Instituto Superior de Engenharia de Lisboa

Área Departamental de Engenharia de Eletrónica e Telecomunicações e de Computadores

Projeto i-on Web

Relatório Intercalar da Unidade Curricular de Projeto

Licenciatura de Engenharia Informática, Redes e Telecomunicações

Projeto 21

Alunos

Ricardo Severino nº 45245

Catarina Palma nº 45241

Orientadores

Professor João Trindade

Professor Luís Falcão

2º Semestre do ano letivo 2020/2021

03 de maio de 2021

Índice

Índice de Ilustrações

Introdução

Ao longo do relatório de progresso tem-se como objetivo primordial descrever sucintamente o desenvolvimento do projeto i-on Web até à data. Entre outros aspetos, serão referidas as decisões tomadas e as tecnologias utilizadas, assim como adversidades encontradas e quais as soluções tomadas para as contornar. Assim sendo, tendo em conta o trabalho desenvolvido nas últimas três quinzenas, é possível dividir o mesmo em três fases:

Fase 1 – Ambientação das tecnologias a utilizar **–** Serão expostas quais as tecnologias usadas, algumas das suas características assim como a razão pela qual foram selecionadas.

Fase 2 – Desenvolvimento do projeto **–** Será apresentada a estrutura do projeto, o trabalho realizado, dificuldades e decisões envolvidas até ao momento.

Fase 3 – i-on Web online e próxima fase **–** Será abordado o processo de *deploy* da aplicação, assim como será referido o trabalho em progresso e alguns dos objetivos pretendidos numa próxima fase.

Fase 1 – Ambientação das tecnologias a utilizar

É frequente, ao longo do percurso académico, desenvolver aplicações que são apenas executadas e exploradas localmente sem realmente existir a experiência de concretizar o *deploy* da aplicação, disponibilizando a mesma online.

O i-on Web, em adição ao que é habitualmente realizado na grande maioria das unidades curriculares, apresenta um cariz mais profissional trazendo consigo a necessidade de ter toda uma infraestrutura bem pensada e estruturada para que seja possível executar a aplicação não só localmente, mas também num servidor que a exponha ao público.

Para atingir tais objetivos, começou-se por investigar ferramentas amplamente utilizadas no mundo profissional e que teriam as características necessárias para cumprir o papel a desempenhar no projeto. Após exploração e seleção das tecnologias a utilizar, estudou-se cada uma delas mais aprofundadamente, dando sempre especial destaque à documentação oficial, assim como ao *feedback* fornecido pelos diversos utilizadores que já experienciaram a ferramenta, o que permite, de antemão, prever possíveis vantagens e contratempos que cada uma das tecnologias nos pode oferecer.

GitHub

Assim como todos os projetos que a iniciativa i-on integra, o projeto i-on Web irá utilizar um dos sistemas de controlo de versões mais famosos no mundo **– repositórios Git –** 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, pretende-se tirar proveito das seguintes *features* presentes no GitHub.

- ***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 será 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. Pretende-se utilizar esta ferramenta para a execução de testes unitários para que o código realizado seja validado, mantendo a integridade do projeto intacta. Virá ainda a ser 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 ficará 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*.

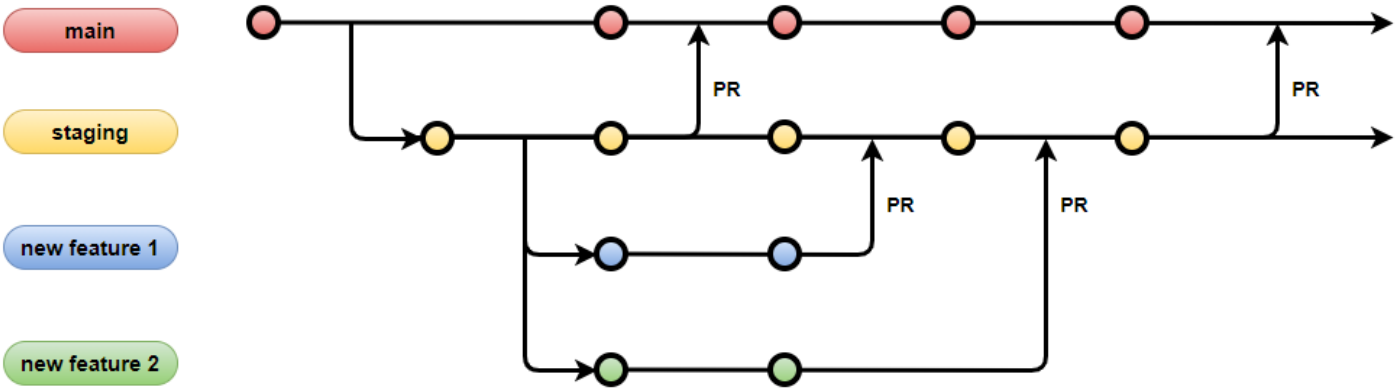

Figura 1 - Esquema referente à lógica de branches e pull requests.

Docker

A partir desta ferramenta pretende-se contornar dificuldades que se tem num modelo típico de desenvolvimento de *software*, nomeadamente, a instalação numa máquina de todas as aplicações e dependências necessárias para executar o código fonte. Este modelo revela diversas dificuldades.

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

O Docker por sua vez, usa o *kernel* do Linux para criar um nível de encapsulamento forte permitindo a segregação de processos, podendo estes serem executados de forma independente. O Docker oferece então a capacidade de empacotar e executar uma aplicação num ambiente isolado denominado ***container*** (contentor).

Associado a cada contentor, o Docker cria um espaço de trabalho dentro do sistema operativo que aparenta ser um sistema completamento isolado (cada contentor apresenta exclusivamente os seus processos) apesar de usar o mesmo *kernel* de sistema operativo e os mesmos dispositivos de *hardware*.

Em síntese, tem-se um sistema a gerir todos os recursos, estes estão fortemente segregados para cada um dos contentores, permitindo a cada um, criar um ambiente de execução pretendido. Este pode ser definido através de uma linguagem descritiva presente no ficheiro **Dockerfile**.

Qualquer sistema que suporte Docker pode então construir e executar a aplicação, adquirindo-se as seguintes vantagens:

- ➔ A versão do *software* utilizado está sob controlo do desenvolvedor;
- ➔ Não interferência entre processos (exceto a partilha de CPU);

Para que as imagens não ocupem muito armazenamento, o Docker usa a noção de *layers*, ou seja, cada imagem apenas contêm a diferença para a imagem anterior.

Como já referido, utilizou-se um Dockerfile (ficheiro de texto simples) contendo os comandos para construir a imagem desejada. Em complemento, utilizou-se a ferramenta **Docker Compose** que permite definir e executar múltiplos *containers*. A definição dos serviços que compõem a aplicação do Docker Compose está presente no ficheiro docker-compose.yml para que todos os *containers* possam ser executados juntos num ambiente isolado.

Em ambos os ficheiros (docker-compose.yml e Dockerfile) é possível definir todas as dependências da aplicação, variáveis de ambiente, portos a utilizar, entre outros.

Tendo em conta as características do Docker descritas, é percetível a facilidade fornecida pelo mesmo para a *deploy* da aplicação na *cloud*.

WSL2 (Windows Subsystem for Linux 2)

De notar que foi possível utilizar o Docker em máquinas Windows uma vez que estes 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.

Heroku

Sendo o objetivo primário ter o sistema em produção numa máquina disponibilizada pelo ISEL (situação ainda em desenvolvimento), 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 contribui 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 nosso caso particular.

O Heroku disponibiliza ainda *add-ons*, ou seja, ferramentas e serviços de desenvolvimento. Destes, já se explorou o Bonsai ElasticSearch e pretende-se utilizar o mesmo para servir como base de dados da aplicação para dados como os de sessão, por exemplo. Também no Bonsai ElasticSearch se tira partido de uma versão grátis (*sandbox*) apresentando também esta algumas limitações. No entanto, tendo em conta que o i-on Core irá conter a maioria da informação, crê-se que a versão gratuita será suficiente.

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.

Em síntese, esta primeira fase correspondeu ao estudo e investigação das tecnologias expostas neste capítulo. De realçar que de modo a ganhar um à-vontade com cada uma destas ferramentas, à medida que se ia estudando e aprendendo, utilizou-se aplicações básicas e posteriormente aplicações realizadas em outras unidades curriculares. Deste modo, foi possível aplicar os conhecimentos adquiridos ajudando a consolidar os mesmos, culminando numa melhor preparação para o seu uso no contexto do projeto i-on Web.

Fase 2 – Desenvolvimento do projeto

O desenvolvimento do projeto será suportado em Node.js, podendo eventualmente utilizar ElasticSearch para armazenar dados de sessão e outras preferências de utilizador. É ainda de referir que a interface será maioritariamente produzida em *server-side*, podendo vir a existir elementos *client-side*.

Camadas arquiteturais

De modo a organizar o projeto e a distinguir tarefas, o projeto pode ser subdividido nas seguintes camadas arquiteturais:

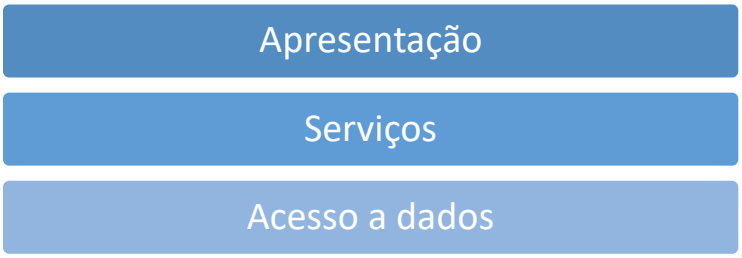

Figura 2 – Camadas arquiteturais.

Apresentação
Responsável pela interface de utilização e por lidar com o protocolo de comunicação, isolando assim o resto da aplicação do mesmo. Recebe os vários pedidos, extraindo a informação e argumentos necessários dos mesmos e consoante os *endpoints* dos pedidos, encaminha os mesmos para os respetivos serviços.

Serviços
Contém 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
Contém o acesso aos dados podendo estes estar numa API, base de dados ou até mesmo em ficheiros.

Seguindo esta arquitetura, a aplicação encontra-se então subdividida nos seguintes módulos, tendo sido atribuído a cada um destes a sua respetiva função.

i-on-web-server.js – Ficheiro que constitui o ponto de entrada na aplicação servidora, contendo as configurações necessárias ao funcionamento da aplicação.

i-on-web-ui.js – Responsável por receber os vários pedidos HTTP. Também utiliza *templates* Handlebars para gerar HTML para a apresentação da interface do utilizador.

i-on-web-services.js – Implementação da lógica de cada uma das funcionalidades da aplicação, nomeadamente, no que diz respeito à validação de erros e tratamento da informação recebida e enviada para o utilizador.

mock-data.js – Acesso aos ficheiros .json com dados fictícios (*mock*).

core-data.js – Acesso ao i-on Core.

i-on-web-errors.js – Ficheiro que contém a listagem dos possíveis erros. Estes são erros próprios da aplicação e são estes que circulam na mesma para que esta não esteja dependente do protocolo de comunicação, ou seja, no acesso a dados existe uma tradução dos erros para os erros da aplicação que por sua vez serão novamente traduzidos no módulo i-on-web-ui para os erros do protocolo de comunicação (no caso HTTP).

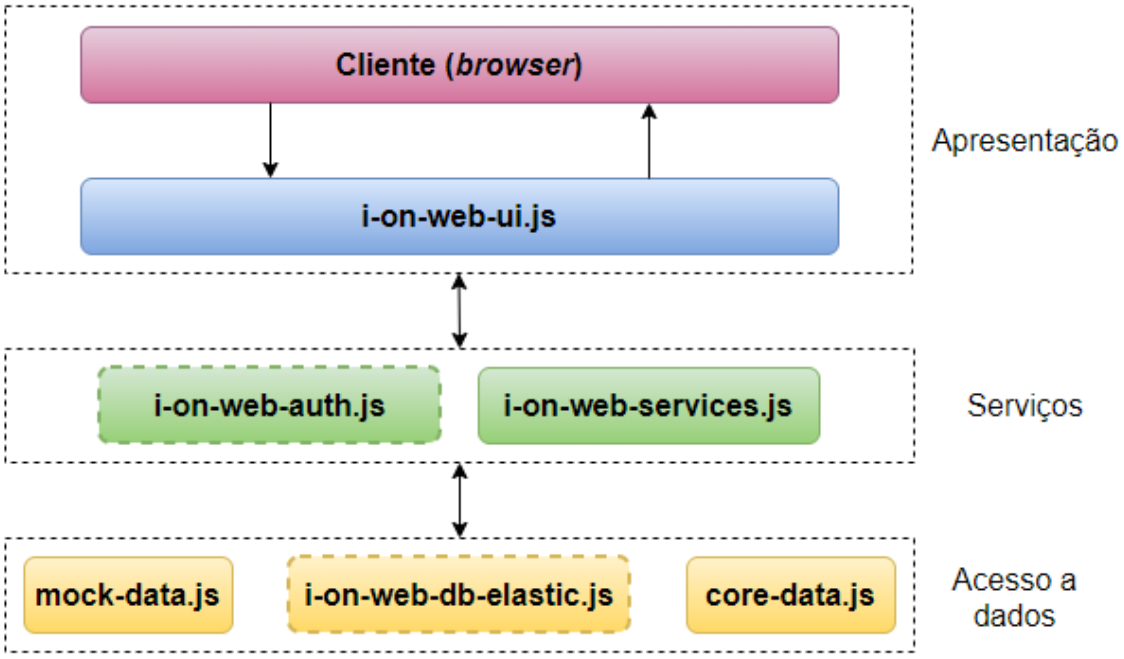

Figura 3 - Enquadramento dos módulos da aplicação na arquitetura de 3 camadas.

É de notar que os seguintes módulos virão ainda ser adicionados:

i-on-web-auth.js – Responsável pelas operações de autenticação;

i-on-web-db-elastic.js – Responsável pelo armazenamento dos dados na base de dados ElasticSearch.

Este tipo de arquitetura permite uma maior flexibilidade para divisão de tarefas permitindo aos desenvolvedores, no futuro, adicionar e/ou alterar pontos/módulos de entrada da aplicação (até com diferentes protocolos de comunicação) e/ou outros módulos de acesso a base de dados mesmo que sigam uma lógica diferente das restantes, sem qualquer afetação nos serviços que a suportam ou no acesso a dados realizado e vice-versa.

Ligação ao i-on Core

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 é 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, foi possível colocar a aplicação *multi-container* do Core em execução.

De seguida, utilizou-se um ficheiro Docker Compose para criar uma aplicação *multi-container* e definir:

- Os valores por omissão das variáveis de ambiente utilizadas;
- Os portos a utilizar;
- A localização do Dockerfile (para criação da imagem);
- Os nomes de serviço, contentor e imagem.

No Dockerfile são definidas as dependências e são fornecidos valores às variáveis de ambiente.

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.

Nota: A aplicação tira partido de 3 variáveis de ambiente:

- **OPERATION_MODE –** Indica o modo de operação;
- **CORE_READ_TOKEN –** *Token* necessário para a obtenção de informação por parte do Core;
- **CORE_URL –** A localização do Core.

Por omissão, os valores que estas possuem são os corretos para a conexão ao Core, no entanto, caso o utilizador o pretenda, no momento da construção da imagem poderá inserir outros valores.

Modos de Operação em execução Local

A obtenção de informação por parte do projeto i-on Web está dependente do projeto i-on Core que por sua vez, depende do projeto i-on Integration. Assim sendo, e encontrando-se também estes em desenvolvimento, a informação disponível ainda é escassa. Deste modo, com a finalidade de desenvolver e demonstrar o i-on Web, até que seja adicionada mais informação, procedeu-se à criação de alguns ficheiros .json com a informação num formato similar ao o que se irá obter do Core, de modo a ser possível ter dois modos de operação. Como tal, a aplicação irá também apresentar resiliência a falhas do Core, não estando totalmente dependente deste para manter funcionais algumas das suas funcionalidades mais básicas.

Os dois modos de operação existentes serão descritos em seguida.

Modo *Standalone*

Este modo consiste na obtenção de dados fictícios (*mock*) por parte da aplicação Web, ou seja, todos os dados expostos na aplicação quando esta é executada neste modo são dados fictícios provenientes de ficheiros .json.

Por omissão, a aplicação não se encontra neste modo, assim sendo, no momento da construção da imagem Docker, caso se pretenda, deve-se colocar a variável de ambiente OPERATION_MODE com o valor “standalone”.

Modo Integrado

Neste modo de operação, a aplicação obtém informação proveniente do Core, ou seja, realiza pedidos ao mesmo. Para tal, é necessário colocar o Core a correr localmente.

É ainda de salientar, que como referido anteriormente, a informação atualmente existente no Core é escassa, deste modo, para a execução das funcionalidades como pretendido, quando é recebida informação do Core esta é aprimorada para ficar mais completa. À medida que nova informação é introduzida no Core menor terão que ser os acréscimos às respostas recebidas e, eventualmente, não será necessário qualquer complemento.

Funcionalidades disponíveis

Atualmente, a aplicação i-on Web disponibiliza as seguintes funcionalidades:

- Exibição do plano curricular de múltiplos cursos, no qual é visível informação referente a cada uma das unidades curriculares, nomeadamente, o número de ECTS, a sua área científica e se as mesmas são opcionais ou obrigatórias no curso em questão;
- Exposição de informação geral sobre múltiplos cursos, nomeadamente, uma pequena introdução aos mesmos, a sua área departamental, duração, contactos e coordenação;

- Seleção de unidades curriculares por parte do utilizador, permitindo a este observar as várias turmas existentes para as disciplinas selecionadas;
- Apresentação da iniciativa i-on e indicação dos docentes e alunos envolvidos, sendo possível ter uma melhor perceção sobre o que é a iniciativa i-on e a sua finalidade, assim como demonstrar os vários projetos existentes na mesma e os alunos que contribuíram para os mesmos.

Interface de utilização

Pretende-se que a interface de utilização seja de fácil utilização, cómoda e que com poucas indirecções o utilizador consiga chegar ao pretendido. A interface será maioritariamente *server-side* e será realizada a partir de HTML para definição da estrutura e lógica de apresentação e CSS para a realização de uma interface mais apelativa. Nomeadamente, para um design mais apelativo é utilizada a *framework* Bootstrap, esta é *open-source* e auxilia no desenvolvimento de componentes de interface e *front-end* para sites e aplicações web utilizando HTML, CSS e JavaScript.

O esquema abaixo ilustra então a navegação de cliente já existente:

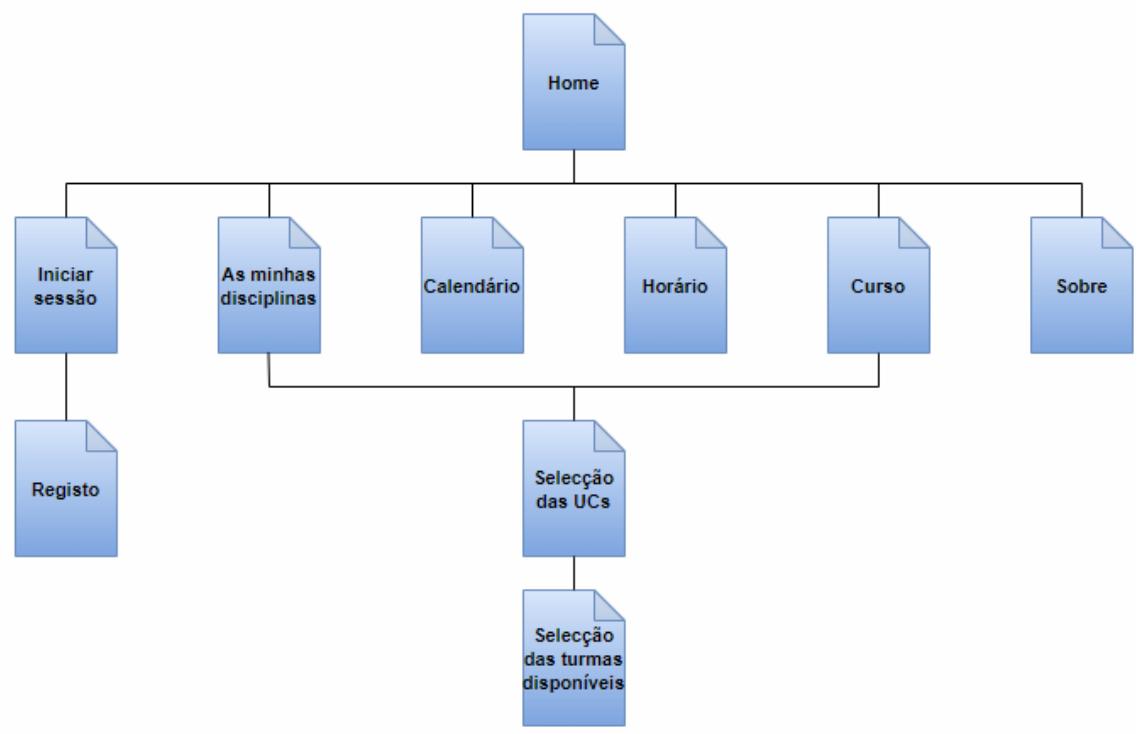

Figura 4 – Navegação de cliente existente.

Fase 3 – i-on Web online e próxima fase

I-on Web online

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 ação (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 do mesmo por parte do GitHub Actions, entre outros. Adicionalmente, contém informações associadas à conta 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.

Eventualmente, tenciona-se tirar partido dos *add-ons* existentes no Heroku, nomeadamente, do Bonsai ElasticSearch de modo a armazenar dados de sessão e preferências de utilizador, como referido anteriormente.

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

O esquema apresentado na figura 5 sintetiza a lógica existente no processo de desenvolvimento do i-on Web. Tenciona-se, brevemente, adicionar *actions* referentes à execução de testes que validem o código desenvolvido.

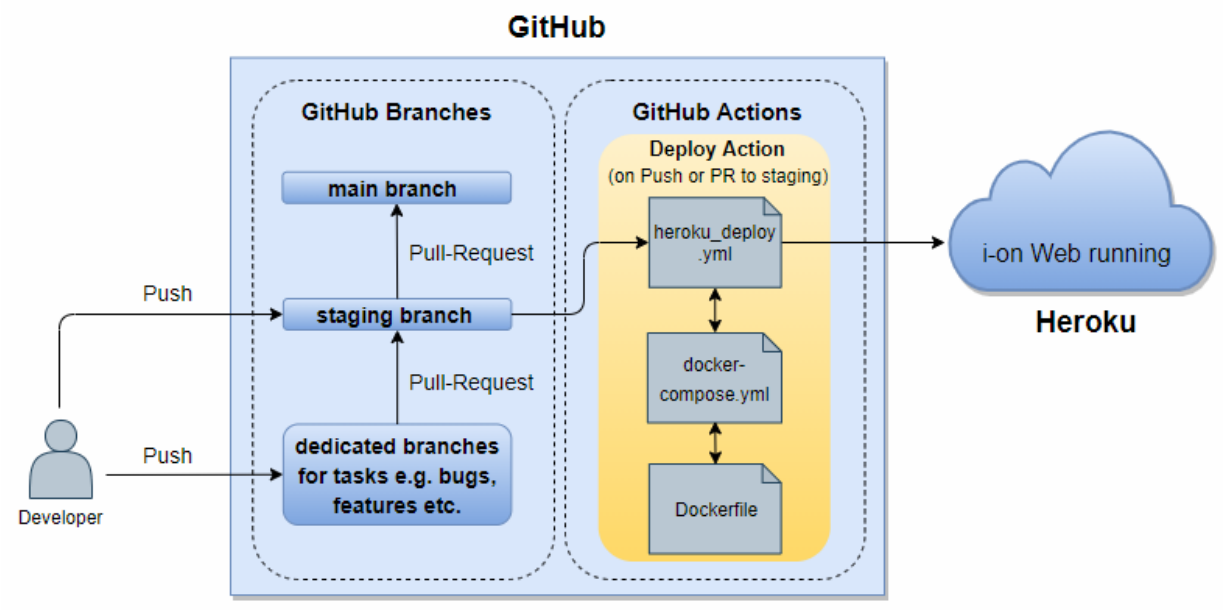

Figura 5 – Fluxo atual das várias componentes intrínsecas ao desenvolvimento do projeto.

Funcionalidades pretendidas

Numa próxima fase pretende-se continuar a implementar as funcionalidades inicialmente pretendidas assim como aprimorar algumas já realizadas e a interface de utilizador, nomeadamente, tornando-a mais apelativa.

As funcionalidades ainda pretendidas são as seguintes:

- Registo na aplicação;
- Autenticação (*login* e *logout*);
- Permitir ao utilizador anular a seleção de disciplinas;
- Construção do horário escolar do aluno (em conformidade com as disciplinas que selecionou);
- Possibilitar a visualização da lista de disciplinas selecionadas.
- Definições de utilizador que permitam a este alterar o curso, *password*, entre outras informações;
- Notificar o aluno, na página inicial, dos próximos eventos (testes, exames, entre outros) das disciplinas que frequenta;
- Construção do calendário do estudante com os eventos das disciplinas a que o mesmo se encontra inscrito;
- Alternar o idioma (entre português e inglês) da interface web;
- Possibilitar o *download* do horário do estudante.

Conclusão e aspirações para a fase final

Em virtude do que foi mencionado ao longo deste relatório é possível assimilar a importância da ambientação nas tecnologias utilizadas, não só para o desenvolvimento do projeto, mas para um futuro a nível profissional, garantindo experiência no que toca ao uso destas tecnologias frequentemente utilizadas. À semelhança das tecnologias, também o facto do projeto i-on Web estar incluído na iniciativa i-on fornece a experiência de, existindo vários projetos envolvidos, todos terem de estar em sincronia de modo a fornecer o necessário uns aos outros assim como se encontrarem funcionais em conjunto.

Referentemente ao planeamento inicialmente definido, de um modo geral, este foi cumprido, no entanto, na fase 2 de desenvolvimento do projeto houve alguns desvios relativamente ao planeamento inicial no que diz respeito às funcionalidades pretendidas.

Relativamente ao esperado numa próxima fase e na continuação do desenvolvimento do projeto espera-se implementar a maioria das funcionalidades pretendidas sendo algumas destas opcionais, mas que tornariam a aplicação i-on Web mais dinâmica e interessante.

Referências

Bootstrap team. (2021). *Build fast, responsive sites with Bootstrap*. Obtido de Bootstrap: https://getbootstrap.com

Docker Inc. (2013-2021). *Orientation and setup*. Obtido de Docker docs: https://docs.docker.com/get-started/

Docker Inc. (2013-2021). *Overview of Docker Compose*. Obtido de https://docs.docker.com/compose/

GitHub, Inc. (2021). *Docs*. Obtido de GitHub Docs: https://docs.github.com/en

Haverbeke, M. (2018). *Eloquent Javascript, A Modern Introduction to Programming.*

Heroku. (2021). *Learn about building, deploying, and managing your apps*. Obtido de Dev Center: https://devcenter.heroku.com

i-on Team. (2021). *i-on*. Obtido de GitHub organization for the i-on projects: https://github.com/i-on-project

OpenJS Foundation. (2021). *Docs*. Obtido de nodeJS: https://nodejs.org/en/docs/