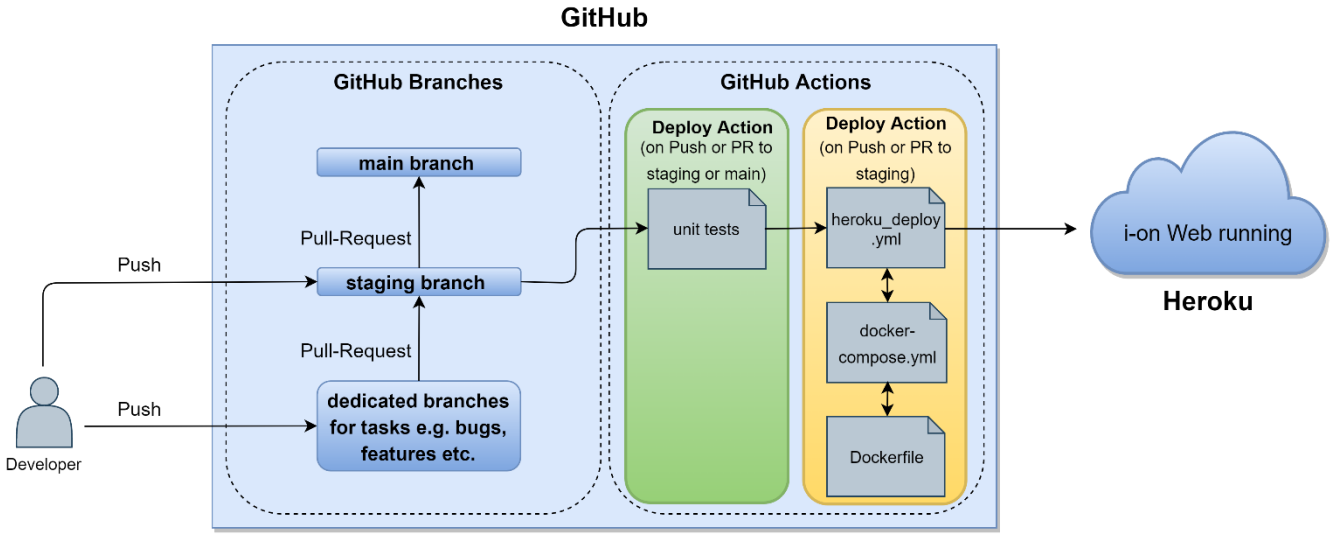

Figura 7 - Fluxo atual das várias componentes intrínsecas ao desenvolvimento do projeto.

É a partir da versão que se encontra na *branch* de '*staging'* que se realiza o *deploy* para o Heroku, neste, como já referido, a aplicação encontra-se a correr em modo *standalone*, para tal, é necessário definir na plataforma o modo *standalone* como modo de operação na variável de ambiente. O valor da variável de ambiente definido na plataforma sobrepõe-se ao valor por *default* (existente no ficheiro *docker-compose*) que a mesma teria, adicionalmente, o valor da variável de ambiente que indica a localização da Elasticsearch é também alterado sendo-lhe atribuído um valor fornecido pelo *add-on Bonsai* Elasticsearch.

5.6 *Deploy* para a máquina hospedeira (produção)

Nesta secção irá ser descrito o *deploy* para a máquina hospedeira, indicando as configurações necessárias para tal.

Dada a construção de imagem anteriormente descrita assim como as variáveis de ambiente existentes (e.g. 'CORE_URL') para parametrização da aplicação, as adaptações para *deploy* de produção necessárias são escassas.

Assim sendo, apenas foi necessário adaptar a aplicação de modo a suportar um prefixo nos seus *endpoints,* tendo sido criada uma variável de ambiente denominada 'PATH_PREFIX'. Esta é necessária devido à possibilidade dos projetos i-on partilharem o mesmo nome de DNS na máquina hospedeira, deste modo, a aplicação deve se encontrar preparada para que lhe seja atribuído um prefixo de forma a distinguir-se dos restantes projetos da iniciativa i-on.

Deste modo, a aplicação encontra-se atualmente em condições para ser *deployed* para um ambiente de produção (máquina hospedeira), por exemplo, através da publicação da sua imagem no *GitHub Container Registry*, que permite armazenar e gerir imagens Docker. Publicação esta, que à semelhança do *deploy* para o Heroku, também pode ser automatizada com o uso de GitHub *actions*.

Capítulo 6

Autenticação

Este capítulo detalha o processo de autenticação presente na aplicação, apresentando e explicando as interações e funções de cada um dos vários intervenientes no processo. Adicionalmente, também será descrita a gestão de sessões do i-on Web.

6.1 Fases do Processo de Autenticação

Primeiramente, irá se falar do processo de autenticação, ou seja, os pedidos e repostas envolvidos desde que o processo de login é solicitado pelo utilizador até este se encontrar autenticado. Este é comum a todas as aplicações cliente do i-on, no entanto, irá particularizar-se para o caso do i-on web, isto pois, o processo de *polling* (que será referido em seguida) poderá diferir de aplicação para aplicação cliente.

O processo de autenticação tem um comportamento inspirado num fluxo de autenticação do protocolo *OpenID connect*, ***Client Initiated Backhchannel Auhtentication Flow (CIBA)*** **Poll Mode**, onde a aplicação cliente (neste caso, i-on Web) inicia o processo de autenticação com o servidor (i-on Core) em nome do utilizador.

É de salientar que o processo de login e registo no Core não possuem qualquer diferenciação, deste modo, também na aplicação web não existe distinção.

O fluxo que se tem atualmente implementado pode ser representado no diagrama da (figura 7).

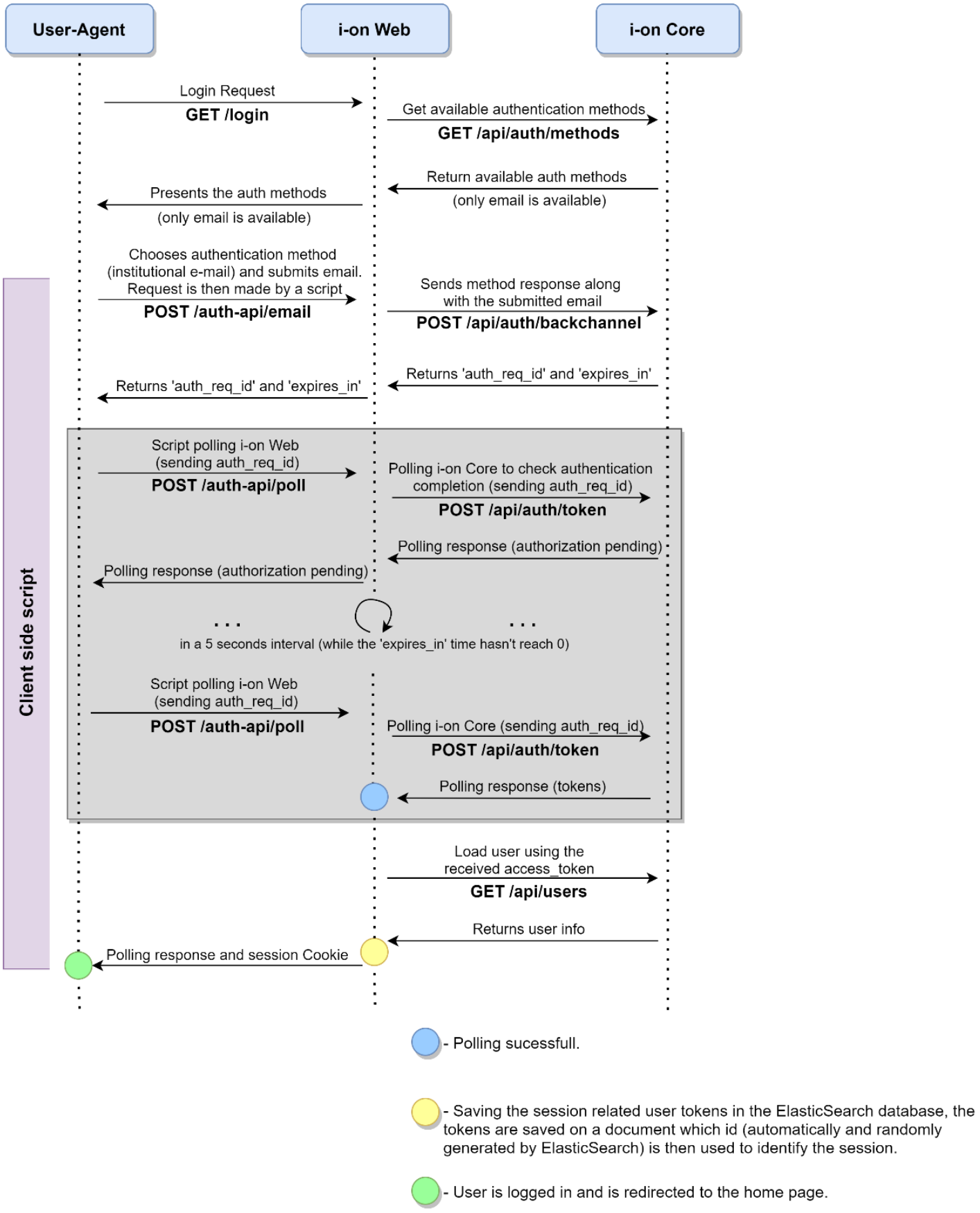

Figura 8 - Processo de autenticação.

Em seguida, detalhar-se-á o processo de autenticação acima esquematizado.

1. O fluxo é iniciado após o utilizador efetuar um pedido de login (*login request*) ao i-on Web, que em seguida faz um pedido ao servidor de modo a adquirir os métodos de autenticação disponíveis.

2. O servidor retorna os métodos de autenticação disponíveis. O único método de autenticação atualmente disponível permite a verificação da identidade do utilizador através, por exemplo, do email institucional correspondente à instituição que o utilizador frequenta.
Na resposta, além do tipo de autenticação também é especificado os domínios permitidos, sendo que neste momento apenas é possível autenticar-se através de email cujo domínio seja "***.isel.pt**" ou "***.isel.ipl.pt**" assim como respetivos subdomínios (e.g., alunos.isel.pt).

3. Já tendo recebido resposta do servidor de autenticação, o i-on Web apresenta os métodos de autenticação ao utilizador. O utilizador escolhe o método de autenticação com o qual deseja autenticar-se (como já referido, neste momento apenas é possível com um email institucional com um os domínios já referidos).

4. Este passo coincide com o início da autenticação entre o i-on Web e o servidor (***Backchannel Authentication***). Após o utilizador ter escolhido o método de autenticação que pretende, o i-on Web faz um pedido ao i-on Core de modo a iniciar o procedimento de autenticação.

O *body* deste pedido contém os seguintes campos:

- **scope** – Particulariza quais os privilégios se pretende ter acesso na utilização do *access_token* ainda a emitir. O i-on Web requere sempre os *scopes* '*openid'*, '*profile*' e '*classes*', o primeiro é obrigatório enquanto o segundo e o terceiro são necessários de modo a gerir, respetivamente, o perfil e informação académica do utilizador;
- **acr_values** – Designa o método de autenticação escolhido (apenas 'email' disponível no momento);
- **login_hint** – O email institucional do utilizador;
- **client_id** – Identificador da aplicação cliente i-on Web;
- **client_secret** – Segredo do i-on Web.

Após este pedido o servidor irá enviar um 'email desafio' ao utilizador, este, de modo a comprovar a sua identidade, terá que aceder à sua conta de email e aceitar o desafio (URL que o redireciona para o servidor) concedendo, ou não, as permissões que o i-on Web solicitou.

5. Uma resposta bem-sucedida terá o identificador do pedido de autenticação (*auth_req_id*) e o tempo para o mesmo expirar (*expires_in*).

6. Para detetar se o pedido de autenticação já foi concluído, o servidor de autenticação tem implementado o modo Poll do protocolo CIBA que, em síntese, atribui à aplicação cliente a responsabilidade de estar sucessivamente a inquirir o servidor (no *token endpoint*) sobre os *tokens* respetivos ao pedido de autenticação, sendo que a receção dos mesmos simboliza que o processo de autenticação foi completado com sucesso.

7. Caso o utilizador tenha cumprido o desafio e o processo de autenticação tenha, de facto, terminado com sucesso, o servidor responde ao pedido de *polling* com os *tokens* que permitam ao i-on Web aceder à API e ter autorização para realizar ações sobre os *scopes* anteriormente descritos.

O i-on Web recebe no *body* da resposta os seguintes campos:

- **access_token** – *Token* para acesso à API, emitido para os *scopes* citados;
- **token_type**– O tipo do *token*, neste caso Bearer;
- **refresh_token** – *Token* para atualizar o *access token*;
- **expires_in** – Segundos para a expiração do *access token*;
- **id_token** – *Token* que contém um conjunto de asserções sobre a identidade do utilizador.

Nota: Caso o utilizador ainda não tenha cumprido o desafio, o processo de *polling* irá continuar (detalhar-se-á mais sobre o processo ainda neste capítulo).

Neste momento o processo de autenticação está terminado e o utilizador encontra-se autenticado no i-on Web.

6.2 Questões relevantes na implementação do processo de autenticação

Descritos os múltiplos fluxos que constituem o procedimento da autenticação, este subcapítulo irá focar-se em detalhes de implementação referentes a este processo no que diz respeito ao i-on Web.

6.2.1 Submissão de email e *polling* ao servidor de autenticação

A lógica de submissão de email assim como o procedimento de, continuamente, averiguar se determinado pedido de autenticação já foi concluído (*polling*) foi implementada tendo como participantes não só a aplicação, mas também o sistema *browser* do utilizador.

Nota: Todos os pedidos referidos na seguinte explicação são atendidos pelo *router* devolvido pelo módulo 'i-on-web-auth-api'. Este, como já referido anteriormente, tem o objetivo de atender e cuidar de todos os pedidos relacionados ao processo de autenticação de um utilizador.

No *template* da página de login, ou seja, onde o utilizador insere o seu email para que se possa autenticar, foi realizada uma função (num *script*) para implementar a lógica de submissão do email e de *polling* à aplicação. Esta função é desencadeada na submissão de um email (após validações de *input*) e o seu comportamento é descrito em seguida:

1. Primeiramente, a função faz um pedido (através da *fetch API* do *browser*) ao i-on Web em que o corpo do pedido tem o email do utilizador.
2. Por sua vez, o i-on Web, envia ao servidor o método de autenticação que o cliente escolheu (email) de forma a iniciar o processo de autenticação e assim obter informações sobre o mesmo (‘auth_req_id’ e ‘expires_in’). Estes, assim que obtidos pelo i-on Web, são retornados para o *browser*.

O *browser* do utilizador ao receber esta resposta dá início ao processo de *polling* ao i-on Web, isto é, não faz o pedido diretamente ao servidor, mas sim à nossa aplicação. Para tal, é feito um *setInterval* que de 5 em 5 segundos, realiza um pedido para o *endpoint* POST /auth-api/poll onde indica no *body* qual o identificador do pedido (*auth_req_id*). O i-on Web por sua vez, indica ao browser se o processo já está, ou não, concluído (enviando o *status code* 200 ou 202, respetivamente).

Se, passado o tempo de vida do pedido de autenticação (‘expires_in’) o processo ainda não foi terminado, o *setInterval* é cancelado, e por consequência o processo de *polling* também.

Nota: Uma vez que o intervalo temporal do processo de *polling* não é especificado pelo servidor, a documentação oficial do OpenID [4] indica que, nestas situações, (valor não é dado pelo servidor) deve ser adotado um valor de 5 segundos.

Esta lógica pode ser realçada pelo seguinte esquema (representação parcial do esquema de autenticação da figura 8).

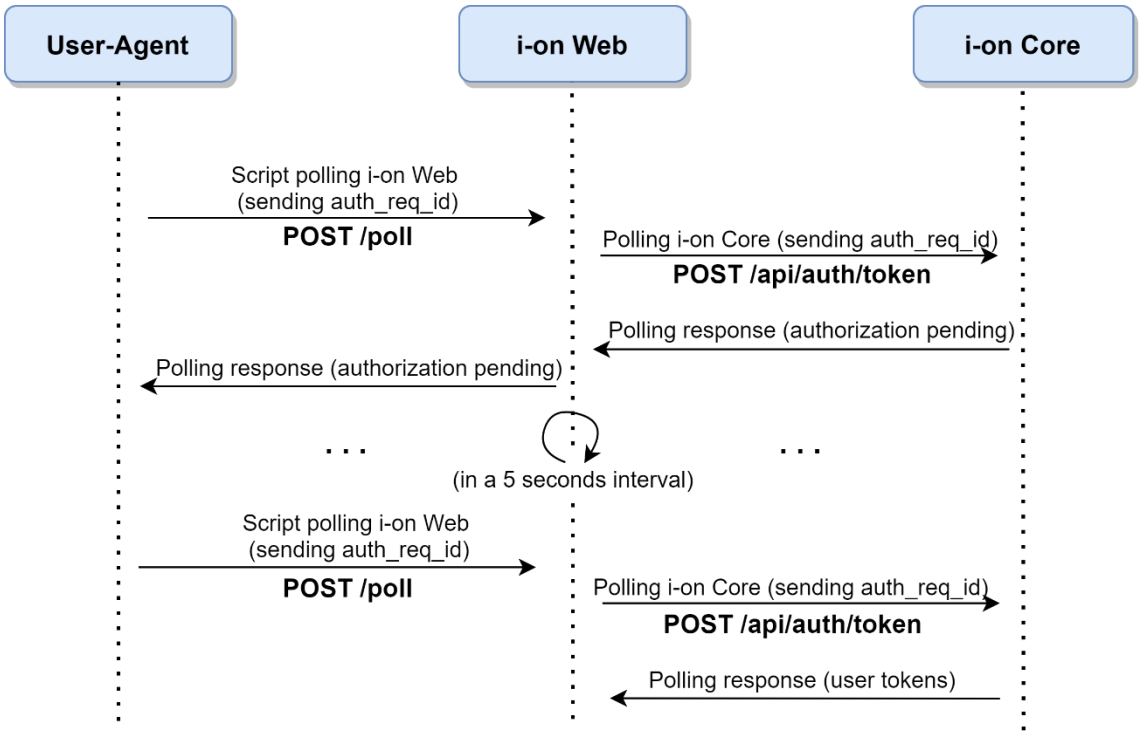

Figura 9 - Processo de polling.

A decisão de realizar o *polling* de 5 em 5 segundos do lado do *browser* utilizador foi, em grande parte, porque se quis libertar o i-on Web da responsabilidade de estar a gerir todos os processos de autenticação. Outra opção de desenho foi o pedido de *polling* passar pelo i-on Web e não ir diretamente do *browser* para o servidor de autenticação, isto pois considerou-se que o *browser* não tem que 'conhecer' o servidor (i-on Core), mas apenas a aplicação Web.

6.2.2 Gestão de sessões

O i-on Web gere sessões através do *middleware* de autenticação *passport* e do *middleware express-session*.

O *passport* tem como finalidade a autenticação de pedidos. Atualmente os processos de autenticação pode tomar muitas formas consoante aa aplicações em causa e o *passport* reconhece isso, podendo ser usado pelas aplicações sem criar dependências desnecessárias. A flexibilidade do uso do *passport* é uma vantagem sendo que, num cenário em que o processo de autenticação do Core mude, o uso de *passport* poderá ser mantido.

Adicionalmente, tendo em conta que o HTTP é *stateless*, é necessária a utilização de *cookies* para manter estado de sessão. Para tal, utilizou-se o *express-session*, nomeadamente, o *middleware session* de modo a gerir as sessões de utilizador, complementando as funcionalidades do *passport* para autenticação.

Como já referido, para a gestão de sessões são utilizadas *cookies* (suporte para manter estado de sessão) mas, adicionalmente, também se tira partido da Elasticsearch para armazenar os *tokens* associados ao utilizador. Sendo os *tokens* correspondentes a cada sessão (quer seja do mesmo utilizador ou não) armazenados em documentos do índice '*sessions'*. Quando um documento é criado para armazenar dados de uma nova sessão, obtém-se o id desse mesmo documento e utiliza-se o próprio para identificar a sessão do utilizador.

O valor deste id estará associado à cookie enviada ao *browser* do utilizador. Assim sendo, quando é recebido um pedido, é possível, a partir do id de sessão obter os *tokens* referentes à sessão do utilizador (***deserializeUser***) para a realização dos pedidos necessários ao Core.

Relativamente aos *tokens*, é de referir que caso o utilizador tente realizar uma operação, mas o *access token* tenha expirado, procede-se à obtenção de um novo *access token* (através do *refresh token*) e à repetição da operação solicitada.

De seguida será descrito o que acontece quando são requisitadas operações que envolvem a eliminação de sessões, nomeadamente quando o utilizador pede para:

- Fazer **logout** – Utiliza-se a operação do *passport* para fazer logout, de seguida pede-se ao i-on Core para revogar o *access token* em questão e por último remove-se o documento referente à sessão da Elasticsearch.
- **Apagar conta** – Utiliza-se a operação do *passport* para fazer logout, de seguida faz-se um pedido ao Core para remover a informação do utilizador e por último apaga-se todas as sessões armazenadas em documentos na Elasticsearch associadas aquele email.

Adicionalmente, supondo o caso em que o utilizador abandone a sessão sem o respetivo logout, os dados de sessão ficariam a ocupar espaço de armazenamento na Elasticsearch. Para prevenir a situação, foi implementado um *script*, que a cada 7 dias, verifica se existem sessões mais antigas do que 7 dias, removendo as mesmas. Tal é possível, pois, quando uma sessão é iniciada ou quando é obtido um novo *access token* é juntamente armazenado/atualizado o campo '*lastRefreshed'* com a data atual. Deste modo, a base de dados encontra-se preparada para lidar com sessões abandonadas.

Capítulo 7

Cache

Neste capítulo irá ser abordada a implementação de cache na aplicação i-on Web, nomeadamente, o porquê da implementação da mesma, assim como as decisões de implementação associadas.

Apesar da implementação de Cache não fazer parte do planeamento inicial, ao longo do desenvolvimento do projeto, notou-se que a **quantidade de pedidos ao Core** e o **tempo de processamento** destes era considerável, por este motivo, a necessidade de realizar *caching* dos recursos tornou-se cada vez mais importante. Nomeadamente, no seu uso para dados que se encontravam constantemente a ser obtidos do Core como planos curriculares e informação sobre cursos que, à semelhança da restante informação académica, são dados que de um modo geral não sofrem alterações frequentemente.

Deste modo, optou-se por implementar uma lógica de cache, sendo possível reduzir o número de pedidos ao Core, o que por consequência, diminui a sobrecarga sobre o mesmo e diminui o tempo de resposta da aplicação web.

É de salientar que de momento o Core não se encontra preparado para lidar com pedidos condicionais nem realiza o envio de informação de *caching* nas suas respostas HTTP, assim sendo, e como se irá ver em seguida, foi necessária a adição de metadados de modo a simular os *headers*, relacionados a *caching*, que seriam supostos receber da fonte de dados.

7.1 Sistema de cache e *headers* utilizados

Nesta secção será abordado o sistema de cache utilizado (node-cache) assim como os *headers* dos quais se decidiu tirar partido para *caching*, justificando o porquê de se ter optado por estes.

7.1.1 Node-cache

Tendo em conta que o tempo para realização do projeto é escasso e que a implementação de cache não fazia parte do plano inicial, pretendeu-se implementar esta de forma **simples** de modo a cumprir a finalidade pretendida que é, reduzir o número de pedidos ao Core e diminuir o tempo de resposta da aplicação. Deste modo,

optou-se pela Node-cache, sendo esta uma cache em memória disponibilizada pelo NodeJS e que dispõe de uma API que se considerou útil para a implementação de cache pretendida.

7.1.2 *Headers*

Não existindo qualquer implementação referente a cache do lado do Core, optaram-se pelos *headers* que se considerou de mais simples utilização e cuja simulação de receção seria a mais fácil. Dos *headers* existentes que podem indicar informação sobre o *caching* dos recursos, optaram-se pelos seguintes:

- ***ETag*** – É um identificador único (opaco para o cliente) gerado pelo servidor que identifica o estado do recurso num dado instante temporal. Sempre que o recurso for modificado, a *tag* é alterada, desta forma, é possível à aplicação web realizar pedidos condicionais, como será explicado mais à frente.

  Em alternativa ou em simultâneo também se poderia ter recorrido ao uso do validador *lastModified*, no entanto, considerou-se utilizar apenas a *ETag* em que, no caso, é uma simples uma comparação de *strings* enquanto o uso do *lastModified* iria envolver a comparação de datas trazendo uma complexidade desnecessária para a implementação pretendida.

- ***Cache-Control* (campo *max-age*)** – *Cache-control* é um *header* usado para especificar diretivas para mecanismos de cache, neste, tirou-se partido do campo '*max-age*' que especifica o tempo máximo (em segundos) que um recurso é considerado "fresco", ou seja, o tempo máximo que este pode ser utilizado. Após esse tempo a representação do recurso em cache é considerada '*stale*' ("não fresca").

Para além destes *headers* existem outros que dão informação de *caching*, assim como outras diretivas no *header* '*Cache-Control*' (referido acima), para especificar os mecanismos de cache, cuja utilização faria sentido para uma implementação de cache mais robusta e completa, por exemplo, diretivas que indicam se determinada representação de um recurso deve ou não ser armazenada (*private*, *public*, entre outras).

7.2 Lógica de implementação da cache

Na secção anterior foi apresentada a cache e *headers* dos quais se tiraram partido, agora, será explicado qual o papel destes na implementação assim como algumas das decisões/opções tomadas relativamente a esta.

Primeiramente, contextualizando a cache na estrutura do projeto tem-se que esta se encontra perto dos módulos de serviços, deste modo, os dados já se encontram armazenados na cache no formato que é esperado pelos serviços, evitando processamento adicional desnecessário.

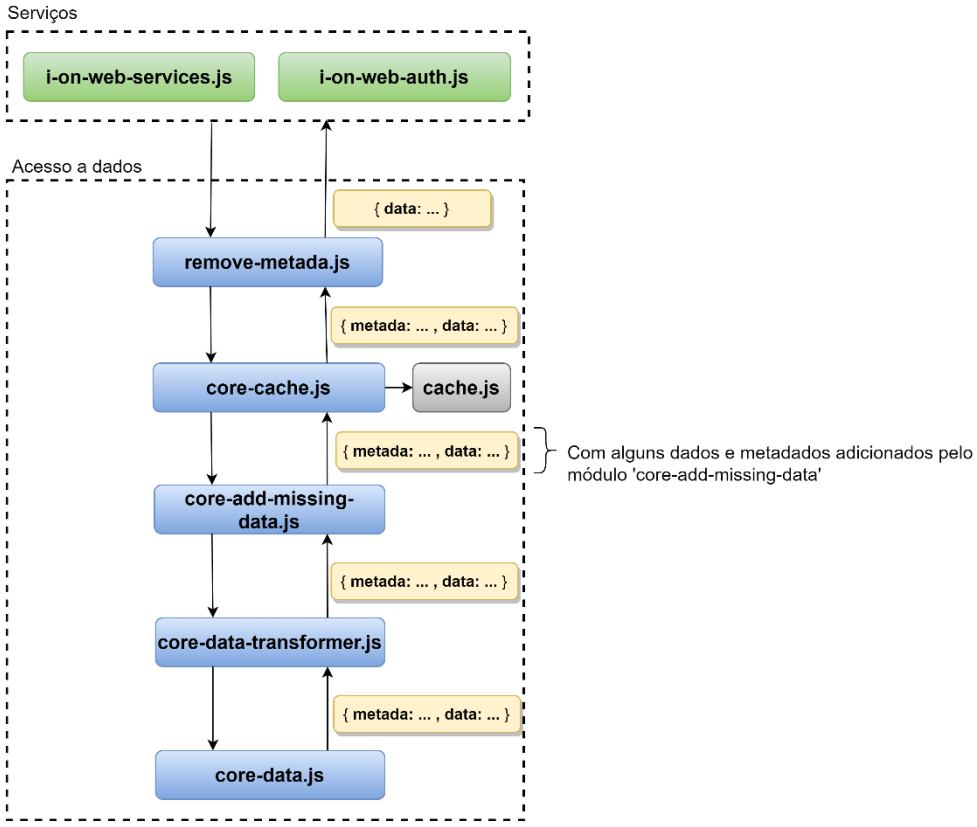

Figura 10 - Ilustração da passagem de dados e metadados no acesso ao i-on Core.

É de salientar que, como já referido, pretendeu-se uma implementação simples. Deste modo, os metadados (neste caso informação extraída dos *headers*) não chegam aos módulos de serviços. Idealmente, os metadados poderiam fluir até ao cliente de modo que também este pudesse realizar cache das respostas da aplicação web, no entanto, dada a escassez de tempo e a complexidade que tal traria optou-se por “parar” o fluxo de metadados na chegada aos módulos de serviços. Para tal, foi adicionado um módulo ‘remove-metadata’ que, como já referido no capítulo 4, tem a função de retirar os metadados das respostas recebidas.

Módulo ‘core-cache’

A lógica de cache presente no módulo ‘core-cache’ pode ser descrita através do seguinte esquema:

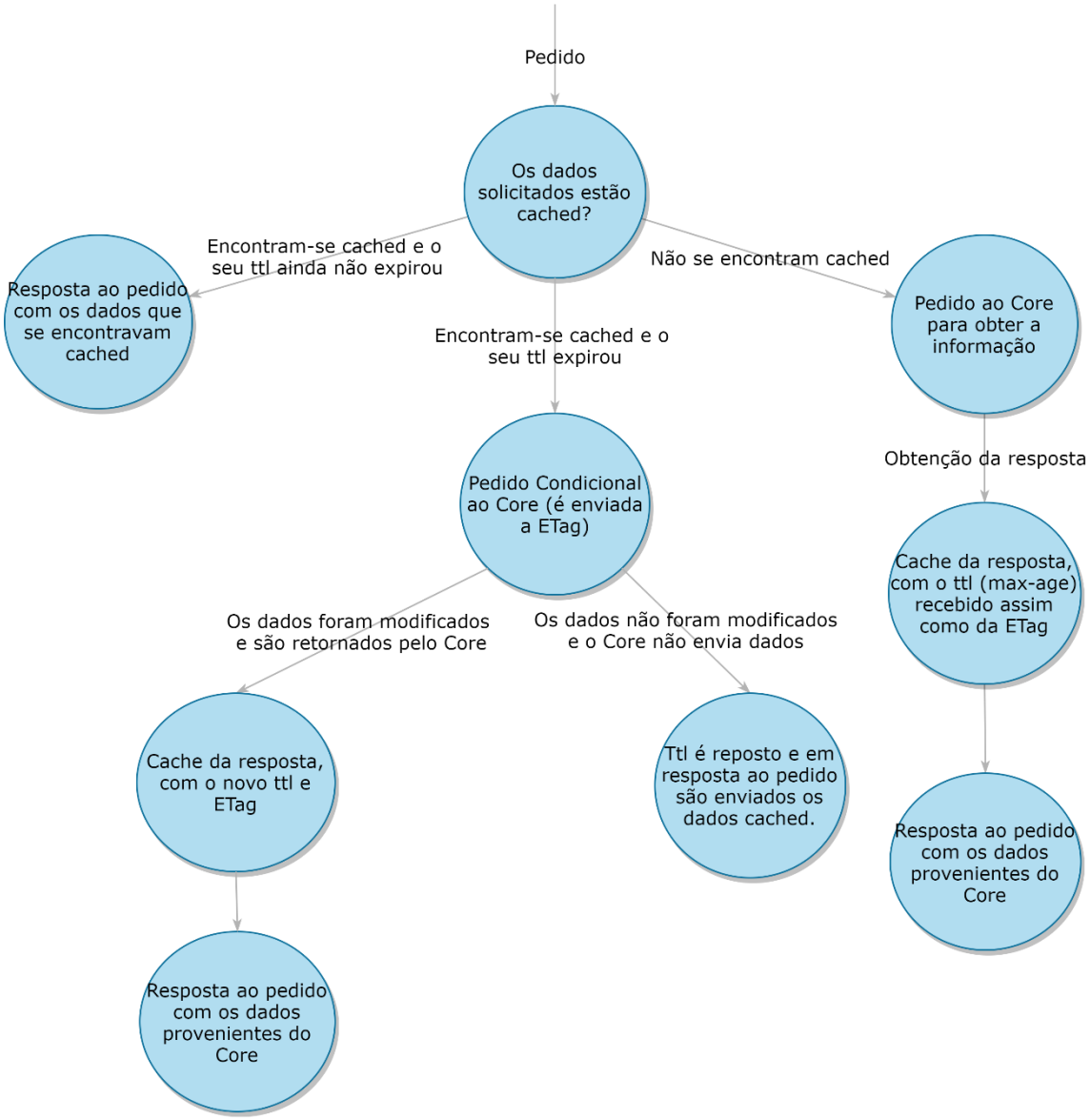

Figura 11 - Lógica de cache.

Pedidos Condicionais

No esquema anteriormente apresentado foi possível observar a lógica de cache, agora, irá se abordar o que são pedidos condicionais e como se processam.

Pedidos condicionais são pedidos cuja resposta depende de determinada condição, mais especificamente, do valor de determinados validadores como *ETag* e

lastModified, sendo que no caso apenas se utiliza a *ETag*. Faz-se uso desta para evitar transferências de recursos desnecessariamente de modo que apenas sejam transferidos recursos que de facto tenham mudado.

Nos pedidos GET é enviado o *header* '*If-None-Match*' (pedido condicional) com o valor da *ETag* que a aplicação possui referentemente aos dados em cache.

O objetivo seria o *origin-server* (neste caso o Core) saber produzir *ETags* colocando-as nas respostas e saber lidar com **pedidos condicionais**. Perante estes, caso o recurso não tivesse sido alterado, ou seja, se os validadores enviados coincidissem com os presentes no *origin-server*, então seria recebida uma resposta com *status code* 304 (*Not Modified*) sem *body*. Desta forma, iria-se reduzir as transferências de conteúdo, poupando, por exemplo, na largura de banda.

No caso de existir alguma mudança nos dados, então aí sim, deveria ser retornada uma resposta com *status code* 200 e com um *body* contendo a nova representação do recurso.

Módulo 'cache'

É ainda de referir, a existência de uma classe 'Cache' (módulo 'cache') que apresenta uma interface com os métodos que se considerou relevantes para a implementação da lógica da Cache, nomeadamente, para efetuar o cache de dados, para a obtenção de dados *cached*, obtenção do *ttl* (*time-to-live*) de determinados dados, entre outros. Na implementação destes são utilizados os métodos disponibilizados pelo node-cache.

Optou-se pela criação desta classe ao invés da utilização direta no node-cache de modo a tornar a utilização de cache no módulo 'core-cache' mais genérica (acoplamento fraco entre este e o node-cache), permitindo, com facilidade, a mudança do sistema de cache.

Valores atribuídos aos metadados

Como já referido, o Core não implementa qualquer tipo de cache, assim sendo, apesar de a implementação existente se encontrar pronta para a receção dos mesmos estes nunca são enviados. Deste modo, os metadados referidos e que são utilizados (*ETag* e *max-age*) são adicionados no módulo 'core-add-missing-data'. No caso da *ETag* é lhe atribuída uma *string* enquanto para o *max-age* são definidos os seguintes valores:

- 24 horas (em segundos) para todos os dados que sejam recursos académicos, uma vez que estes não sofrem mudanças frequentemente;
- 5 minutos (em segundos) para o perfil do utilizador, uma vez que informações de utilizador são bastante instáveis podendo ser constantemente alteradas.

Nota:

É de referir que no caso de informações do utilizador como é o caso com a obtenção do perfil do mesmo, de modo que o utilizador ao, por exemplo, editar o seu perfil (mudar o nome de utilizador) possa ver de imediato o seu novo *username* sem ter que aguardar 5 minutos, a cada pedido de edição de perfil, o perfil guardado em cache é

eliminado, de modo a que quando surgir o pedido para a obtenção do mesmo o utilizador possa obter o mais recente em vez do que se encontrava em cache.

Esta lógica só se aplica à nossa aplicação, caso o utilizador esteja em simultâneo na aplicação web e android e altere o nome na aplicação android só após 5 minutos é que verá o novo nome na aplicação web. Apesar desta 'desvantagem' em que a alteração poderá não ser vista de imediato, considerou-se que neste *tradeof* entre:

- A experiência de utilizador, ao não ver de imediato na aplicação i-on Web edições ao perfil feitas em outras aplicações cliente;
- O tempo de resposta da aplicação.

Considerou-se que o último seria mais vantajoso uma vez que, esta opção, diminuiria o tempo de resposta da aplicação o que se crê ser mais importante na experiência de um utilizador.

Capítulo 8

User Interface

O capítulo presente descreve aspetos da *user interface*, nomeadamente, como esta foi estruturada, como foi tirado partido da *framework* Bootstrap e dos *templates* Handlebars, assim como questões de implementação mais relevantes que lhe estejam associadas.

A interface é maioritariamente renderizada no lado do servidor (*server-side rendering*) e é realizada a partir de HTML para definição da estrutura e lógica de apresentação e CSS para a realização de uma interface mais agradável.

8.1 Fluxo de navegação

Ao longo do desenvolvimento do projeto teve-se como objetivo que a *user interface* fosse de fácil utilização, cómoda e que com poucas indirecções o utilizador conseguisse chegar ao pretendido, deste modo, desenhou-se um fluxo de navegação relativamente simples. Neste, um dos elementos que mais contribui para o fluxo de navegação é a navbar, uma vez que esta permite que o utilizador navegue de qualquer ponto da aplicação para uma outra página com somente um clique.

O fluxo de navegação pode ser representado pelo esquema abaixo.

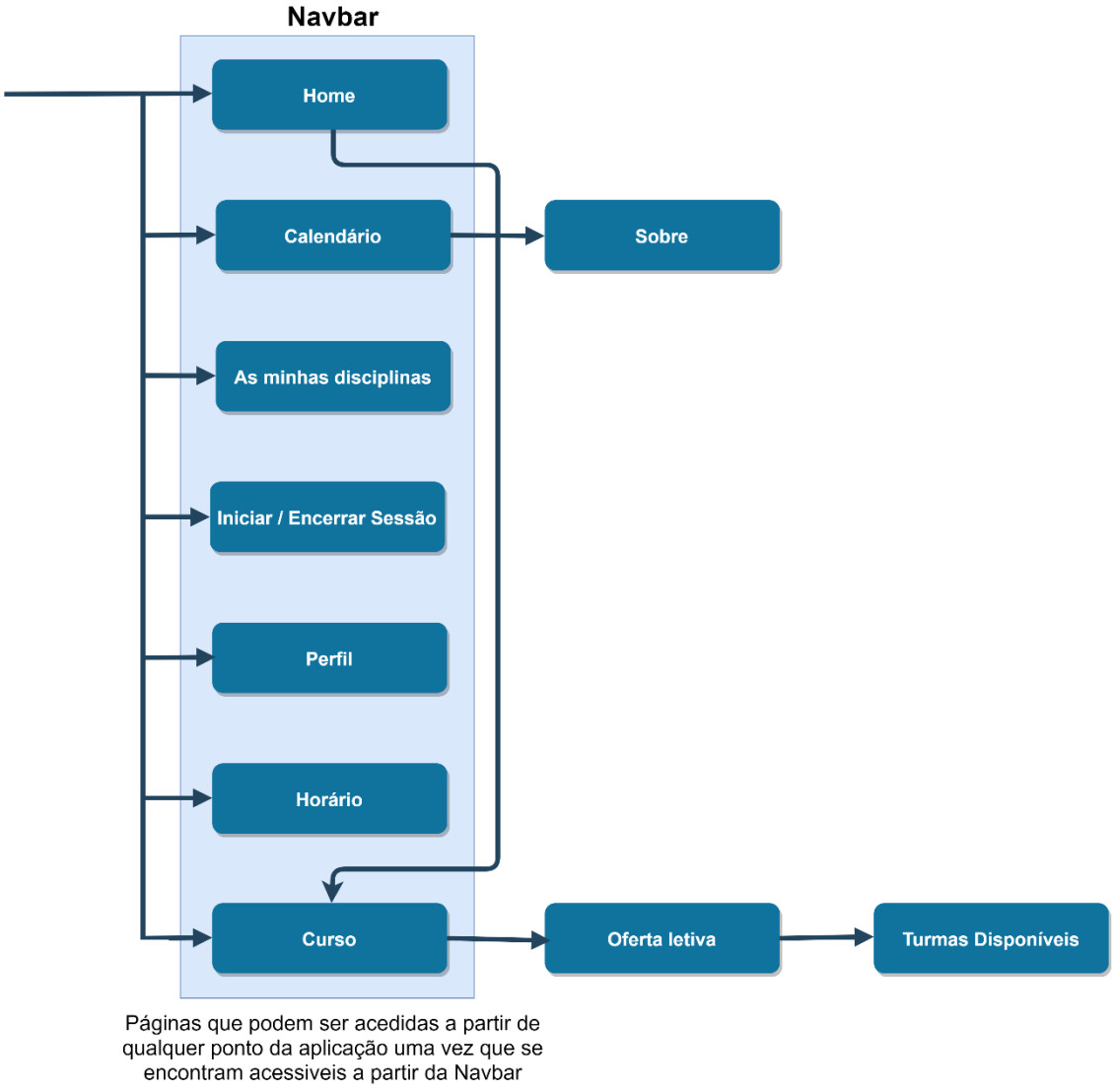

Figura 12 - Fluxo de navegação da aplicação.

As páginas da figura 12 apresentam a seguinte informação:

- **Home –** Página inicial da aplicação. Tem como maior objetivo apresentar a lista de cursos (neste momento apenas licenciaturas e mestrados) da instituição. Caso o utilizador se encontre autenticado também são expostos os próximos eventos (entrega de trabalhos e exames) associados às unidades curriculares que este subscreveu.

Adicionalmente, é apresentado um elemento *footer* com informações e links referentes à iniciativa e à instituição

- **About –** Menciona os projetos inerentes à iniciativa i-on assim como os alunos e docentes participantes;
- **Iniciar sessão –** Página que disponibiliza ao utilizador os métodos e formulário de autenticação;
- **Perfil –** Página de perfil do utilizador, permitindo a este alterar o seu *username* e apagar a sua conta;
- **Curso –** Apresenta informação geral sobre o curso, como por exemplo, coordenação, departamento, plano curricular, entre outros;
- **Oferta letiva –** Apresenta a oferta letiva de determinado curso no semestre corrente. Nesta página o utilizador pode selecionar as unidades curriculares pretendidas;
- **Turmas disponíveis –** Página resultante da seleção de unidades curriculares da página anterior. Neste, é possível o utilizador escolher as turmas às quais deseja subscrever;
- **As minhas disciplinas –** Apresenta as disciplinas e respetivas turmas a que o utilizador se encontra subscrito, permitindo ainda a anulação da subscrição;
- **Calendário –** Apresentação de um calendário que contém informação genérica do calendário escolar (data de início e fim das aulas, épocas de exame, etc.).

  Caso o utilizador esteja autenticado, em adição à informação genérica de calendário escolar, também lhe é apresentada informação dos eventos (entrega de trabalhos e exames) que este possui, à semelhança da página home.
- **Horário –** Apresenta ao utilizador um horário personalizado com informação das salas e horas das aulas associadas às unidades curriculares que se encontra subscrito. Adicionalmente, disponibiliza-se a transferência do horário do utilizador no formato pdf.

8.2 Dinâmica das páginas

Nesta secção será mencionada a forma como se tirou partido das tecnologias Bootstrap, Handlebars, JavaScript e jQuery de forma a desenhar as páginas de forma que se adaptem em conformidade com o tamanho e orientação do ecrã do utilizador.

Capacidade de adaptação a diferentes ecrãs (uso de Bootstrap e CSS)

Das várias formas de adaptação de componentes a diferentes ecrãs é de salientar, o uso o sistema de *grid*, a *framework* Bootstrap.

Mais especificamente, o *bootstrap* utiliza '*breakpoints*' (baseados na largura do dispositivo) de modo a alterar o número de colunas que determinado elemento irá ocupar na página (dividida em 12 colunas).

ara fazer um uso efetivo deste sistema, utilizou-se, por exemplo, classes que especificam o número de colunas que se pretende que determinado elemento ocupe consoante o tamanho do ecrã. Em seguida, são apresentados alguns *screenshots* de páginas da aplicação visualizadas a partir de diferentes ecrãs.

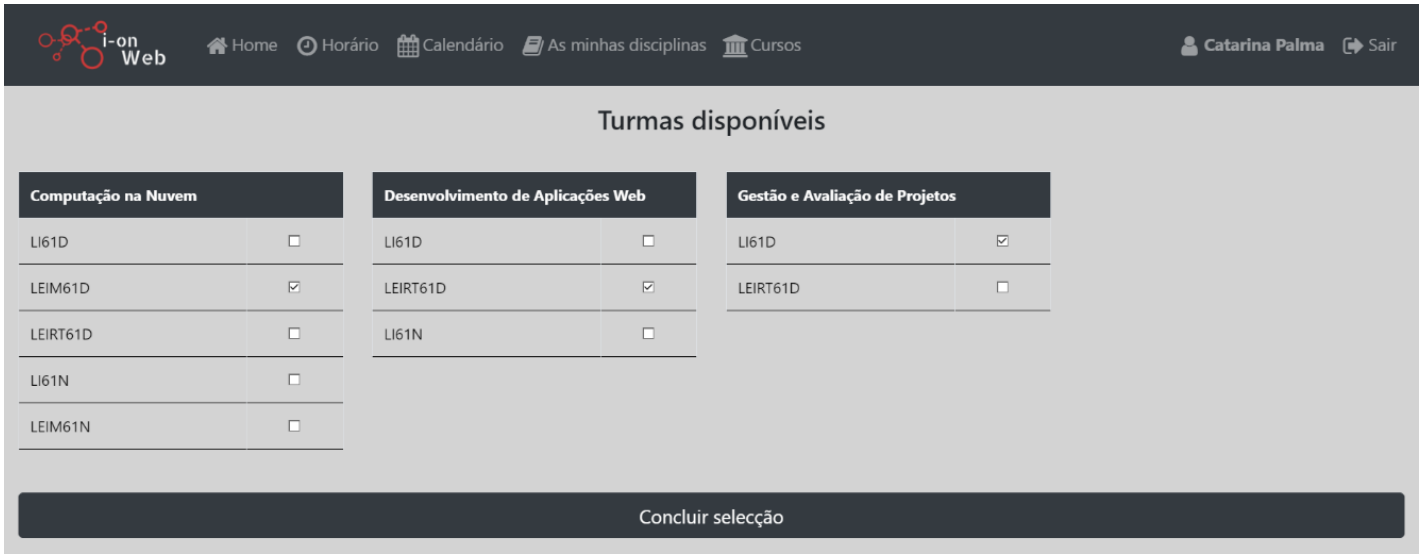

Figura 13 - Página da aplicação web referente às turmas disponíveis vista a partir de um portátil.

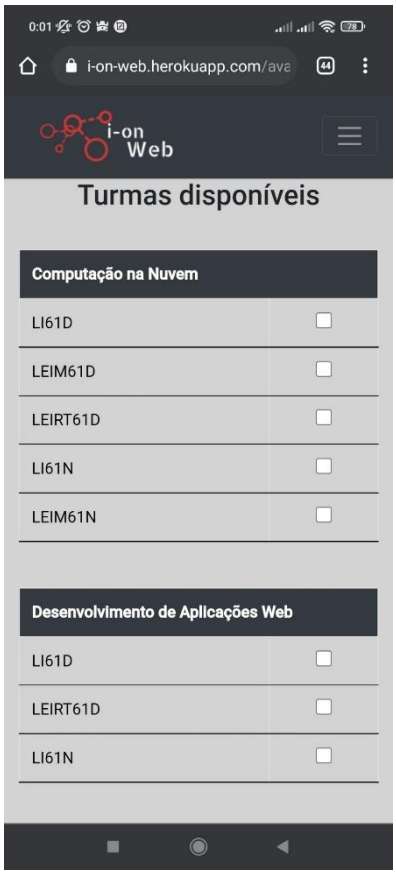

Figura 14 - Página da aplicação web referente às turmas disponíveis vista a partir de um telemóvel.

Atualmente, existem elementos na aplicação cuja capacidade de adaptação poderia ser mais trabalhada, no entanto, teve-se em atenção elementos que se considerou de elevada relevância, nomeadamente, a *navbar* que ao permitir o utilizador aceder a múltiplas páginas, considerou-se fundamental que este elemento 'respondesse' bem a ecrãs de diferentes dimensões, deste modo, utilizou-se classes construídas pelo *bootstrap* para que em ecrãs de menor dimensão, a *navbar* tivesse um efeito de 'colapso'. Tal efeito é visível nas figuras seguintes:

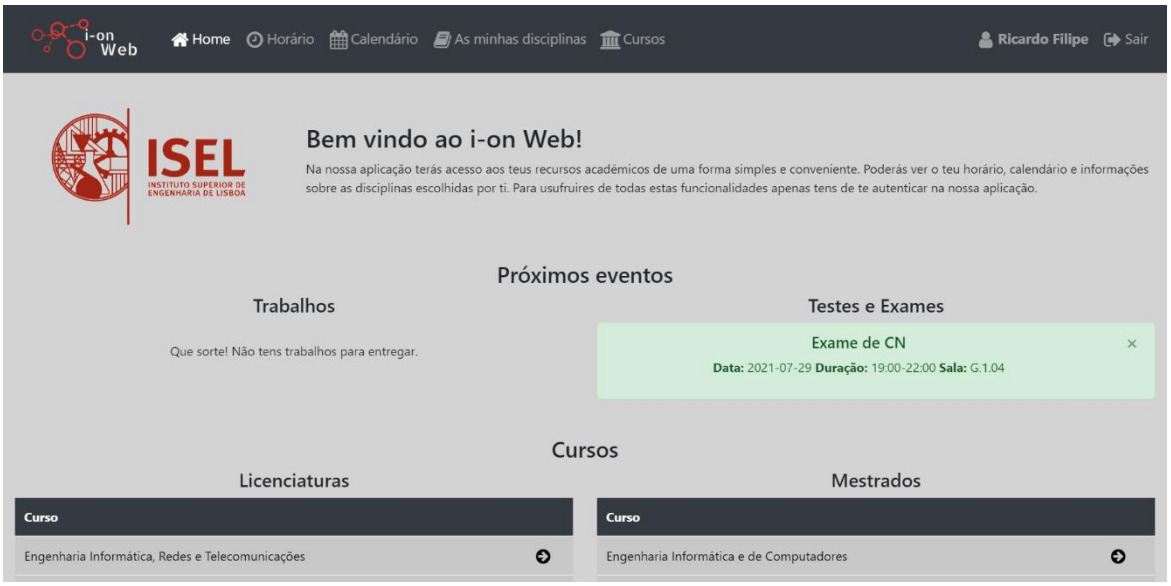

Figura 15 - Página home da aplicação web vista a partir de um portátil.

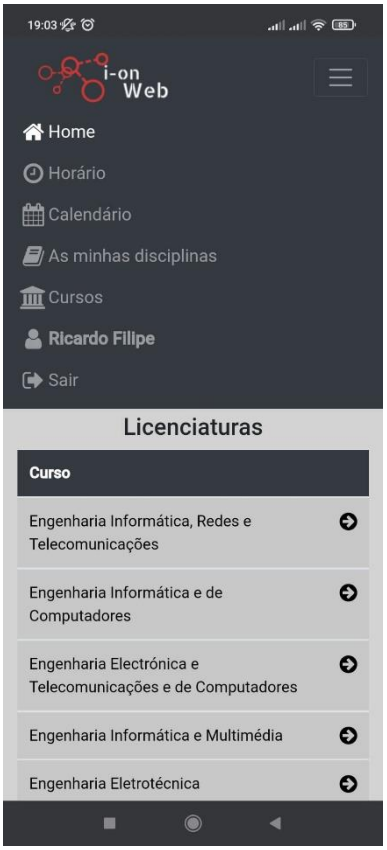

Figura 16 - Página home da aplicação web vista a partir de um telemóvel.

Em algumas circunstâncias, em complemento ao sistema grid, foi utilizada a propriedade **@media**, da tecnologia CSS, de modo a alterar o tamanho e conteúdo de alguns elementos de acordo, por exemplo, com a largura do ecrã. Um exemplo da sua utilização é visível nas figuras 17 e 18 em que, o nome dos cabeçalhos da tabela e o nome das unidades curriculares, é alterado caso se esteja perante um ecrã de menor dimensão (fig. 18).

Figura 17 - Página da aplicação web referente à oferta letiva vista a partir de um portátil.

Figura 18 -Página da aplicação web referente à oferta letiva vista a partir de um telemóvel.

Templates Handlebars

Em complemento à *framework bootstrap* é de referir a utilização de alguns elementos interessantes do Handlebars.

De modo que, consoante a informação passada ao *template*, seja renderizado um bloco HTML ao invés de outro, tirou-se partido do já implementado *helper* '**#if**' presente nos *templates* Handlebars. Um exemplo da sua utilização, é a existência ou não de eventos de um determinado utilizador, isto é, no *template* da página home utilizou-se o *helper #if* de forma a verificar a existência de eventos, apresentando os mesmos ao utilizador ou apenas um parágrafo caso não os haja.

É ainda de referir a utilização de o *helper* '**#each**' que permite iterar sobre uma lista assim como referenciar o elemento que está a ser iterado (*helper* utilizado, por exemplo, para iterar e apresentar a lista de cursos na página home).

Uso de Javascript e jQuery

Em alguns casos, foram utilizados *scripts* do lado do cliente de modo a tornar a interface mais dinâmica e evitar alguns pedidos considerados desnecessários à aplicação web, nomeadamente, na página de perfil em que para ser demonstrado o formulário para edição de perfil considerou-se inconveniente a realização de mais um pedido à aplicação. Assim sendo, através da biblioteca jQuery foi possível a interação entre os scripts e a página HTML, apresentando e omitindo elementos em função das ações do utilizador.

8.3 Páginas de implementação complexa

Por último, é ainda de mencionar algumas das páginas cuja realização consumiu mais tempo, nomeadamente, as páginas de calendário e horário. Estas, apesar de inspiradas em implementações já existentes online, exigiram compreensão e alterações para adaptação ao contexto do i-on Web.

Horário

A grelha horária é constituída de vários elementos 'div' cuja disposição e tamanho é especificada em pixéis. Para cada um destes são especificadas classes CSS indicando a que hora/dia a que este se encontra associado. Deste modo, a colocação de eventos no horário encontra-se então dependente da informação do dia da semana e da hora de início e fim do evento (aula) para a sua devida colocação no horário.

Na divisão do horário em blocos de tempo, optou-se por intervalos de 30 minutos, com a finalidade de englobar a diversidade de horários dada a variedade de cursos, unidades curriculares e turmas, mas simultaneamente, não se tornar um horário 'exaustivo' ou até confuso visualmente no que toca à divisão das horas.

Relativamente à forma como são expostas as aulas, optou-se por atribuir uma cor aleatória a cada unidade curricular de modo a melhor distinguir visualmente, as diferentes unidades curriculares. Outro aspeto relevante na exposição do horário é a sobreposição de aulas, para tal, as cores atribuídas apresentam alguma "transparência" de modo que, mesmo dada a existência de sobreposição, por exemplo, de duas aulas sobrepostas, a visualização destas continua a ser percetível.

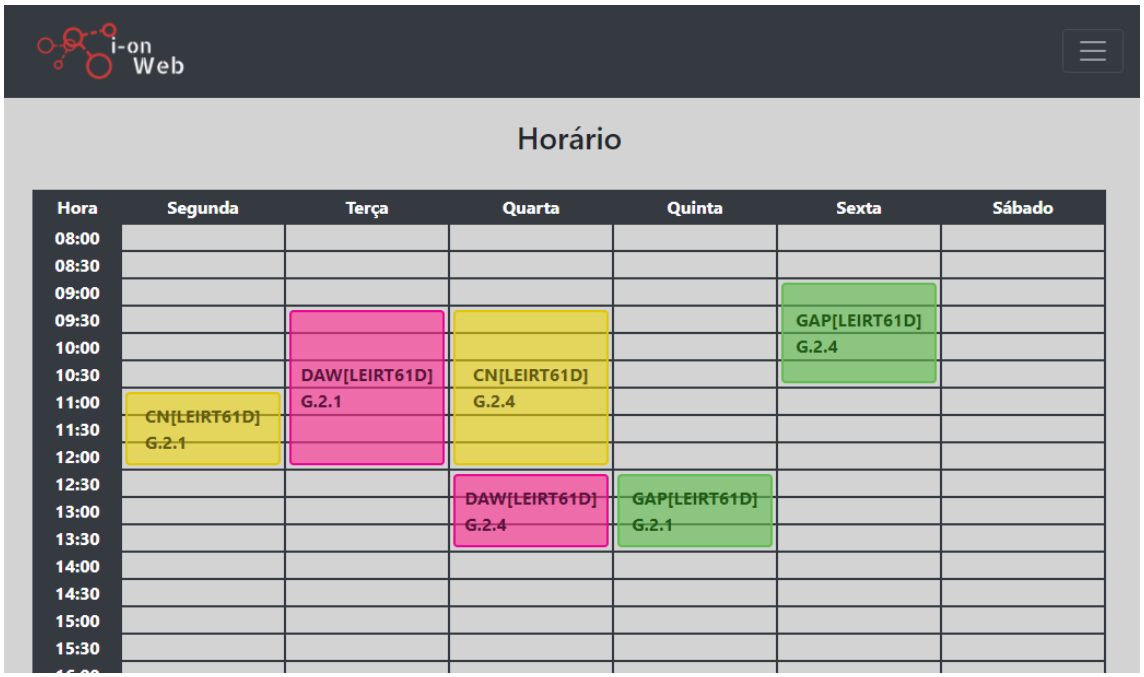

Figura 19 - Página da aplicação referente ao horário.

Calendário

O calendário, à semelhança do horário, também é construído tendo por base vários elementos 'div', nestes em conformidade com a data atual e eventos existentes são adicionados outros elementos (através de um *script*) de forma a apresentar o calendário de acordo com os mesmos.

Optou-se por mostrar as descrições dos eventos numa barra lateral, existindo apenas uma pequena indicação no calendário da existência de eventos em determinada data, cabendo ao utilizador, caso pretenda visualizar quantos e quais, selecionar o dia em questão para a visualização dos mesmos. Por omissão, ou seja, no estado inicial da página o dia que se encontra selecionado e por consequência os eventos que são expostos são os referentes à data atual.

A colocação de eventos é realizada com base na data e título do mesmo podendo opcionalmente ser atribuída uma descrição. Os eventos expostos são obtidos com base na data selecionada pelo utilizador (como já referido) sendo esta utilizada para a procura de eventos associados à mesma, obtendo o título e descrição (caso exista) dos mesmos e apresentando-os. Existe ainda a possibilidade de ver meses anteriores e posteriores, permitindo assim a visualização de eventos que já ocorreram ou que irão ocorrer em meses futuros.

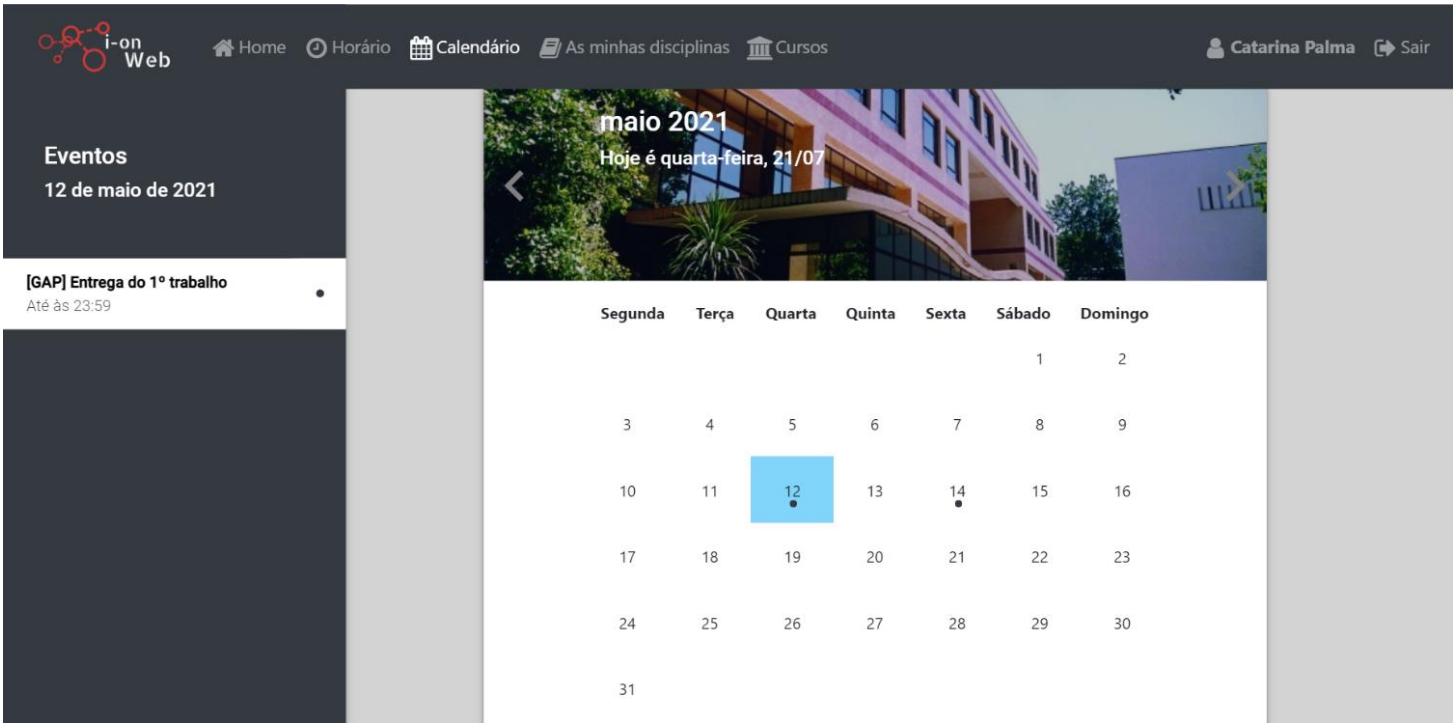

Figura 20 - Página da aplicação referente ao calendário.

Por último, é de mencionar que algumas páginas (nomeadamente, as de horário e calendário) não estão ainda totalmente preparadas para se adaptarem a ecrãs de diferentes dimensões. Tal deve-se à complexidade da sua construção para ecrãs maiores, de forma que, não se conseguiu da melhor forma que estas se adaptassem a ecrãs de *smartphones*, abrindo margem a melhorias. O horário também poderia ser melhorado com a implementação de um mecanismo que detetasse unidades curriculares sobrepostas e reagisse em conformidade, por exemplo, apresentando ao utilizador um aviso.

Capítulo 9

Conclusões e Trabalho Futuro

No decorrer deste último capítulo será feito um sumário de tudo o que foi e não foi alcançado dado os objetivos iniciais, assim como, possível trabalho futuro para melhorar o projeto caso houvesse mais tempo para desenvolver o mesmo. Adicionalmente, irá indicar-se algumas das maiores dificuldades sentidas ao longo do desenvolvimento do projeto.

Objetivos alcançados

Em termos de objetivos alcançados é possível concluir que, com base no planeamento inicial todos os objetivos classificados como fundamentais foram cumpridos, assim como a restante maioria de objetivos delineados, como por exemplo:

- Autenticação dos utilizadores através do email institucional;
- Colocar a aplicação disponível na plataforma Heroku.
- Possibilidade de o utilizador obter a oferta letiva de determinado curso num dado semestre, e por consequência a possibilidade de subscrição às turmas disponíveis numa dada unidade curricular;
- Apresentar ao utilizador a lista de turmas às quais se encontra subscrito e, logicamente, por consequência, a opção de editar a mesma (subscrevendo ou anulando a subscrição a determinada turma);
- Exposição do horário do utilizador em conformidade com as turmas a que este se encontra subscrito assim como a opção de transferir o mesmo (em formato pdf);
- Apresentar na página inicial lembretes relativos aos próximos eventos (exames, trabalhos, entre outros) das turmas a que este se encontra subscrito;
- Exposição de um calendário com os eventos das disciplinas a que o utilizador se encontra subscrito assim como informação relativa ao calendário escolar;

Um dos objetivos definidos inicialmente e que não foi concretizado foi a possibilidade de alternar o idioma (entre português e inglês) da interface web, no entanto, é de frisar que já no planeamento inicial esta funcionalidade era de baixa prioridade, sendo considerada um aspeto interessante, mas não uma funcionalidade essencial. Como no decorrer do desenvolvimento surgiu a questão da possibilidade da implementação de

cache dos dados do Core, optou-se por dar-se mais importância a este uma fez que está diretamente relacionado com a performance da aplicação.

Adversidades

Uma das maiores dificuldades sentida ao longo do desenvolvimento projeto foi a constante necessidade de adaptação da aplicação à medida que o projeto i-on Core também era desenvolvido. Tal exigiu uma constante preocupação e atenção em estar a par das atualizações que iam acontecendo neste. Além disto, e como referido ao longo do relatório, os outros projetos encontravam-se também em desenvolvimento o que resultou na falta ou inconsistência de recursos, algo que, poderá ter causado alguns entraves no desenvolvimento da aplicação e muitas vezes levado a encontrar soluções alternativas.

Trabalho futuro

Apesar da aplicação ter uma implementação e conjunto de funcionalidades que se considera sólida ainda poderia usufruir de melhorias, nomeadamente:

- Permitir alternar o idioma (entre português e inglês) da interface web;
- Expandir a lógica da cache para o lado do *browser* para que também este possa realizar cache dos dados (o que envolveria enviar informação de caching para o *browser* assim como saber interpretar pedidos condicionais);
- Além de pedidos condicionais feitos ao Core seria interessante realizar atualizações condicionais o que permitia resolver possíveis problemas de concorrência quando várias aplicações cliente tentassem atualizar o mesmo recurso;
- Os testes de integração encontram-se numa versão muito básica não realizando validações às componentes que seriam de facto interessantes, assim sendo, poder-se-ia melhorar e aumentar as asserções realizadas nestes;
- Alguns detalhes/elementos na *user interface* não são tão dinâmicos como o pretendido, deste modo, poder-se-ia melhorar a *user interface* neste aspeto;
- Disponibilizar uma página com informação referente a determinada unidade curricular, apresentando informação mais detalhadas sobre a mesma.

Bibliografia

[1] i-on, “Dossier de Projeto (Repositório GitHub), i-on Web,” 2021. [Online]. Available: https://github.com/i-on-project/web. [Acedido em 22 Julho 2021].

[2] Elasticsearch, “Bonsai Elasticsearch,” 3 Junho 2021. [Online]. Available: https://elements.heroku.com/addons/bonsai. [Acedido em 14 Junho 2021].

[3] i-on, “Core, the i-on System's repository of academic information.,” 2021. [Online]. Available: https://github.com/i-on-project/core. [Acedido em 17 Julho 2021].

[4] G. Fernandez, “OpenID Connect Client-Initiated Backchannel Authentication Flow,” OpenID Connect, 4 Junho 2021. [Online]. Available: https://openid.net/specs/openid-client-initiated-backchannel-authentication-core-1_0.html. [Acedido em 10 Junho 2021].

[5] NodeJS, “Dockerizing a Node.js web app,” OpenJS Foundation, 31 Março 2021. [Online]. Available: https://nodejs.org/en/docs/guides/nodejs-docker-webapp/. [Acedido em 12 Abril 2021].

[6] Cheerio Collaborators, “Cheerio,” 1 Junho 2021. [Online]. Available: https://www.npmjs.com/package/cheerio. [Acedido em 18 Junho 2021].

[7] J. Kresner, “ExpressJS and PassportJS Sessions Deep Dive,” 1 Janeiro 2014. [Online]. Available: https://www.airpair.com/express/posts/expressjs-and-passportjs-sessions-deep-dive. [Acedido em 29 Maio 2021].

[8] M. McCarty, “Securing Node.js: Managing Sessions in Express.js,” 6 Janeiro 2017. [Online]. Available: https://lockmedown.com/securing-node-js-managing-sessions-express-js/. [Acedido em 29 Maio 2021].

[9] MDN contributors, “JavaScript reference,” 2021. [Online]. Available: https://devdocs.io/javascript/. [Acedido em 12 Julho 2021].

[10] P. Félix, “A docker course,” 28 Março 2021. [Online]. Available: https://github.com/pmhsfelix/course-docker. [Acedido em 15 Julho 2021].

[11] C. Arsenault, “HTTP Cache Headers - A Complete Guide,” 24 Maio 2018. [Online]. Available: https://www.keycdn.com/blog/http-cache-headers. [Acedido em 5 Junho 2021].

[12] i-on, “i-on Project Vocabulary,” 11 Março 2020. [Online]. Available: https://github.com/i-on-project/meta/blob/master/arch/vocab.md. [Acedido em 15

Março 2021].

[13] Github, “About pull requests,” [Online]. Available: https://docs.github.com/en/github/collaborating-with-pull-requests/proposing-changes-to-your-work-with-pull-requests/about-pull-requests. [Acedido em 7 Abril 2021].

[14] Github, “Mastering Issues,” 24 Julho 2020. [Online]. Available: https://guides.github.com/features/issues/. [Acedido em 7 Abril 2021].

[15] Collaborators, “CodePen,” 2021. [Online]. Available: https://codepen.io. [Acedido em 18 Junho 2021].

[16] Github, “Deploy Multiple Docker Images to Heroku Apps,” 2021. [Online]. Available: https://github.com/marketplace/actions/deploy-multiple-docker-images-to-heroku-apps. [Acedido em 15 Abril 2021].

[17] Github, “Heroku Deploy,” 2021. [Online]. Available: https://github.com/marketplace/actions/deploy-to-heroku. [Acedido em 15 Abril 2021].

[18] Heroku, “Heroku Dev Center - Documentation,” 2021. [Online]. Available: https://devcenter.heroku.com/categories/reference. [Acedido em 25 Junho 2021].

[19] Elasticsearch, “Elasticsearch Guide,” 2021. [Online]. Available: https://www.elastic.co/guide/en/elasticsearch/reference/current/search-search.html. [Acedido em 8 Julho 2021].

[20] Mozilla, “Resources for developers,” 2021. [Online]. Available: https://developer.mozilla.org/. [Acedido em 16 Julho 2021].

[21] Docker, “Documentation,” 2021. [Online]. Available: https://docs.docker.com. [Acedido em 30 Abril 2021].

[22] “Node-cache,” 2021. [Online]. Available: https://www.npmjs.com/package/node-cache. [Acedido em 10 Julho 2021].

[23] Bootstrap, “Get started with Bootstrap,” 2021. [Online]. Available: https://getbootstrap.com/docs/4.6/getting-started/introduction/. [Acedido em 17 Julho 2021].

[24] wjdgus0117@hanmail.net, “Responsive Calendar,” [Online]. Available: https://codepen.io/wjdgus0117/pen/KKWQjoR. [Acedido em 19 Junho 2021].

[25] Kogisune, “Schedule Grid,” [Online]. Available: https://codepen.io/kogisune/details/yLMKvPa. [Acedido em 10 Junho 2021].