IPG Politécnico da Guarda
Polytechnic of Guarda

RELATÓRIO DE PROJETO

Licenciatura em Engenharia Informática

Rui Filipe Ferreira Araújo

julhol | 2018

Escola Superior de Tecnologia e Gestão

Instituto Politécnico da Guarda

RELATÓRIO DE PROJETO EM CONTEXTO DE ESTÁGIO

RUI FILIPE FERREIRA ARAÚJO

RELATÓRIO PARA A OBTENÇÃO DO GRAU DE LICENCIADO EM ENGENHARIA INFORMÁTICA

Julho/2018

Agradecimentos

Agradeço,

ao Instituto Politécnico da Guarda, à Escola Superior de Tecnologia e Gestão e a todo o corpo docente da Licenciatura em Engenharia Informática, pelo conhecimento transmitido ao longo destes três anos de aprendizagem; ao meu professor orientador, Fernando Melo Rodrigues, por ter aceite fazer parte desta minha última etapa, antes de dar a licenciatura como terminada; aos meus colegas de curso e amigos, por me terem ensinado tanto e me terem ajudado a crescer, quer como profissional, quer como pessoa. Um agradecimento especial à minha namorada, Elsa Pereira, pela paciência e apoio prestado nesta etapa, bem como pela ajuda na hora da revisão do relatório. Um forte agradecimento ao Leandro Fernandes, ao Rui Paredes, ao Miguel Brito, ao João Elvas e à Marta Pereira; a toda a família Pereira, nomeadamente à Natália e Albino Pereira e à Carolina Pereira; por último e mais importante, à minha família, o meu pilar, que me acompanhou em todos os dias desta minha jornada. Em especial, aos meus pais, José e Clementina Araújo, e aos meus irmãos, Catarina e Daniel Araújo, por me incentivarem a ir sempre mais longe, por acreditarem em mim, por me ajudarem incansavelmente, por me darem a força para nunca desistir e lutar sempre por aquilo que quero.

A todos, o meu muito obrigado!

Ficha de identificação

Nome	Rui Filipe Ferreira Araújo
Email	ruiaraujo012@gmail.com
Nº. de Aluno	1012184
Instituição	Instituto Politécnico da Guarda
Escola	Escola Superior de Tecnologia e Gestão
Curso	Engenharia Informática
Professor Orientador	Fernando Melo Rodrigues
Empresa	Pplware
Morada	Av. Narciso Ferreira, 138 – 4º 4765-202 Riba de Ave
Contacto	915 215 491
Email	geral@pplware.com
Website	www.pplware.sapo.pt
Supervisor	Vítor Martins
Data do Estágio	14 de maio a 18 de julho
Duração	280 Horas

Resumo

O presente relatório de projeto em contexto de estágio descreve a experiência vivida na empresa *Pplware*, ao longo de um estágio de dois meses, onde foram desempenhadas funções de *web developer*, nomeadamente *web developer front end*.

Este divide-se em oito capítulos, nos quais é possível observar todo o processo de desenvolvimento do projeto.

O *Passport*, nome atribuído ao presente projeto, consiste no desenvolvimento de uma aplicação web para a gestão de viagens de utilizadores, utilizando como principais tecnologias o Reactjs e o Redux, sendo que estas usam uma linguagem de programação em comum, o JavaScript. Esta aplicação web comunica com uma API em Nodejs e Express, desenvolvida por outro membro da empresa, o que permite o tratamento dos dados.

No final é apresentada uma reflexão acerca de toda a experiência vivida durante os meses de estágio, dos conhecimentos adquiridos ao longo dos três anos de licenciatura em Engenharia Informática, bem como da importância que estes têm para dar resposta às exigências do mercado de trabalho.

Palavras-chave

front end, javascript, reactjs, redux, web developer.

Abstract

This internship project report describes the experience of the Pplware company over a two-month internship where web developer roles were developed, namely web developer front end.

This is divided into eight chapters, in which it is possible to observe the entire process of project development.

Passport, the name assigned to this project, consists of the development of a web application for user travel management, using Reactjs and Redux as its main technologies, which use a common programming language, JavaScript. This web application communicates with an API in Nodejs and Express, developed by another member of the company, which allows the treatment of the data.

At the end, a reflection on the whole experience during the internship months, the knowledge acquired during the three years of the degree in Informatics Engineering, and the importance they must meet the demands of the labor market is presented.

Keywords

front end, javascript, reactjs, redux, web developer.

Índice Geral

Índice de Figuras

Índice de Tabelas

Lista de siglas e acrónimos

API	Application Programming Interface
CSS	Cascading Style Sheets
DOM	Document Object Model
ESTG	Escola Superior de Tecnologia e Gestão
IPG	Instituto Politécnico da Guarda
POST	Power-on self-test
SVG	Scalable Vector Graphics
UC	Unidade Curricular
URL	Uniform Resource Locator

1 Introdução

O presente relatório foi desenvolvido no âmbito da UC (Unidade Curricular) de Projeto de Informática, do curso de Engenharia Informática, sendo esta lecionada na ESTG (Escola Superior de Tecnologia e Gestão) do IPG (Instituto Politécnico da Guarda).

Na unidade curricular mencionada é proposta a realização de um projeto ou projeto em contexto de estágio, deixando esta decisão ao critério do aluno.

Após a decisão da realização de um projeto em contexto de estágio, foi proposto um estágio na empresa *Pplware*, a qual entregou a tarefa de realizar um projeto denominado *Passport*.

1.1 Caraterização sumária da Instituição

A empresa *Pplware* foi fundada no dia 18 de abril de 2005 e atualmente é composta por uma equipa de nove elementos, sendo um deles o fundador e responsável da empresa.

Esta empresa conta com o maior canal de tecnologia português, bem como de fóruns, canais de educação para os mais novos e ainda uma loja online.

1.2 Motivação | Enquadramento

Atualmente verifica-se que o turismo, tanto em Portugal como em todo o mundo, tem aumentado, ou seja, há cada vez mais pessoas a viajar. De modo a facilitar a gestão do histórico de viagens dos indivíduos, surgiu a possibilidade de realizar este projeto em contexto de estágio.

Além disso, as tecnologias utilizadas para a realização do projeto são recentes e bastantes poderosas, as quais estão em grande crescimento, podendo ser muito úteis para o futuro.

Uma vez que as tecnologias abordadas ao longo da Licenciatura em Engenharia Informática não foram devidamente aprofundadas, devido aos extensos programas e ao tempo limitado para a sua abordagem, ou até mesmo o facto de outras tecnologias nem sequer terem sido lecionadas, devido ao seu recente aparecimento no mundo da informática, é reconhecida a pertinência e o enriquecimento deste projeto para o percurso académico e profissional dos seus intervenientes.

1.3 Descrição do problema

Não existe nenhuma aplicação ou *web site* que faça a gestão do histórico de viagens dos indivíduos, de forma a que estes tenham uma rápida perceção dos seus percursos, nomeadamente a apresentação de um mapa onde os países visitados aparecem realçados; a disponibilização de uma tabela que apresenta as datas das viagens, bem como os seus destinos e o feedback dos mesmos; e, possivelmente, algumas fotografias das viagens já realizadas.

Sem estes componentes torna-se difícil para o viajante lembrar-se de todas estas informações, principalmente para quem realiza muitas viagens ao longo do ano, apresentando-se este como o principal problema deste projeto, o qual tem os seus objetivos descritos abaixo.

1.4 Objetivos

Uma vez descrito o problema, é necessário traçar os objetivos do projeto, sendo eles os seguintes:

- Registo de utilizadores;
- Registo de viagens de utilizadores;
- Feedback das viagens;
- Mapa mundo com países visitados;

É de referir que este projeto é complementado com um segundo projeto desenvolvido por outro estagiário.

1.5 Plano do estágio | Etapas do estágio

Após terem sido definidos os objetivos do projeto, é agora necessário estabelecer as etapas para a sua realização.

A Figura 1 apresenta o planeamento do projeto. Dado serem tecnologias recentes, a análise e aprendizagem das mesmas decorrerá ao longo do estágio.

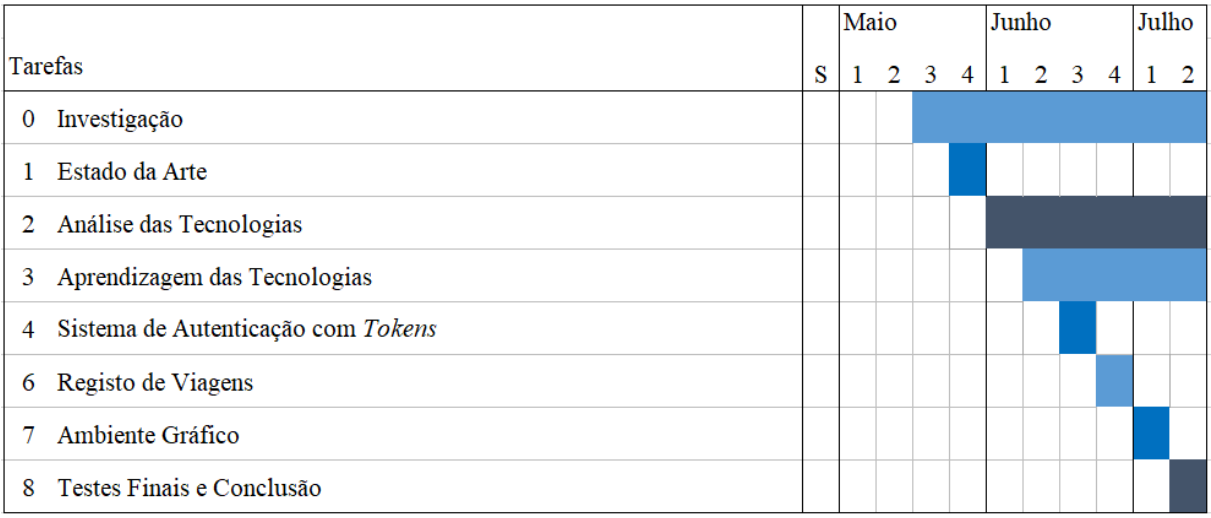

Figura 1 – Planeamento do Projeto

1.6 Estrutura do documento

Este documento encontra-se dividido em oito capítulos.

O presente capítulo, capítulo 1, faz uma introdução ao que será o projeto, falando acerca da empresa acolhedora do estágio, a motivação do projeto, o problema e os objetivos, sendo apresentado um diagrama com o planeamento e a estrutura do documento;

O capítulo 2 diz respeito ao estado da arte, onde são analisadas aplicações já existentes e comparadas com a que se pretende desenvolver, de forma a investigar a viabilidade e inovação da mesma;

No capítulo 3 é apresentada a metodologia adotada para o desenvolvimento do projeto, de forma a tornar todo o processo de desenvolvimento mais fácil e organizado;

Após isso, no capítulo 4, faz-se uma análise dos requisitos da aplicação, de forma a que todos os casos de uso sejam documentados;

No capítulo 5 são descritas todas as tecnologias utilizadas no desenvolvimento do projeto, bem como algumas das bibliotecas a que se recorreu;

No que concerne ao capítulo 6, este apresenta todo o processo de desenvolvimento e implementação do projeto, sendo apresentadas algumas imagens do resultado da implementação de cada caso de uso;

Após a implementação, é no capítulo 7 que se explica como foram realizados os testes e validações à aplicação;

Por fim, no capítulo 8, é apresentada uma reflexão final acerca de todo o trabalho realizado e quais as conclusões a que se chegou, bem como o trabalho que se pode realizar no futuro para o melhoramento da aplicação web aqui desenvolvida.

2 Estado da Arte

De modo a analisar a funcionalidade e viabilidade da aplicação *web*, pretende-se com este capítulo realizar a análise de outras aplicações ou *web sites* com funcionalidades idênticas às pretendidas com este projeto.

Após a realização de uma vasta pesquisa, foram encontrados inúmeros resultados que apresentam semelhanças ao presente projeto, os quais serão analisados de seguida.

2.1 Aplicações existentes

Das várias aplicações analisadas, foram escolhidas duas como sendo as mais completas e relevantes para a comparação, sendo elas a *TripAdvisor Attractions* [1] e o *Google Trips* [2].

2.1.1 TripAdvisor Attractions

A secção *Attractions*, presente no site e na aplicação do *TripAdvisor*, apresenta diversas funcionalidades, as quais auxiliam o utilizador na pesquisa de locais turísticos e atividades de lazer e desporto. O seu utilizador tem ainda a possibilidade de escrever e ler avaliações de outras pessoas, bem como publicar e ver fotografias.

Além disso, apresenta os preços, horários e outros detalhes acerca dos locais pesquisados.

A Figura 2 apresenta a página inicial do *TripAdvisor Attraction*, com a sua barra de pesquisa de locais.

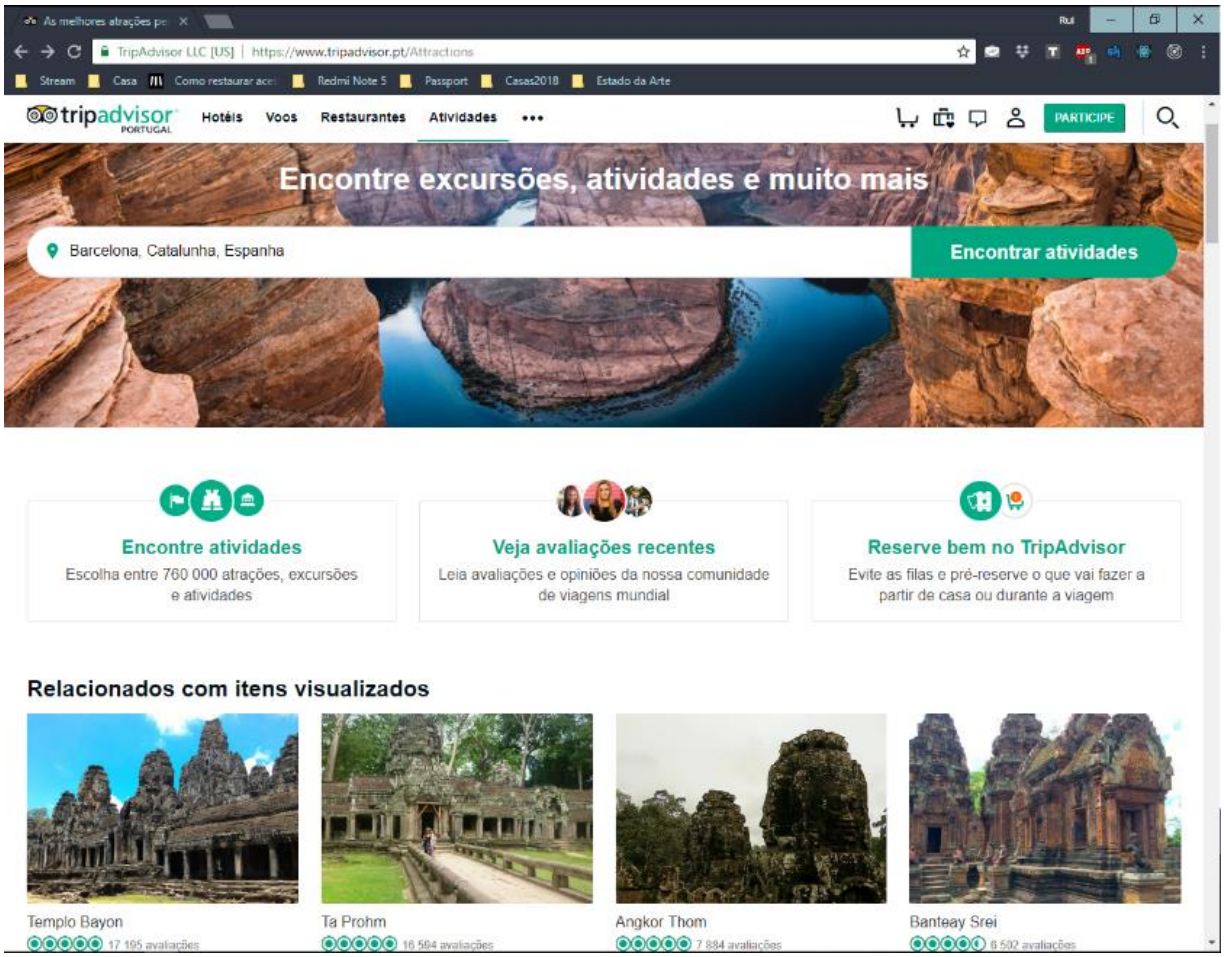

Figura 2 – Página principal TripAdvisor Attractions [1]

Um aspeto importante é o facto desta aplicação ter uma versão *web* e uma versão aplicação para *mobile*.

2.1.2 Google Trips

A aplicação *Google Trips* apresenta diversas funcionalidades, tais como o planeamento e organização de viagens; a criação de rotas para um determinado dia e local, apresentando os pontos turísticos nas proximidades.

Tal como na aplicação anterior, o *Google Trips* apresenta avaliações dos pontos de interesse, fotografias e outras informações relevantes para o seu utilizador.

Na Figura 3 são visíveis algumas das funcionalidades da aplicação.

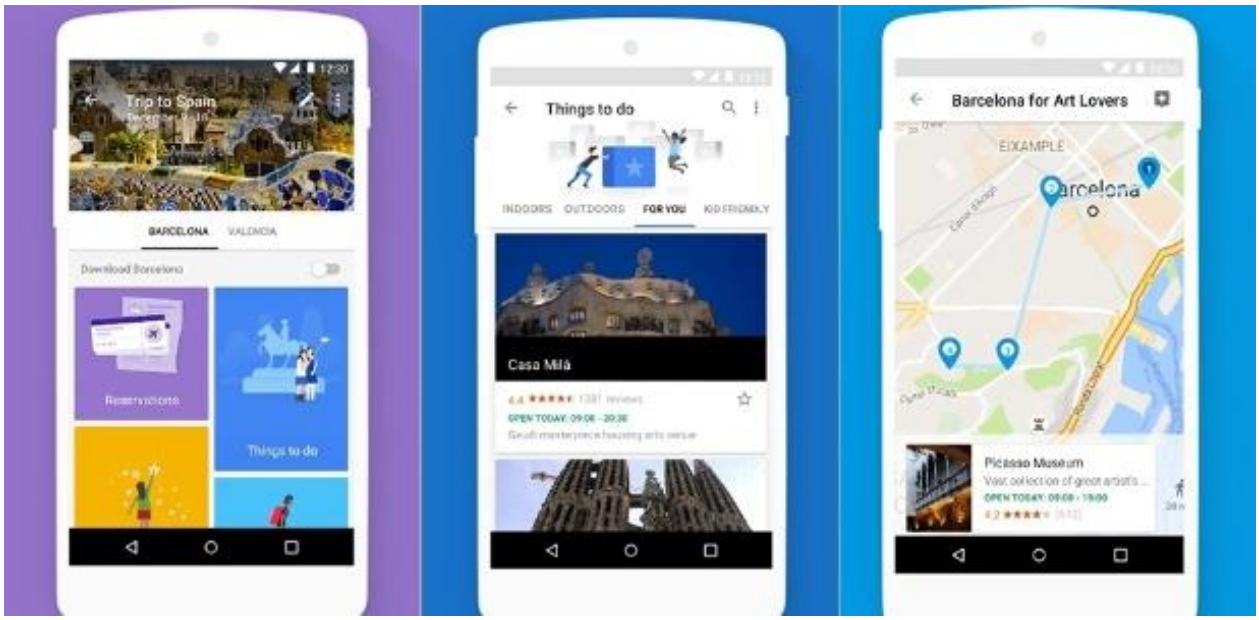

Figura 3 – Principais páginas da aplicação Google Trips [2]

Uma funcionalidade bastante relevante da aplicação é a possibilidade de transferir o plano da viagem, possibilitando, assim, o seu acesso pelo utilizador, sem a necessidade de ligação à internet.

2.2 Análise crítica das aplicações analisadas

Após uma análise detalhada das duas aplicações escolhidas, foi observado que ambas contêm funcionalidades idênticas ao pretendido com o presente projeto. No entanto, nenhuma delas reúne os requisitos necessários para a resolução do problema.

Desta forma avançou-se com o projeto, uma vez que este se apresenta como viável e inovador, pelo facto de apresentar, de forma clara, uma resolução ao problema inicial.

A Tabela 1 compara, de forma resumida, as funcionalidades das aplicações analisadas com a aplicação a ser desenvolvida.

Funcionalidades/Aplicações	**TripAdvisor Attractions**	**Google Trips**	**Passport**
Aplicação mobile	✓	✓	✗
Web Site	✓	✗	✓
Partilha de fotografias	✓	✗	✗
Mapa com países visitados realçados	✗	✗	✓
Feedback	✓	✓	✓
Seriação de utilizadores	✗	✗	✓
Métricas de comparação de utilizadores	✗	✗	✓

Tabela 1 – Comparação das funcionalidades [3]

Após a análise das duas aplicações existentes, comparativamente com a que se pretende desenvolver, é necessário escolher a metodologia que melhor se adequa às tarefas necessárias para o desenvolvimento do projeto.

3 Metodologia

De forma a minimizar os erros e a perda de tempo durante o processo de desenvolvimento de *software*, é importante o seguimento de metodologias. Ao longo deste capítulo serão descritas as metodologias utilizadas, bem como uma breve explicação das mesmas.

Dos diferentes tipos de metodologias existentes optou-se pela metodologia de desenvolvimento ágil, nomeadamente pelo processo Scrum, o qual é bastante utilizado na gestão de projetos.

A metodologia Scrum é extremamente flexível, tendo como objetivo definir um processo de desenvolvimento iterativo e incremental. Este pode ser aplicado a qualquer projeto, havendo a necessidade de comunicação entre as equipas de desenvolvimento.

Esta metodologia está ilustrada na Figura 4.

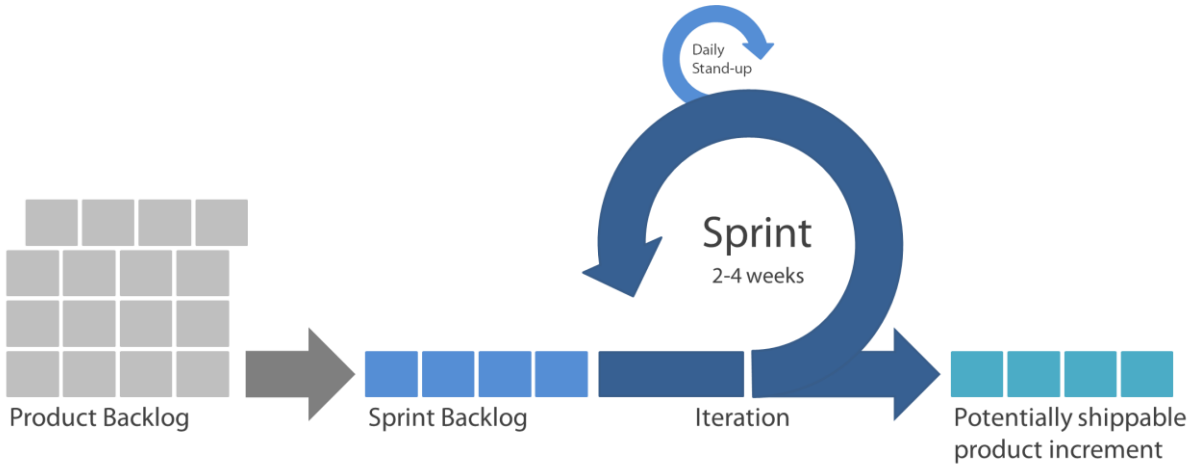

Figura 4 – Esquema da metodologia Scrum [4]

Na metodologia Scrum, os projetos são divididos em ciclos chamados de Sprints. O Sprint representa um espaço de tempo, no qual um conjunto de tarefas deve ser executada. As funcionalidades a serem implementadas no projeto são mantidas numa lista, a qual é conhecida como *Product Backlog* e, no início de cada Sprint, é realizada uma reunião, na qual o Scrum Master prioriza as tarefas do Sprint Backlog e a equipa seleciona as atividades que serão capazes de implementar durante essa Sprint.

Além disso, são realizadas reuniões diárias sobre o progresso do trabalho no projeto, com o intuito de responder a 3 questões:

- O que foi realizado, desde o dia anterior, em direção ao objetivo final da Sprint?
- O que está planeado fazer no dia seguinte para atingir o objetivo da Sprint?
- Existe algum problema a impedir de atingir o objetivo?

Por padrão, cada Sprint tem uma duração de 2 a 4 semanas, contudo, neste caso, cada Sprint teve a duração de entre 1 a 2 semanas, uma vez que o espaço temporal para a realização do projeto era reduzido.

4 Análise de requisitos

Os requisitos de um sistema são toda a informação que descreve as funcionalidades da aplicação, antes de esta ser desenvolvida, de forma a minimizar os erros e evitar a criação de uma aplicação incompleta.

Desta forma, neste capítulo irão analisar-se todos os requisitos funcionais e não funcionais para o desenvolvimento desta aplicação, o diagrama de contexto, o diagrama de casos de uso e a sua respetiva descrição, bem como o diagrama de sequência, o diagrama de classes e, por fim, o dicionário de dados.

Após a análise das necessidades do projeto e, posteriormente, em reunião com os representantes da empresa acolhedora do estágio, foram encontrados requisitos funcionais e não funcionais, sendo que estes serão descritos de seguida.

4.1 Requisitos funcionais

Os requisitos funcionais são a descrição das diversas funcionalidades que o cliente ou os utilizadores desejam que o software disponibilize. Como principais requisitos funcionais foram selecionados os seguintes:

- Registo de utilizadores;
- Confirmação de email nos registos;
- *Login* de utilizadores;
- Associar conta às Redes Sociais;
- Partilhar viagens nas Redes Sociais;
- Inserção de países visitados para utilizadores registados;
- Consultar países visitados para utilizadores registados;
- Ver mapa mundo com países visitados para utilizadores registados;
- Inserir *feedback* na viagem realizada.

4.2 Requisitos não funcionais

Os requisitos não funcionais são as qualidades gerais do software, como a usabilidade e desempenho da aplicação. Como requisitos não funcionais foram selecionados os apresentados de seguida:

- Criação de um *web site* responsivo;
- Validação de campos nos formulários;
- Otimização para melhorar a performance.

4.3 Diagrama de contexto

O diagrama de contexto é constituído por fluxos de dados entre o sistema e as entidades externas, os utilizadores do sistema. Este fluxo representa a interação dos utilizadores com o sistema e vice-versa. Desta forma, a Figura 5 representa essa interação.

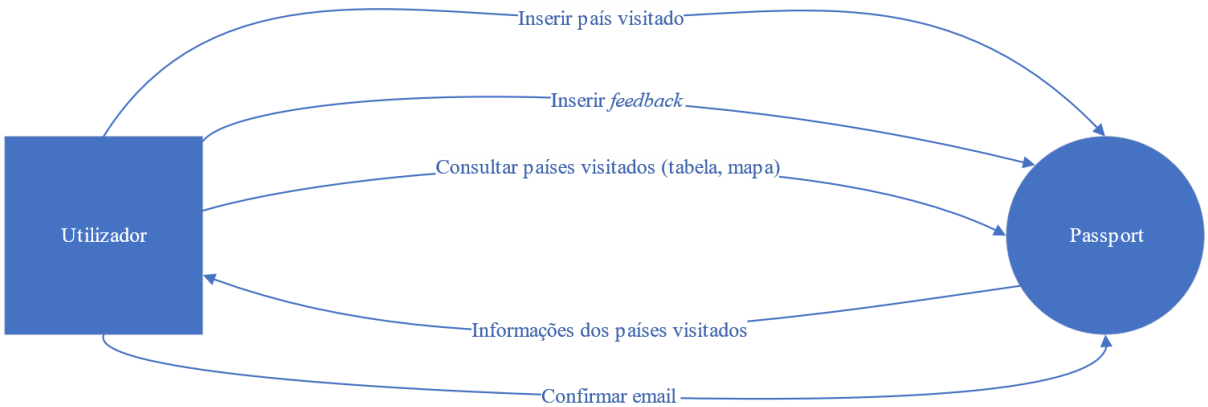

Figura 5 – Diagrama de Contexto

4.4 Diagrama de casos de uso

Os casos de uso são úteis para o auxílio na análise de requisitos do sistema. Cada caso de uso corresponde a um requisito do sistema. Seguidamente, na Figura 6 estão representados os casos de uso para este projeto.

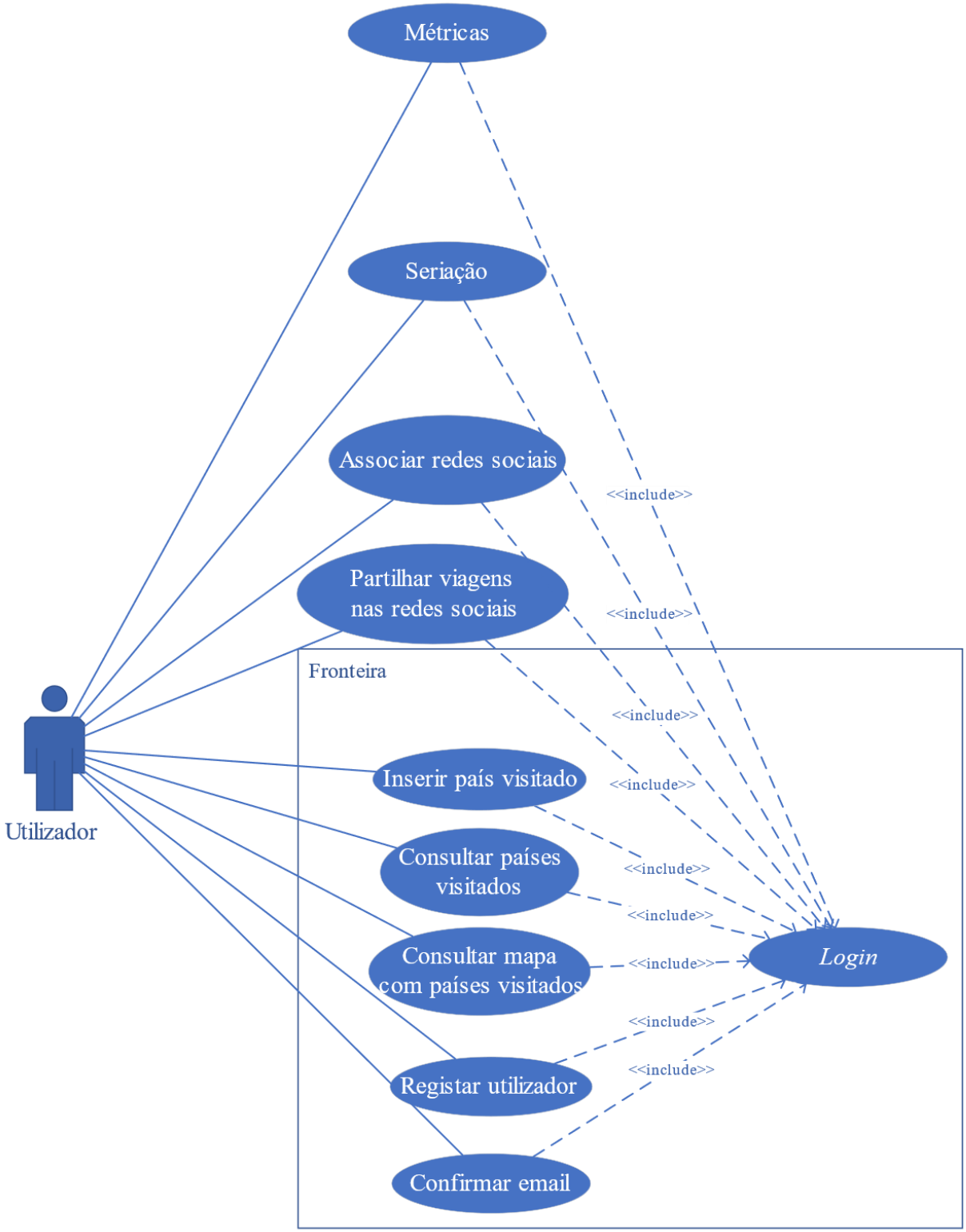

Figura 6 – Diagrama de casos de uso

4.5 Descrição de casos de uso

Na descrição de um caso de uso é apresentada a sequência de eventos fundamentais que um ator deve realizar para concluir um processo, o qual se espera que corra bem (caminho principal). No caso de algo não correr como o esperado, é necessário haver alternativas, de modo a que o ator não fique preso nesse passo (caminho alternativo).

De seguida são descritos os dois principais casos de uso do projeto, os quais são inserir um país visitado e consultar o mapa com os países já visitados, sendo que os restantes se encontram no anexo A.1.

4.5.1 Inserir país visitado

Nome: Inserir país visitado;

Descrição: Inserir um país visitado, com datas e com uma imagem. Inserir também um *feedback*.

Pré-condições: Ser utilizador registado no site com email confirmado.

Caminho principal:

1. O utilizador seleciona no menu a opção "Dashboard";
2. O sistema apresenta um botão para adicionar uma nova viagem e uma tabela com as viagens existentes;
3. O utilizador clica no botão "Add new travel";
4. O sistema apresenta um formulário com dados para a inserção de uma nova viagem;
5. O utilizador seleciona as datas da viagem, o país visitado, escreve um feedback e clica no botão "Add Travel";
6. O sistema insere a viagem na base de dados.

Caminhos alternativos:

2a. O sistema apresenta apenas um botão para adicionar uma nova viagem caso o utilizador ainda não tenha viagens.

5a. O utilizador clica no botão "Add Travel" mas deixa campos por preencher;

6a. O sistema rejeita a inserção da viagem, apontando quais os campos em falta, para que a submissão seja aceite.

Suplementos ou adornos:

1. O formulário deve permitir a inserção de feedback acerca da viagem.

Pós-condição:

Redirecionar o utilizador para a página "Dashboard".

4.5.2 Consultar mapa com países visitados

Nome: Consultar mapa com países visitados;

Descrição: Consultar, num mapa mundo, os países já visitados, estes aparecerão com realce perante os ainda não visitados.

Pré-condições: Ser utilizador registado no site com email confirmado.

Caminho principal:

2. O utilizador seleciona no menu a opção "Home";
3. O sistema apresenta um mapa mundo com os países já visitados com realce;

Caminhos alternativos:

2a. O sistema apresenta apenas um mapa mundo vazio caso o utilizador ainda não tenha viagens.

Suplementos ou adornos:

1. O mapa deve permitir que o país onde o cursor esteja, fique realçado.

Pós-condição:

4.6 Diagrama de sequência

Os diagramas de sequência descrevem como e em que ordem um grupo de objetos funciona. Estes diagramas são utilizados para analisar as necessidades do sistema, sendo assim possível observar o fluxo de mensagens, ações e eventos entre os objetos.

De forma a definir a sequência de eventos da aplicação web, serão apresentados, seguidamente, os diagramas de sequência dos casos de uso descritos anteriormente, sendo que os restantes se encontram no anexo A.2.

4.6.1 Inserir país visitado

Na Figura 7 encontra-se representado o diagrama de sequência para o caso de uso, inserir país visitado.

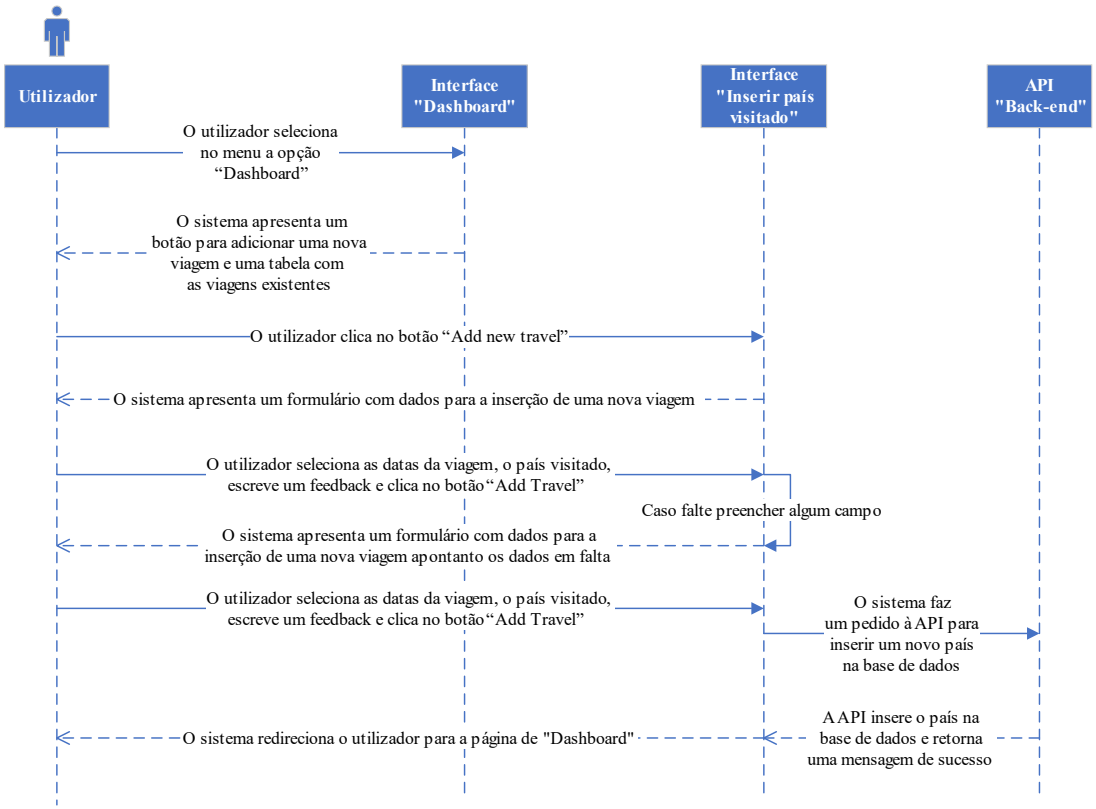

Figura 7 – Diagrama de sequência inserir país visitado

4.6.2 Consultar mapa com países visitados

De seguida, na Figura 8, é visível o diagrama de sequência do caso de uso, consultar mapa com países visitados.

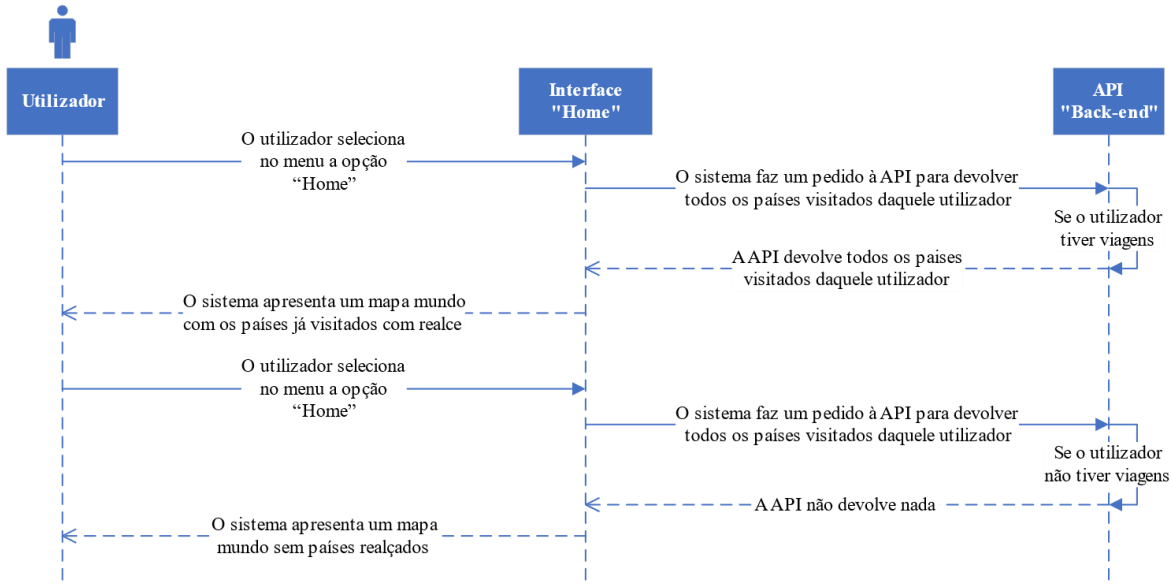

Figura 8 – Diagrama de sequência consultar mapa com países visitados

4.7 Diagrama de classes

Segundo João Pascoal Faria, "um diagrama de classes serve para modelar o vocabulário de um sistema, do ponto de vista do utilizador/problema ou do implementador/solução" [5]. Este diagrama mostra a relação entre as diferentes classes presentes no sistema a desenvolver.

Para além de serem visíveis as relações entre as classes, neste diagrama é também possível ver o seu nome; os atributos, ou seja, a informação que deverá ser armazenada e/ou analisada; e, as possíveis operações que os utilizadores têm no sistema.

Este projeto conta apenas com a presença de duas classes, representadas na Figura 9, que, embora sendo poucas, permitem satisfazer os objetivos.

As duas classes são a classe dos "Utilizadores", onde serão guardados todos os dados do utilizador e os dados para o *login*; e, a classe das "Viagens", onde se encontram todos os dados das viagens realizadas pelos utilizadores.

Figura 9 – Diagrama de classes da aplicação web

4.8 Dicionário de dados

O dicionário de dados não é nada mais do que a explicação do dicionário de classes, ou seja, descrever todos os campos, dizendo qual o tipo de dados desse campo; a descrição; quais os valores válidos; e, as restrições.

Todos estes aspetos estão representados nas tabelas seguintes.

4.8.1 Utilizadores

A Tabela 2 descreve os campos relativos à tabela dos utilizadores. É de salientar que, alguns dos campos, nomeadamente o `CreatedAt` e `UpdatedAt`, são criados automaticamente pela base de dados utilizada, assim como o campo `PasswordHash` que é o *hash* da password, de forma a que estas sejam guardadas de forma protegida.

Nome do Campo	Tipo de Dados	Descrição	Valores Válidos	Restrições
ID_User (PK)	ObjectId	Chave primária gerada automaticamente pela base de dados;	Letras e números	Único; Criado pelo sistema; Não alterável;
FirstName	String	Corresponde ao primeiro nome do utilizador;	Letras e espaços	Obrigatório;
LastName	String	Corresponde ao último nome do utilizador;	Letras e espaços	Obrigatório;
Email	String	Corresponde ao email do utilizador;	Letras, números, pontos e arroba	Obrigatório;
BirthdayDate	Date	Corresponda à data de nascimento do utilizador;	Datas	Obrigatório; Idade mínima de 16 anos;
PasswordHash	String	Corresponde ao Hash da password do utilizador;	Letras, números, pontos e símbolos;	Obrigatório;
Confirmed	Boolean	Corresponde ao estado de confirmação da conta do utilizador;	True ou false	Alterável pelo sistema;
ConfirmationToken	String	Corresponde ao token para validar a conta;	Letras, números, pontos e símbolos	Único; Gerado pelo sistema;
CreatedAt	Date	Corresponde à data de criação da conta do utilizador;	Datas	Não alterável; Gerado pelo sistema;
UpdaterAt	Date	Corresponde à data de alteração da conta do utilizador;	Datas	Alterável; Gerado pelo sistema;

Tabela 2 – Dicionário de dados da classe Users

4.8.2 Viagens

A Tabela 3 descreve os campos relativos à tabela das viagens. De forma a evitar o envio de pedidos ao servidor, a chave estrangeira das viagens não é o ID do utilizador, mas sim o email, uma vez que este se encontra no *token*.

Nome do Campo	Tipo de Dados	Descrição	Valores Válidos	Restrições
ID_Travel (PK)	ObjectId	Chave primária gerada automaticamente pela base de dados;	Letras e números	Único; Criado pelo sistema; Não alterável;
Email (FK)	String	Corresponde ao email do utilizador;	Letras, números, pontos e arroba	Obrigatório;
CheckIn	Date	Corresponda à data de inicio da viagem	Datas	Obrigatório; Datas anteriores à data do sistema;
CheckOut	Date	Corresponda à data de fim da viagem	Datas	Obrigatório; Datas anteriores à data do sistema;
Country	String	Corresponda ao país visitado	Letras e espaços	Obrigatório;
CountryCode	String	Corresponda ao código do país visitado	Leras e espaços	Obrigatório; Inserido automáticamente com base no país selecionado;
Feedback	String	Corresponda ao feedback da viagem	Letras, números, espaços e simbolos	Obrigatório;

Tabela 3 – Dicionário de dados da classe Travels

Após terem sido desenhados todos os diagramas acima apresentados, foram reunidos todos os elementos essenciais para a resolução do problema.

É agora necessário escolher a linguagem que melhor se adequa para o desenvolvimento do projeto, seleção realinhada no capítulo seguinte.

5 Tecnologias

Neste capítulo serão mencionadas as tecnologias escolhidas para o desenvolvimento do projeto, bem como uma breve explicação da razão pela qual foram selecionadas.

Dado tratar-se de uma aplicação para a *web*, foi escolhida como linguagem principal do projeto o *JavaScript*. No entanto, foram usadas bibliotecas de forma a aumentar o poder desta tecnologia.

Uma vez escolhida a linguagem, foram escolhidas duas tecnologias bastantes completas e de fácil uso, sendo elas o *ReactJS* e o *Redux*. Foram também usadas outras tecnologias e bibliotecas para auxiliar o desenvolvimento, as quais serão mencionadas abaixo.

5.1 JavaScript

O *JavaScript* é "uma linguagem de programação interpretada. Foi originalmente implementada como parte dos navegadores web para que scripts pudessem ser executados do lado do cliente e interagissem com o usuário sem a necessidade deste script passar pelo servidor, controlando o navegador, realizando comunicação assíncrona e alterando o conteúdo do documento exibido." [6]

5.2 ReactJS

Segundo a documentação do *React* [7], esta linguagem é declarativa, o que facilita a criação de interfaces de utilizador interativas, sendo baseada em componentes, o que permite criar componentes encapsulados que fazem a gestão do próprio estado. Além disso, esta linguagem é escrita em *JavaScript.*

Sendo esta uma tecnologia com bastante potencial e de fácil aprendizagem, foi escolhida para criar todas as interfaces do utilizador.

5.3 Redux

Sendo o *ReactJS* uma tecnologia poderosa, combinada com o *Redux*, torna-se mais eficaz e organizado o desenvolvimento de aplicações para a *web*. Esta ferramenta permite lidar com os estados dos componentes criados em *ReactJS*, tornando o fluxo de dados mais rápido.

O *Redux* está dividido em 4 secções, ilustradas no esquema de funcionalidades apresentado na Figura 10, sendo elas:

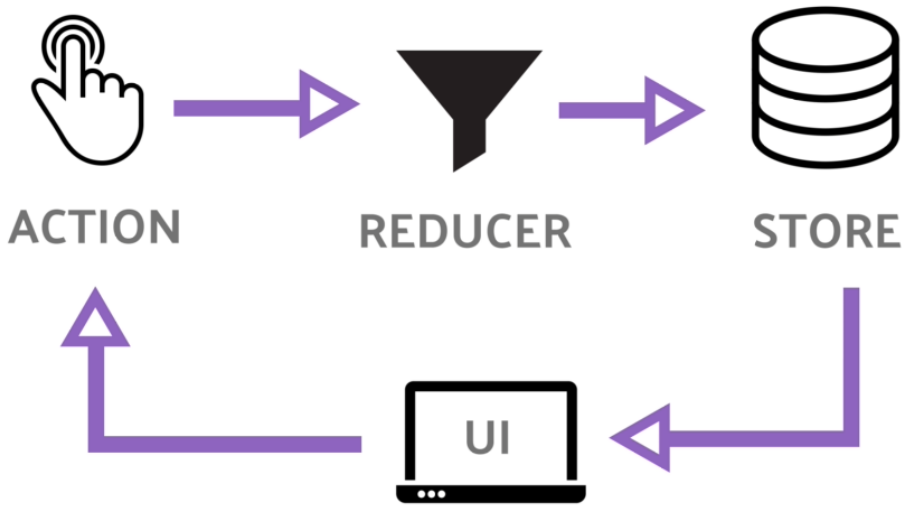

Figura 10 – Esquema de funcionamento do Redux [8]

5.3.1 Action

A única maneira de mudar o estado da aplicação é quando há uma ação. Quando uma ação é disparada, esta tem detalhes dela própria, tais como o seu tipo e os parâmetros.

5.3.2 Reducer

Após ocorrer uma ação, é necessário que algo aconteça. Os *Reducers* são responsáveis por criar um estado novo da aplicação com base na ação.

5.3.3 Store

A *Store* guarda os estados da aplicação e só ela os pode alterar através de ações. Além disso, é possível recuar no histórico de estados, já que a *Store* guarda todos os estados por ordem cronológica.

5.3.4 Render (UI)

Depois da ação ter sido disparada, ter sido alterado o estado e guardado na *Store*, é possível agora alterar a página da aplicação com os novos estados.

5.4 Yarn

O *Yarn* é um gestor de pacotes para JavaScript, sendo também usado para instalar, partilhar, distribuir código e gerir dependências no projeto, entre outras funcionalidades.

Em alternativa ao *Yarn* poderia ser usado o NPM, visto ser mais completo. No entanto, para este projeto, o *Yarn* é mais apropriado, uma vez que é mais leve e, consequentemente, mais rápido, além de ser *Open-Source*.

5.5 GitHub

Uma vez que este projeto tem como principal objetivo a criação de uma aplicação *web*, é necessário programar várias linhas de código.

Dado ser uma equipa de dois elementos, muitas vezes é necessário partilhar o código desenvolvido. A pensar nestas situações foi usado o *GitHub*, como repositório para todo o código desenvolvido durante o estágio. Assim, os testes foram facilitados, uma vez que toda a equipa tinha acesso ao mesmo repositório.

Algumas das funcionalidades desta ferramenta estão representadas de forma sintética na Tabela 4

Vantagens	Descrição
Criação de branches	Possibilidade de criar um brach, o que permite a criação de código em paralelo com o que já existe, sem que este seja alterado.
Merge	Adição de um brach criado ao brach principal do projeto.
Histórico de versões	Possibilidade de retroceder a um ponto no passado.

Tabela 4 – Algumas funcionalidades da plataforma GitHub

5.6 Dependências

Na Tabela 5 são apresentadas todas as dependências do projeto, bem como a versão utilizada e ainda uma breve descrição das mesmas. As dependências são geralmente bibliotecas que

auxiliam o desenvolvimento do projeto, permitindo assim acrescentar funcionalidades à aplicação.

Dependência	Versão	Breve Descrição
axios	0.18.0	Cliente HTTP baseado em promessas para o navegador e node.js
eslint	4.19.1	Analista de código JavaScript procurando encontrar erros.
jwt-decode	2.2.0	Biblioteca que ajuda a decodificar JSONWebTokens que são codificados em Base64Url
prettier	1.13.5	Formatador de código
prop-types	15.6.1	Verificador do tipo de execução para objetos React
react	16.4.1	Descrito no capítulo 5.2
react-dom	16.4.1	Biblioteca do react para trabalhar com o DOM (Document Object Model)
react-md	1.4.1	Biblioteca de estilos para editar o calendário
react-redux	5.0.7	Ligações entre react e redux
react-router	4.3.1	Rotas declarativas para react
react-router-dom	4.3.1	Rotas declarativas para react com ligações com o DOM
react-scripts	1.1.4	Configuração e *scripts* para create-react-app
react-simple-maps	0.12.1	Mapa SVG (Scalable Vector Graphics) contruído e usado no react
redux	4.0.0	Descrito no capítulo 5.3
redux-devtools-extension	2.13.2	Biblioteca para auxiliar o redux
redux-thunk	2.3.0	*Thunk middleware* para redux.
semantic-ui-css	2.3.1	Distribuidor de CSS(Cascading Style Sheets) para Semantic UI
semantic-ui-react	0.81.1	Biblioteca de estilos para react
validator	10.3.0	Biblioteca para validação de expressões
yarn	1.7.0	Descrito no capítulo 5.4-

Tabela 5 – Dependências do projeto

Uma vez que todas as tecnologias usadas foram descritas, pode-se agora descrever a implementação da solução. Esta foi implementada usando todas as tecnologias e dependências mencionadas anteriormente.

6 Implementação

Neste sexto capítulo será explicado todo o processo de implementação da solução ao problema inicial.

Para isso, foi necessário criar sete módulos, de forma a facilitar e organizar o projeto. Estes módulos estão inumerados por ordem cronológica e são o *login*, o *lignup*, a implementação de tokens, a criação de rotas autorizadas, a confirmação de email, a inserção de países visitados e a implementação de um mapa mundo.

Uma vez que as tecnologias utilizadas são recentes, foi necessária muita pesquisa e aprendizagem. Para este efeito foi usada a documentação oficial das linguagens e alguns tutoriais, de forma a consolidar os conceitos necessários ao desenvolvimento do projeto. No entanto, houve um primeiro contacto com algumas destas tecnologias na UC de Sistemas Distribuídos, o que facilitou a aprendizagem.

6.1 Login

Iniciou-se o projeto com a implementação do mecanismo de *login*, uma vez que, quando um utilizador cria uma conta, faz automaticamente *login* na aplicação, no caso de todos os dados terem sido validados.

Como se pode observar na Figura 11, para se fazer login são necessários apenas o *email* e a *password*. Estes dados não são tratados no *front-end*, no entanto, aqui são validados se os campos estão vazios, evitando fazer pedidos desnecessários ao servidor. No caso de os campos estarem preenchidos, é enviado um pedido à API (*Application Programming Interface*) que valida os dados e, no caso de tudo estar correto, o utilizador faz *login* e pode começar a utilizar a aplicação.

Encontra-se no anexo A.3.1 o código respetivo à criação deste formulário.

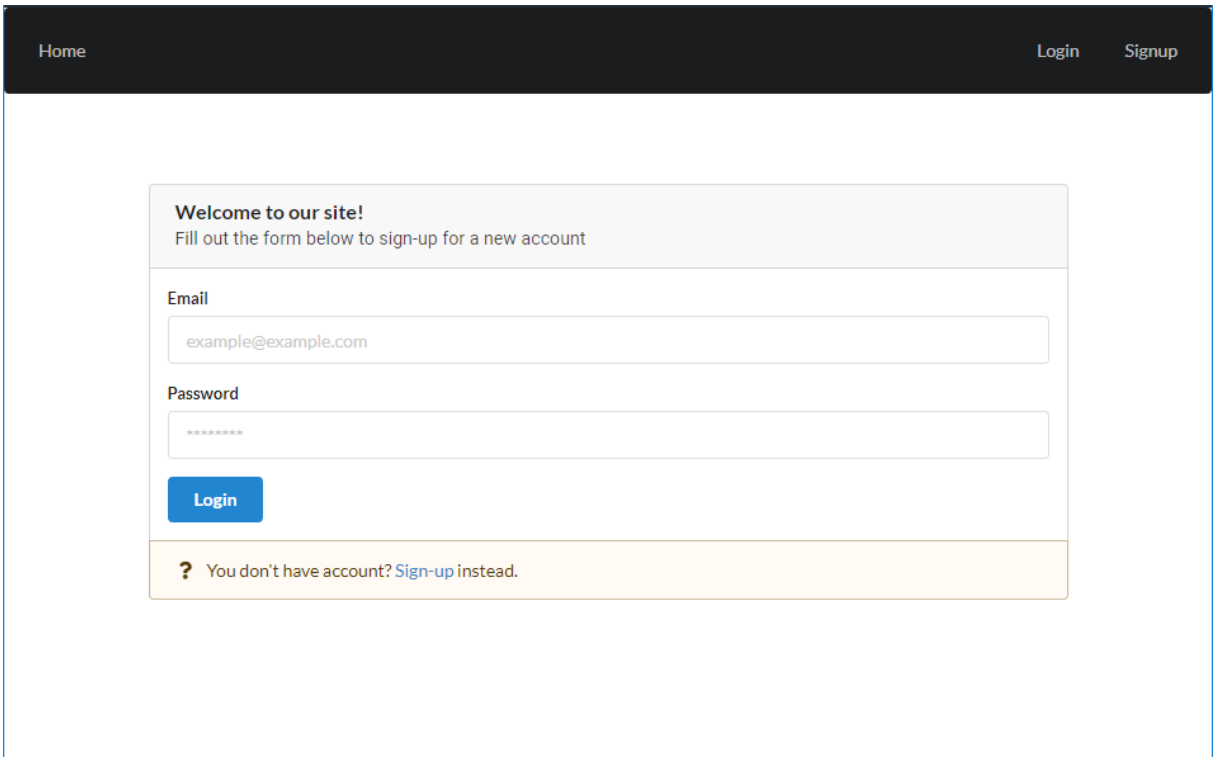

Figura 11 – Página de login da aplicação web

6.2 Signup

Depois do módulo de *Login* estar implementado e a funcionar, foi inserido o módulo de *Signup*, permitindo assim que os utilizadores pudessem criar conta.

Neste módulo, e como é visível na Figura 12, o utilizador terá de inserir o seu nome, a data de nascimento, o *email*, a *password* e a confirmação da *password*. Estes últimos campos serão usados para que o utilizador possa fazer *login* posteriormente na aplicação.

Tal como no módulo anterior, na aplicação *web* não serão validados dados para além dos campos vazios, no caso da confirmação de *password* verificar se é igual à *password* e, no caso do *email*, verificar se os dados inseridos coincidem com o modelo padrão dos *emails*.

Encontra-se no anexo A.3.2 o código respetivo à criação deste formulário.

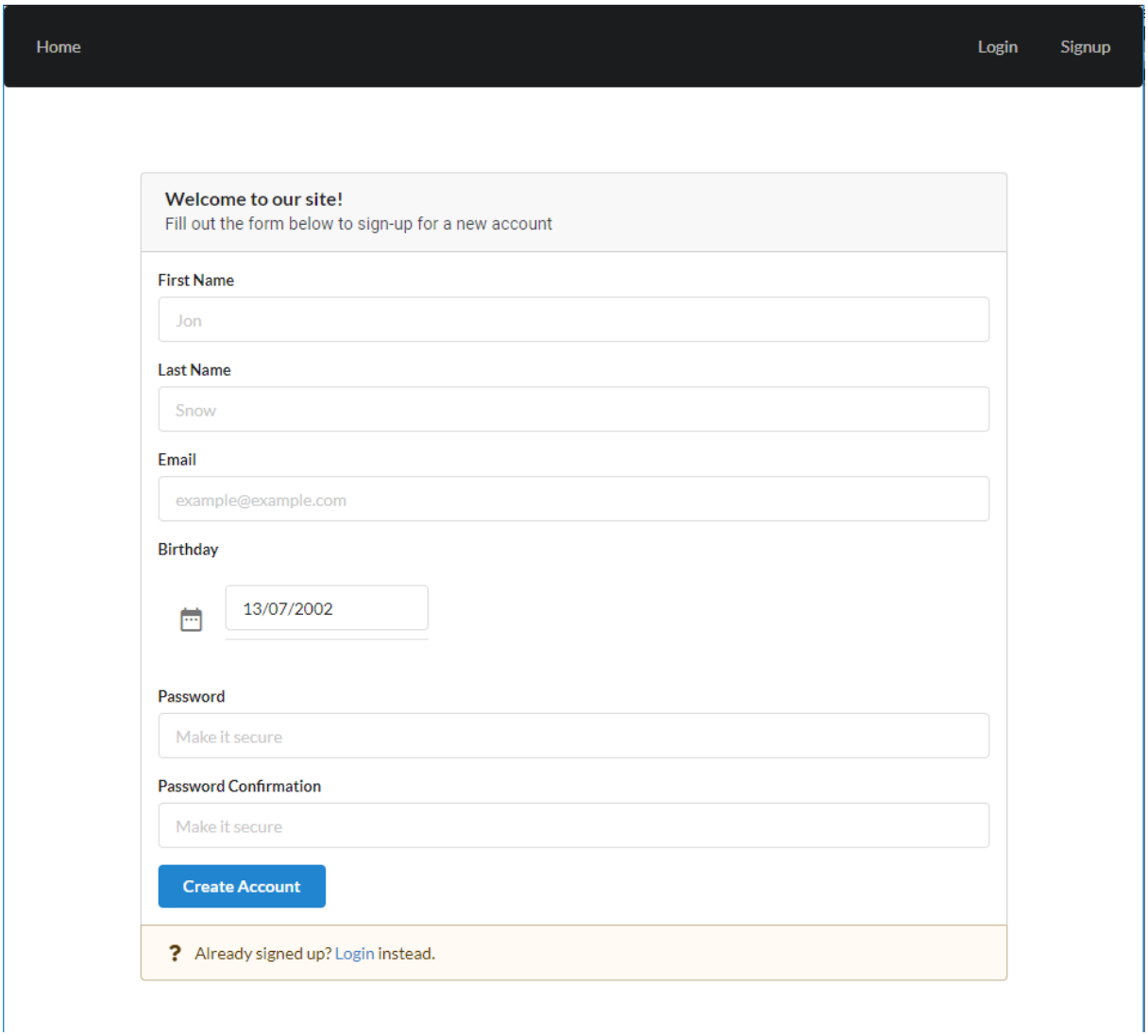

Figura 12 – Página de signup da aplicação web

6.3 Implementação de tokens

Após estarem concluídos os módulos referentes aos utilizadores, foi necessário implementar *tokens* com o intuito de tornar a aplicação mais segura e, além disso, como o *token* contém informações do utilizador, como o seu email por exemplo, poderá ser útil como forma a evitar pedidos desnecessários à API.

Estes *tokens* são gerados pela API quando o utilizador faz *login* ou quando se regista. Além disso, o *token* gerado é sempre diferente para cada sessão. Este *token*, uma vez gerado pela API, é enviado para a aplicação *web* e guardado no *LocalStorage* do navegador até que o utilizador faça *logout*, onde este é removido.

De forma a garantir que apenas utilizadores registados possam realizar certas operações na aplicação *web*, sempre que se faz um pedido à API é enviado no cabeçalho do pedido o *token* do utilizador. Assim, a API poderá validar o *token* e, se este for válido, responde ao pedido. A Figura 13 mostra o exemplo de um pedido POST (Power-on self-test) ao servidor com o *token* no cabeçalho.

Encontra-se no anexo A.3.3 o código respetivo à implementação dos *tokens*.

Figura 13 – Representação padrão de um pedido POST

6.4 Criação de rotas autorizadas

Embora a aplicação já tenha *tokens* implementados, um utilizador não registado ainda consegue aceder a páginas que apenas dizem respeito a utilizadores registados, como é o caso da página de "dashboard". Basta, para isso, colocar no URL (*Uniform Resource Locator*) à frente do domínio "/dashboard".

Para evitar que isso aconteça, foram criadas rotas para os utilizadores registados e rotas para os não registados. Desta forma, sempre que o utilizador tente aceder a uma página não autorizada, é redirecionado para a página principal, no caso de ser um utilizador não registado, ou para a página de "dashboard", no caso de ser um utilizador registado.

Encontra-se no anexo A.3.4 o código respetivo à implementação das rotas autorizadas.

A Figura 14 seguinte mostra a implementação dessas rotas.

```
<GuestRoute location={location} path='/login' exact component={LoginPage} />
<GuestRoute location={location} path='/signup' exact component={SignupPage} />
<UserRoute location={location} path='/dashboard' exact component={DashboardPage} />
<UserRoute location={location} path='/travels/add' exact component={AddTravelPage} />
```

Figura 14 – Código da implementação das rotas autorizadas

6.5 Confirmação de email

De forma a tornar a aplicação realista, é necessário sempre que o utilizador se regista, confirmar o email. No caso de este não confirmar, fica interdito de inserir países visitados, no entanto, assim que confirmar, esta funcionalidade fica visível.

A forma de confirmação é o envio de uma hiperligação para o email, com um *token*, sendo que o utilizador apenas terá de copiar esta hiperligação e colar no navegador. Assim que colar, a aplicação *web* faz um pedido à API para validar o *token* e, caso este esteja correto, o utilizador confirma a conta e pode agora inserir países visitados. Caso contrário, terá de repetir o passo com uma hiperligação válida.

Encontra-se no anexo A.3.5 o código respetivo à página de confirmação de email.

A Figura 15 mostra um exemplo de uma hiperligação enviada para o email.

```
http://localhost:3000/confirmation/eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlb
WFpbCI6Impvc2UxMjNAZ21haWwuY29tIiwiY29uZmlybWVkIjpmYWxzZSwiaWF0IjoxNTMxNDg4ND
kxfQ.PpDgfcBN6DPWiZd7JBEBm2fwaxNe0GfUIOXy3YjAqqw
```

Figura 15 – Exemplo de uma hiperligação de confirmação

6.6 Inserção países visitados

Neste módulo tem-se como objetivo permitir que os utilizadores registados e com email confirmado possam inserir um país visitado. Para isso, foi criado um formulário ilustrado na Figura 16 que permite que o utilizador insira as datas da visita ao país, selecionar o país a partir de uma lista de todos os países e, por fim, deixar um *feedback* da viagem.

Assim que o utilizador clica no botão “Add Travel”, a aplicação *web* faz um pedido ao servidor para que este insira a nova viagem na base de dados. Aqui, para associar a viagem inserida ao utilizador, é usado o *token* onde é extraído o email do utilizador e inserido na base de dados, juntamente com a viagem.

Após a viagem ser inserida, o utilizador é redirecionado para a página "dashboard", onde é apresentada uma tabela com todos os países visitados desse utilizador.

Encontra-se no anexo A.3.6 o código respetivo ao formulário de inserção de países visitados.

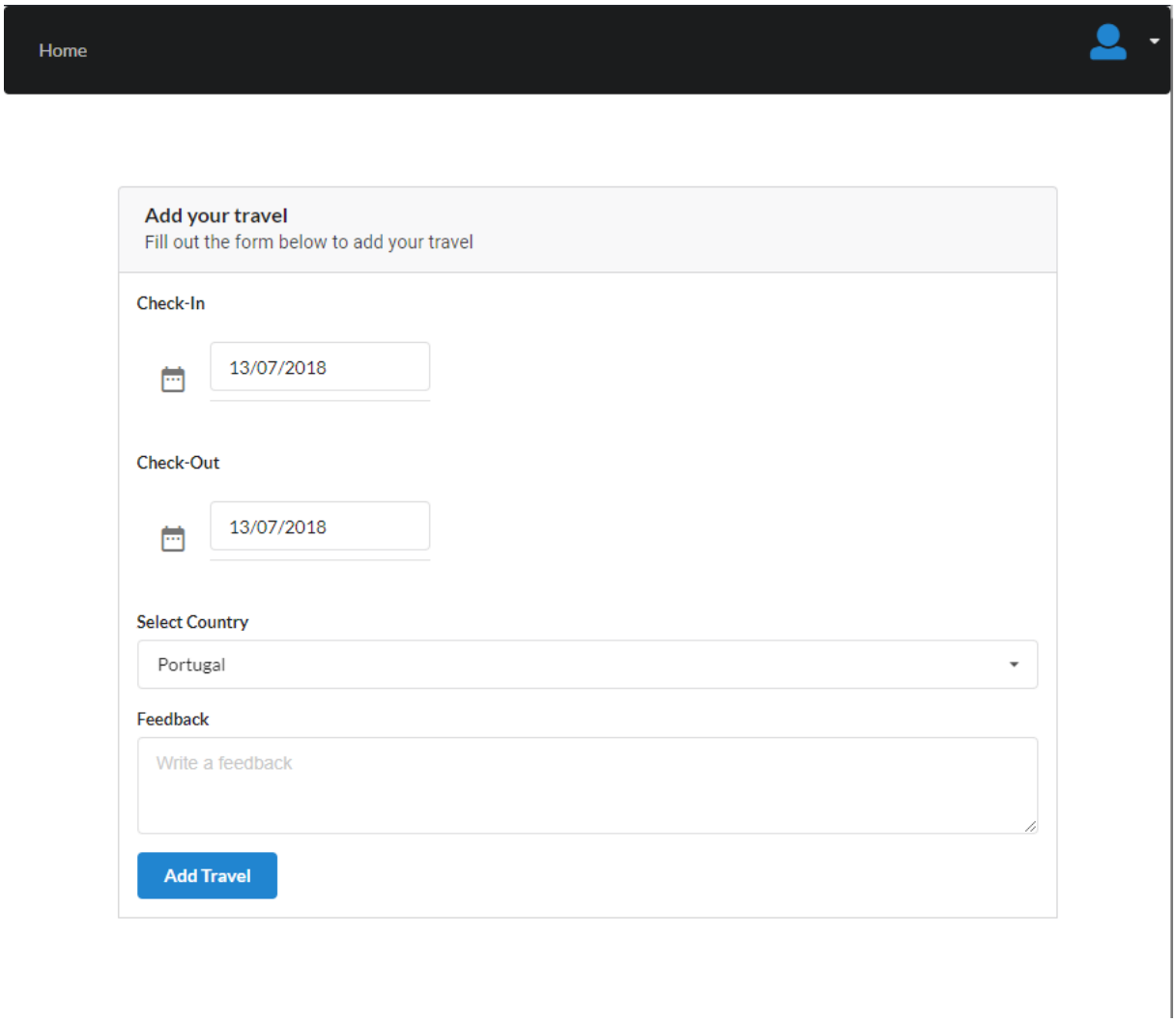

Figura 16 – Formulário para inserção de país visitado.

6.7 Implementação de um mapa mundo

De maneira a que seja possível visualizar os países visitados de uma forma facilitada e simples, optou-se pela implementação de um mapa mundo vetorial, onde os países já visitados aparecem realçados.

Para a implementação deste mapa foi necessária a utilização de uma biblioteca destinada ao efeito, já descrita neste documento na secção 5.6, sendo ela a biblioteca *react-simple-maps*.

Esta biblioteca permitiu, de uma forma simples, a alteração das propriedades do mapa, possibilitando a criação de um mapa com realces.

Este mapa encontra-se na página *home* e aparecerá sem dados se o utilizador não estiver registado, ou ainda não tiver países registados. No caso de o utilizador ter viagens, a aplicação *web* faz um pedido à API para devolver todas as viagens daquele utilizador, e, assim que a API enviar a resposta, aparecerá no mapa os países visitados com realce, como mostra a Figura 17.

Encontra-se no anexo A.3.7 o código respetivo à implementação do mapa mundo.

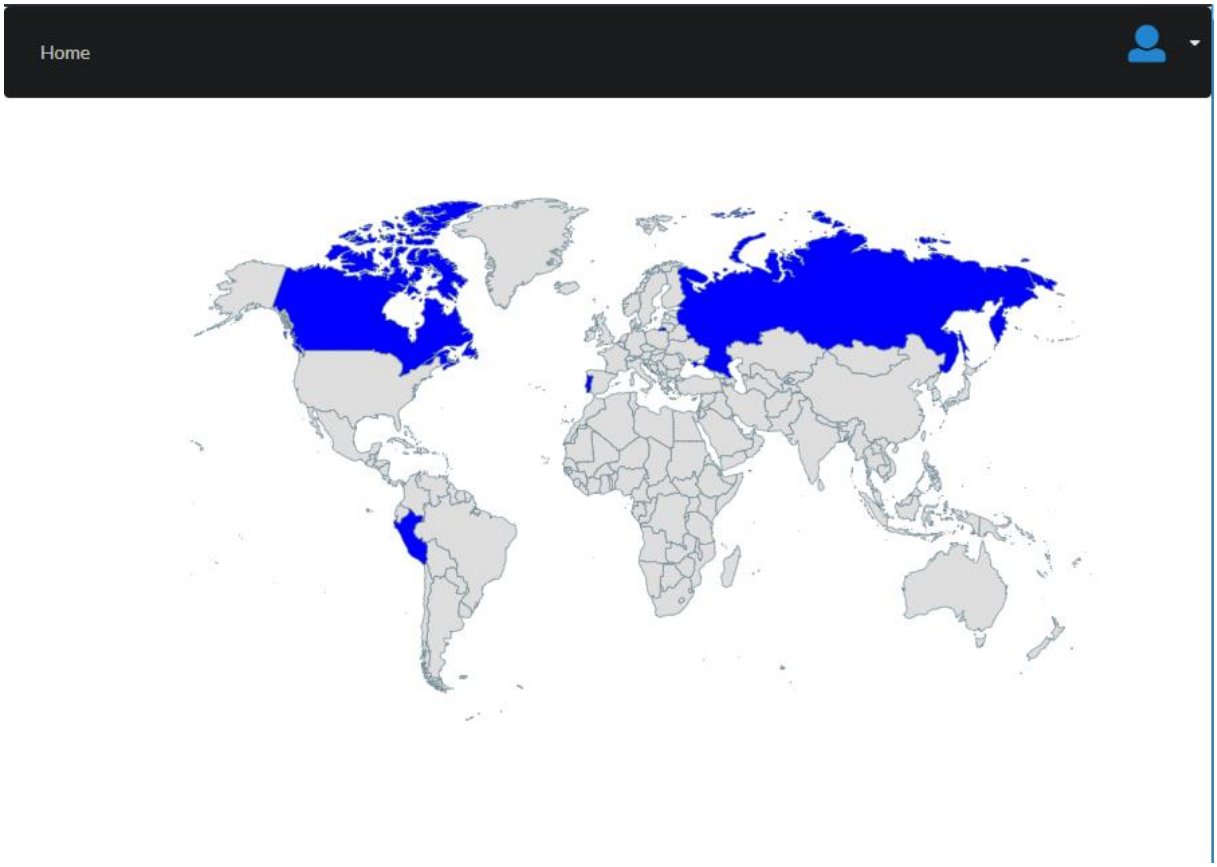

Figura 17 – Exemplo do mapa mundo com países realçados

Uma vez que todos os sete módulos foram desenvolvidos, é necessário testar se tudo está a funcionar perfeitamente. Esses testes serão analisados no próximo capítulo.

7 Verificação e validação

Neste capítulo serão analisados e validados todos os testes realizados à aplicação *web*.

Para isso, foram realizados vários testes ao longo do desenvolvimento de forma a proceder à correção do erro o mais rápido possível.

Embora se tenha testado a aplicação à medida que se ia desenvolvendo, há sempre a possibilidade de algo não ser testado. Dessa forma, foram realizados testes através da ajuda de terceiros. Estes testes foram realizados colocando uma pessoa em contacto com a plataforma para que esta a utilizasse como se fosse um potencial utilizador da aplicação. Assim, garantiu-se que um maior número de testes foi realizado.

Sempre que se encontrava um erro, este era apontado e classificado perante o seu grau de impacto na aplicação *web*, para que, posteriormente, na sua resolução, os erros com grau mais elevado fossem resolvidos primeiro.

Além dos testes indicados acima, foram também realizados testes em vários navegadores, uma vez que se trata de uma aplicação para a *web*, sendo essencial testar nos diversos navegadores, para testar a compatibilidade.

7.1 Validação do login

A Figura 18 e Figura 19 representam as validações de campos no formulário de *login*.

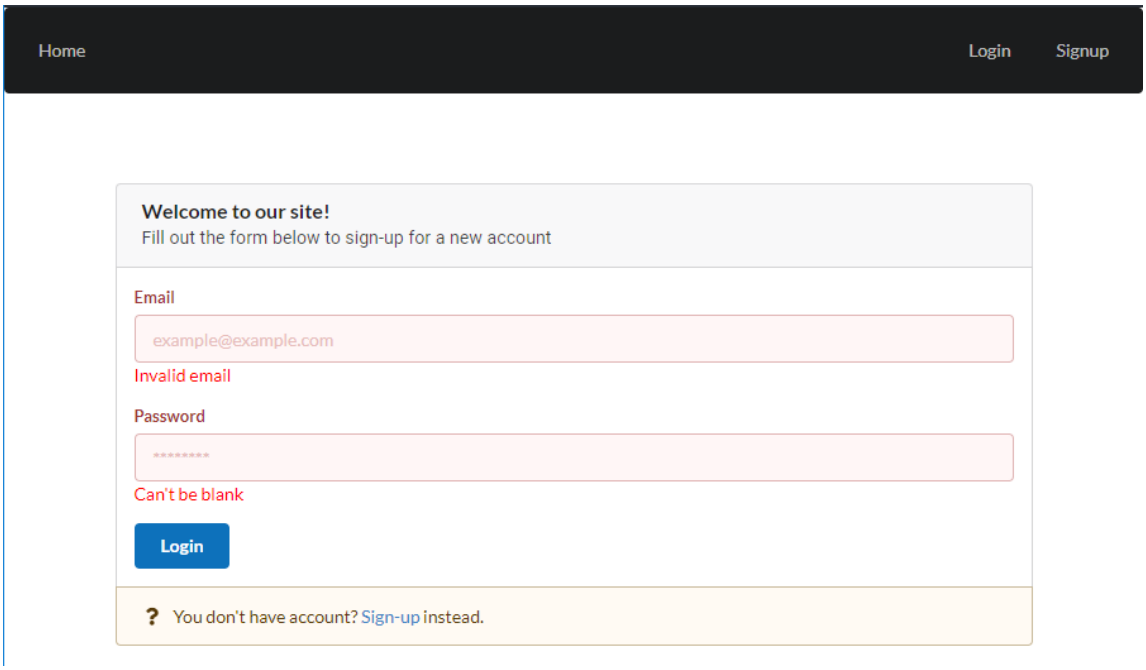

Figura 18 – Validação de erros pela aplicação web no formulário de login

Como se pode observar na Figura 18, a aplicação *web* validou os campos e indicou que os mesmos se encontram vazios ou inválidos. Já a Figura 19 apresenta os erros vindos da API, informando o utilizador que os dados introduzidos não coincidem.

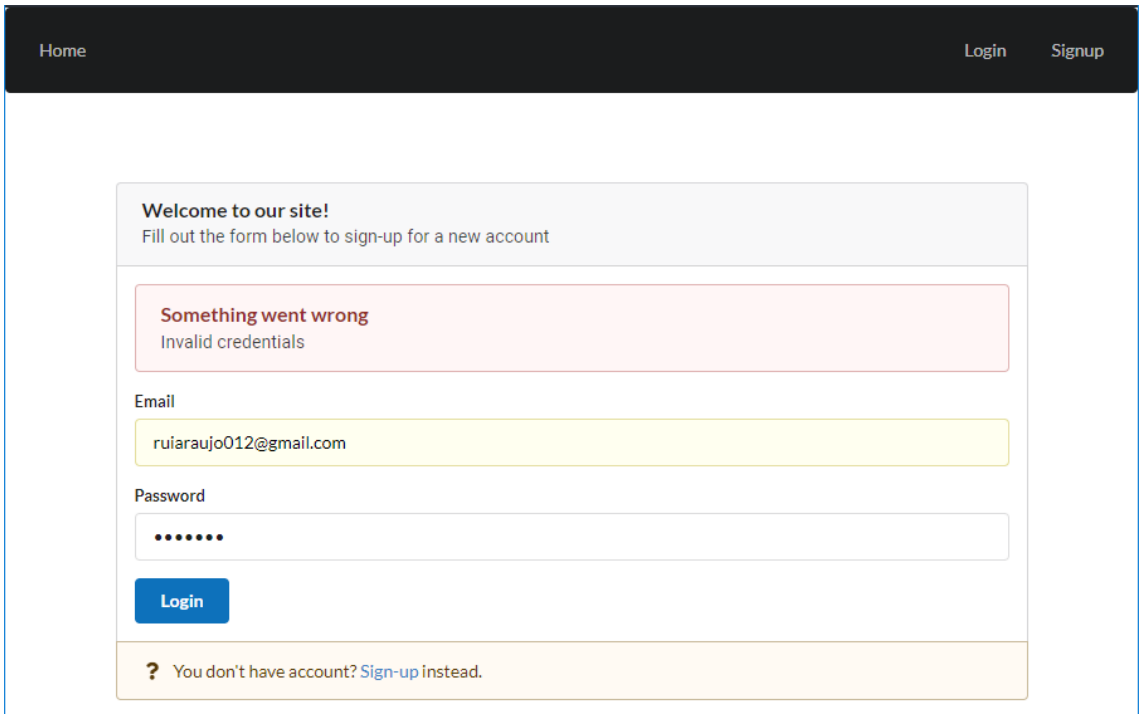

Figura 19 – Validação de erros pela API no formulário de login

7.2 Validação do signup

Como no formulário anterior, este outro valida também a existência de campos vazios ou inválidos, mas também se o email já está em uso. Este último é validado pela API.

As Figura 20 mostra a validação dos campos vazios ou inválidos, bem como as *passwords* que não coincidem.

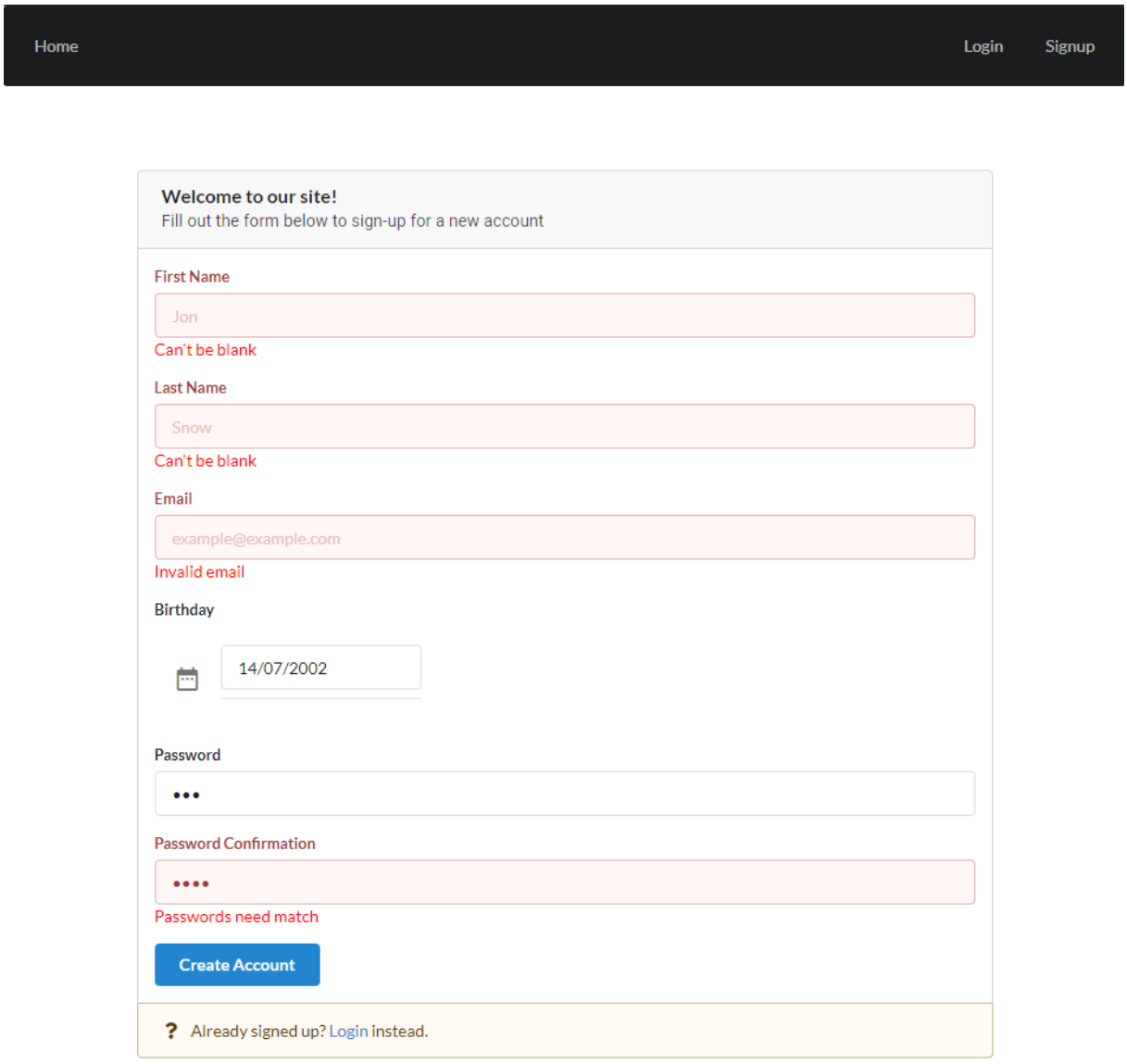

Figura 20 – Validação de erros pela aplicação web no formulário de signup

No entanto, no caso de o email inserido já estar em utilização, a API envia uma mensagem de erro para a aplicação *web*, erro este que está ilustrado na Figura 21.

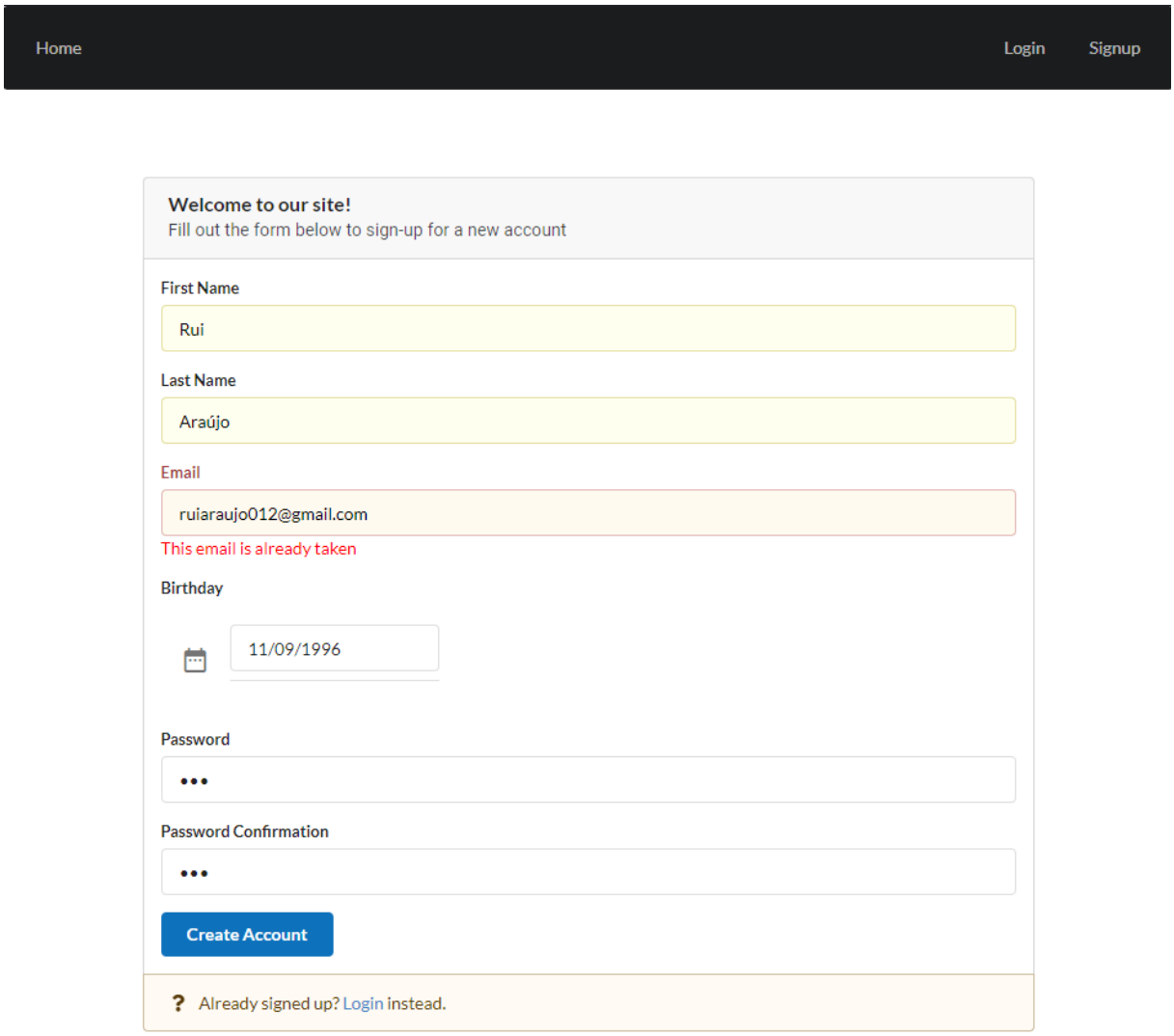

Figura 21 Validação de erros pela API no formulário de signup

Após isto, no capítulo seguinte, irá fazer-se uma conclusão acerca do trabalho realizado, as dificuldades sentidas, bem como o que se poderá fazer no futuro, de forma a melhorar o trabalho desenvolvido.

7.3 Validação da confirmação de email

No caso das validações da confirmação de email, se o utilizador introduzir o *link* de confirmação de conta errado é apresentado um erro, ilustrado na Figura 22, informando que o *token* está incorreto.

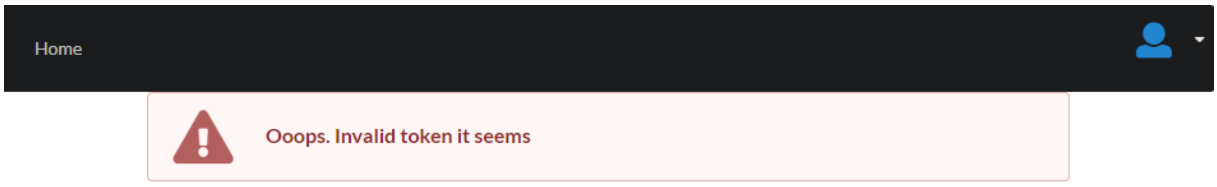

Figura 22 – Mensagem de erro de token inválido

Sempre que o utilizador introduzir o *link* correto, e no caso de este ainda não ter sito validado, aparecerá uma mensagem a confirmar a validação que se encontra na Figura 23.

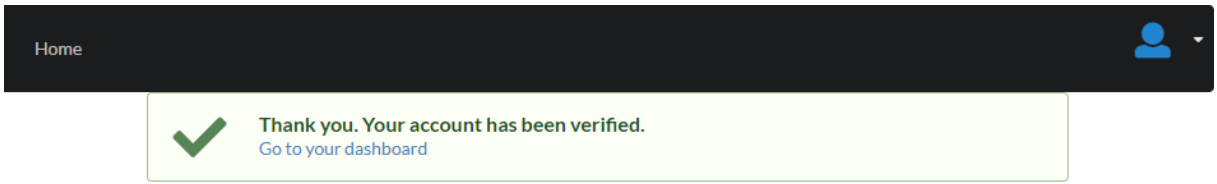

Figura 23 – Mensagem de validação de conta

8 Conclusão

O estágio curricular é uma componente bastante importante para a formação do aluno, uma vez que lhe permite obter experiência profissional antes de entrar no mercado de trabalho.

Foram alcançados todos os objetivos previstos para o presente projeto. A aplicação desenvolvida encontra-se funcional e com boa performance, sem erros detetados.

Este estágio foi bastante enriquecedor, no sentido em que permitiu melhorar a perceção do mundo empresarial e das exigências do mesmo. Contribuiu para compreender a importância do trabalho em equipa que, embora sendo esta uma equipa pequena, as necessidades aplicam-se de igual forma às das equipas com um maior número de elementos.

Para que todo o trabalho fosse desenvolvido no tempo previsto, dado a metodologia adotada, foram realizadas reuniões diárias com a equipa desenvolvedora, de modo a analisar o trabalho desenvolvido durante aquele dia e como este trabalho poderia contribuir para o objetivo final.

Além disso, este estágio foi muito vantajoso, uma vez que permitiu o domínio das tecnologias utilizadas, dado que nem todas são lecionadas na Licenciatura em Engenharia Informática. Esta foi uma das dificuldades do desenvolvimento deste projeto, sendo que, para a ultrapassar, foi necessária uma vasta pesquisa e um estudo intensivo das tecnologias desconhecidas. Apesar de uma dificuldade, acabou por se tornar bastante enriquecedor a vários níveis.

Alguns pontos positivos foram o facto de, ao longo dos três anos de Licenciatura, terem sido adquiridas competências e metodologias, as quais permitiram que a aprendizagem destas tecnologias fosse mais simples. Além de que, para a elaboração do presente relatório, foram necessários conceitos aprendidos nas UC’s de Engenharia de Software I e de Engenharia de Software II, bem como em todas as restantes UC.

Todas experiências adquiridas, tanto no estágio como durante toda a licenciatura, foram muito enriquecedoras, pois permitem preparar os alunos para as exigências do mercado de trabalho. A possibilidade que estes têm para realizar um projeto em contexto de estágio permite-lhes uma melhor preparação para a futura vida profissional.

8.1 Trabalho futuro

Ao longo das 280 horas de estágio foi desenvolvido o máximo de trabalho possível. No entanto, ainda há aspetos a melhorar e outros possíveis de implementar, de forma a tornar o projeto mais completo e enriquecedor. Alguns dos módulos que poderiam ser implementados são:

- Gestão do perfil do utilizador;
- Gestão de Utilizadores com privilégios;
- Envio de fotografias;
- Login e partilha de dados com as Redes Sociais;
- Seriação de utilizadores;
- Métricas de comparação de utilizadores.

Bibliografia

[1] TripAdvisor, “Attractions,” [Online]. Available: https://www.tripadvisor.pt/Attractions. [Acedido em Maio 2018].

[2] Google, “Trips,” Maio 2018. [Online]. Available: https://get.google.com/trips/.

[3] R. e. L. (Pplware), *Estado da Arte do Projeto Passport,* 2018.

[4] bounswe. [Online]. Available: https://github.com/bounswe/bounswe2016group6/wiki/Scrum:-A-new-perspective-on-agile-development.

[5] J. P. Faria, “UML - DIagrama de Classes - v.1.2,” outubro 2002.

[6] wiki. [Online]. Available: https://pt.wikipedia.org/wiki/JavaScript.

[7] Facebook, “ReactJS,” [Online]. Available: https://reactjs.org/.

[8] R. F. Fernandes, “itnext,” fevereiro 2016. [Online]. Available: https://itnext.io/integrating-semantic-ui-modal-with-redux-4df36abb755c.

Anexos

A.1. Descrição de casos de uso

A.1.1. Consultar países visitados

Nome: Consultar países visitados.

Descrição: Consultar, numa tabela, os países já visitados.

Pré-condições: Ser utilizador registado no site com email confirmado.

Caminho principal:

1. O utilizador seleciona no menu a opção “Dashboard”;
2. O sistema apresenta um botão para inserir uma nova viagem e uma tabela com os países já visitados;

Caminhos alternativos:

2a. O sistema apresenta apenas um botão para inserir uma nova viagem caso o utilizador ainda não tenha viagens.

Suplementos ou adornos:

Pós-condição:

A.1.2. Confirmar email

Nome: Confirmar email.

Descrição: Confirmar o email após o utilizador se ter registado.

Pré-condições: Ser utilizador registado no site.

Caminho principal:

1. O utilizador copia o link enviado para o email e cola no navegador;
2. O sistema verifica o link e confirma o utilizador;

Caminhos alternativos:

2a. O sistema verifica o link mas não confirma o utilizador uma vez que o link se encontra incorreto;

2b. O sistema verifica o link mas não confirma o utilizador uma vez que já se encontra confirmado;

Suplementos ou adornos:

1. Garantir que o link enviado é único para cada novo utilizador.

Pós-condição:

A.1.3. Registar utilizador

Nome: Registar utilizador

Descrição: Registar um novo utilizador no site.

Pré-condições:

Caminho principal:

2. O utilizador seleciona no menu a opção “Signup”;
3. O sistema apresenta um formulário com campos para o registo de utilizador;
4. O utilizador introduz o primeiro e último nome, o email, seleciona a data de nascimento, introduz a palavra passe e a confirmação e clica no botão “Create Account”;
5. O sistema insere um novo utilizador na base de dados.

Caminhos alternativos:

3a. O utilizador clica no botão “Create Account” mas deixa campos por preencher;

4a. O sistema rejeita a inserção do novo utilizador, apontando quais os campos em falta, para que a submissão seja aceite.

4b. O sistema rejeita a inserção do novo utilizador caso o email já esteja a ser utilizado.

Suplementos ou adornos:

1. Garantir que apenas seja possível associar um email a cada conta criada, não podendo haver emails repetidos.

Pós-condição:

Enviar um email para o utilizador com o *link* de confirmação da conta.

A.2. Diagrama de sequência

A.2.1. Consultar países visitados

Na Figura 24 encontra-se representado o diagrama de sequência para o caso de uso, inserir país visitado.

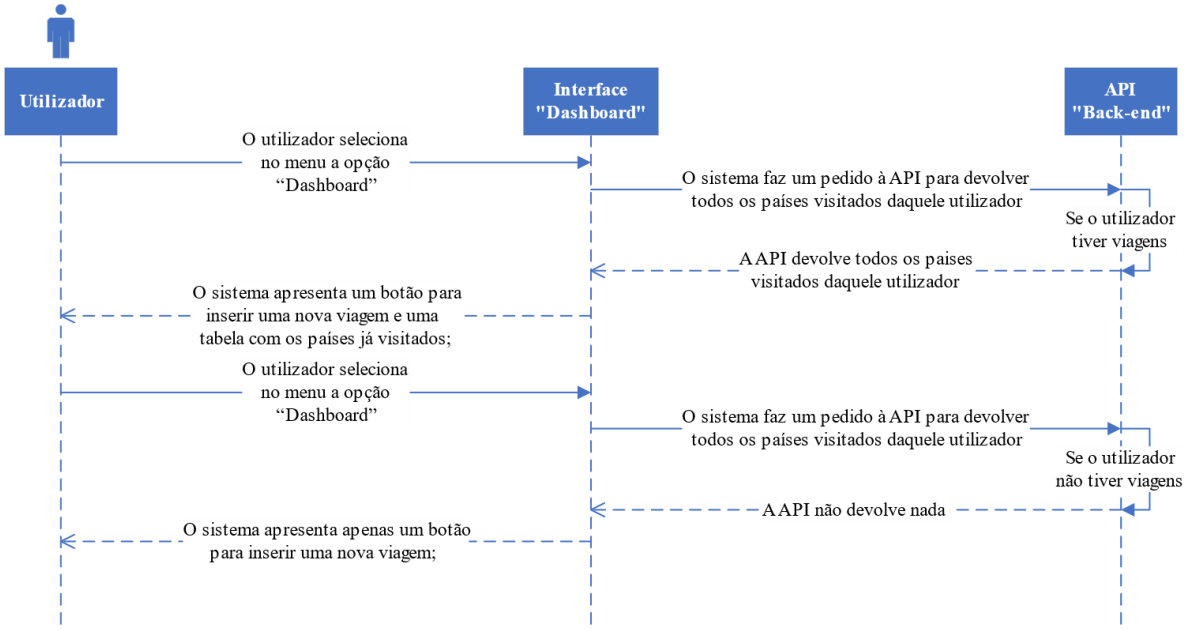

Figura 24 – Diagrama de sequência consultar países visitados

A.2.2. Confirmar email

Na Figura 25 encontra-se representado o diagrama de sequência para o caso de uso, inserir país visitado.

Figura 25 – Diagrama de sequência confirmar email

A.2.3. Registar utilizador

Na Figura 26 encontra-se representado o diagrama de sequência para o caso de uso, inserir país visitado.

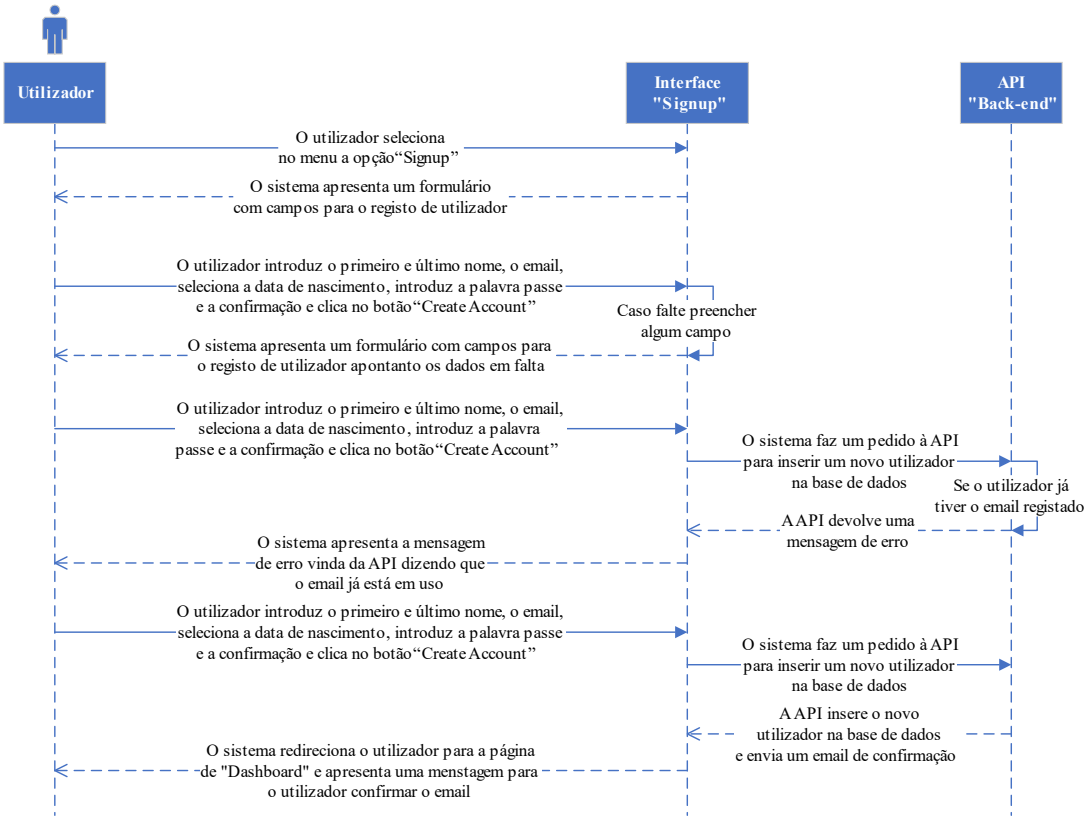

Figura 26 – Diagrama de sequência registar utilizador

A.3. Código

Dado tratar-se de um projeto em contexto de estágio, não é possível mostrar todo o código desenvolvido uma vez que este apresenta um caracter confidencial. No entanto, foi-nos permitido a partilha de algumas partes do código, as quais se encontram abaixo.

A.3.1. Código de login

O código abaixo, representa todo o formulário de *login*, apresentando também algumas validações.

```
render() {
        const { data, errors, loading } = this.state;
        return (

            <div className="ui container">

                <Message
                    attached
                    header='Welcome to our site!'
                    content='Fill out the form below to sign-
up for a new account'
                />

                <Form onSubmit={this.onSubmit} loading={loading} className='attach
ed fluid segment'>

                    {errors.global &&
                        <Message negative >
                            <Message.Header>Something went wrong</Message.Header>

                            <p>{errors.global}</p>
                        </Message>
                    }

                    <Form.Field error={!!errors.email}>
                        <label htmlFor="emial">Email</label>
                        <input
                            type="email"
                            id="email"
                            name="email"
                            placeholder="example@example.com"
                            value={data.email}
                            onChange={this.onChange}
                        />
                        {errors.email && <InlineError text={errors.email} />}
                    </Form.Field>

                    <Form.Field error={!!errors.password}>
                        <label htmlFor="password">Password</label>
                        <input
                            type="password"
                            id="password"
                            name="password"
                            placeholder="********"
                            value={data.password}
                            onChange={this.onChange}
                        />
```

```
                        {errors.password && <InlineError text={errors.password} />
    }
                    </Form.Field>

                    <Button primary loading={loading}>Login</Button>

                </Form>

                <Message attached='bottom' warning>
                    <Icon name='help' />
                    You don't have account? <Link to='/signup'>Sign-
    up</Link> instead.
                </Message>

            </div>
        );
    }
```

A.3.2. Signup

O código abaixo, representa todo o formulário de *signup*, apresentando também algumas validações.

```
render() {
        const { data, errors, loading } = this.state;

        const DEFAULT_DATE = new Date();
        const SIXTEEN_YEARS = new Date(DEFAULT_DATE);
        SIXTEEN_YEARS.setYear(DEFAULT_DATE.getFullYear() - 16);

        return (
            <div className="ui container">

                <div><br /><br /><br /></div>

                <Message
                    attached
                    header='Welcome to our site!'
                    content='Fill out the form below to sign-
    up for a new account'
                />

                <Form onSubmit={this.onSubmit} loading={loading} className='attach
    ed fluid segment'>

                    {errors.global &&
                        <Message negative >
                            <Message.Header>Something went wrong</Message.Header>

                            <p>{errors.global}</p>
                        </Message>
                    }

                    <Form.Field error={!!errors.firstName}>
                        <label htmlFor="firstName">First Name</label>
                        <input
                            type="text"
                            id="firstName"
                            name="firstName"
                            placeholder="Jon"
                            value={data.firstName}
```

```
36.                                 onChange={this.onChange}
37.                             />
38.                             {errors.firstName && <InlineError text={errors.firstName}
    />}
39.                         </Form.Field>
40.
41.                         <Form.Field error={!!errors.lastName}>
42.                             <label htmlFor="firstName">Last Name</label>
43.                             <input
44.                                 type="text"
45.                                 id="lastName"
46.                                 name="lastName"
47.                                 placeholder="Snow"
48.                                 value={data.lastName}
49.                                 onChange={this.onChange}
50.                             />
51.                             {errors.lastName && <InlineError text={errors.lastName} />
    }
52.                         </Form.Field>
53.
54.                         <Form.Field error={!!errors.email}>
55.                             <label htmlFor="email">Email</label>
56.                             <input
57.                                 type="email"
58.                                 id="email"
59.                                 name="email"
60.                                 placeholder="example@example.com"
61.                                 value={data.email}
62.                                 onChange={this.onChange}
63.                             />
64.                             {errors.email && <InlineError text={errors.email} />}
65.                         </Form.Field>
66.
67.                         <Form.Field error={!!errors.birthdayDate}>
68.                             <label htmlFor="birthdayDate">Birthday</label>
69.                             <div className='md-grid'>
70.                                 <DatePicker
71.                                     className="md-cell"
72.                                     id="birthdayDate"
73.                                     name="birthdayDate"
74.                                     defaultValue={SIXTEEN_YEARS}
75.                                     maxDate={SIXTEEN_YEARS}
76.                                     onChange={this.onChangeDate}
77.                                 />
78.                             </div>
79.                         </Form.Field>
80.
81.                         <Form.Field error={!!errors.password}>
82.                             <label htmlFor="password">Password</label>
83.                             <input
84.                                 type="password"
85.                                 id="password"
86.                                 name="password"
87.                                 placeholder="Make it secure"
88.                                 value={data.password}
89.                                 onChange={this.onChange}
90.                             />
91.                             {errors.password && <InlineError text={errors.password} />
    }
92.                         </Form.Field>
93.
94.                         <Form.Field error={!!errors.passwordConfirmation}>
95.                             <label htmlFor="password">Password Confirmation</label>
96.                             <input
97.                                 type="password"
98.                                 id="passwordConfirmation"
```

```
                                name="passwordConfirmation"
                                    placeholder="Make it secure"
                                    value={data.passwordConfirmation}
                                    onChange={this.onChange}
                                />
                                {errors.passwordConfirmation && <InlineError text={
    errors.passwordConfirmation} />}
                            </Form.Field>

                            <Button primary loading={loading}>Create Account</Butto
    n>

                        </Form>

                        <Message attached='bottom' warning>
                            <Icon name='help' />
                            Already signed up? <Link to='/login'>Login</Link> inste
    ad.
                        </Message>

                    </div>
                );
            }
```

A.3.3. Implementação de tokens

O código abaixo, apresenta a implementação do *token* no cabeçalho dos pedidos à API.

```
import axios from 'axios';

export default (token = null) => {
    if (token) {
        axios.defaults.headers.common.authorization = `Bearer ${token}`;
    } else {
        delete axios.defaults.headers.common.authorization;
    }
};
```

A.3.4. Criação de rotas autorizadas

O código abaixo, apresenta as rotas autorizadas da parte da aplicação *web*.

```
const App = ({ location }) => {
    return (
        <div>
            <MainMenu />
            <Route location={location} path='/' exact component={HomePage} />
            <UserRoute location={location} path='/confirmation/:token' exact compo
    nent={ConfirmationPage} />
            <GuestRoute location={location} path='/login' exact component={LoginPa
    ge} />
            <GuestRoute location={location} path='/signup' exact component={Signup
    Page} />
            <UserRoute location={location} path='/dashboard' exact component={Dash
    boardPage} />
            <UserRoute location={location} path='/travels/add' exact component={Ad
    dTravelPage} />
        </div>
```

```
    )
};
```

A.3.5. Confirmação de email

O código abaixo, representa as mensagem de confirmação de email.

```
render() {
        const { loading, success } = this.state;
        return (
            <div className="ui container">
                {loading && (
                    <Message icon>
                        <Icon name='circle notched' loading />
                        <Message.Header>Validation your email</Message.Header>
                    </Message>
                )}

                {!loading && success && (
                    <Message success icon>
                        <Icon name='checkmark' />
                        <Message.Content>
                            <Message.Header>Thank you. Your account has been verif
ied.</Message.Header>
                            <Link to='/dashboard'>Go to your dashboard</Link>
                        </Message.Content>
                    </Message>
                )}

                {!loading && !success && (
                    <Message negative icon>
                        <Icon name='warning sign' />
                        <Message.Content>
                            <Message.Header>Ooops. Invalid token it seems </Messag
e.Header>
                        </Message.Content>
                    </Message>
                )}
            </div>
        );
    }
```

A.3.6. Inserção de países

O código abaixo, representa todo o formulário de inserção de países, apresentando também algumas validações.

```
render() {
    const { data, errors, loading } = this.state;

    return (
      <div className='ui container'>

        <div><br /><br /><br /></div>

        <Message
          attached
```

```
11.           header='Add your travel'
12.           content='Fill out the form below to add your travel'
13.         />
14.
15.         <Form onSubmit={this.onSubmit} loading={loading} className='attached fluid
    segment'>
16.
17.           {errors.global &&
18.             <Message negative >
19.               <Message.Header>Something went wrong</Message.Header>
20.               <p>{errors.global}</p>
21.             </Message>
22.           }
23.
24.           <Form.Field error={!!errors.checkIn}>
25.             <label htmlFor="checkIn">Check-In</label>
26.             <div className='md-grid'>
27.               <DatePicker
28.                 className="md-cell"
29.                 id="checkIn"
30.                 name="checkIn"
31.                 defaultValue={data.checkIn}
32.                 onChange={this.onChangeCheckIn}
33.               />
34.             </div>
35.           </Form.Field>
36.
37.           <Form.Field error={!!errors.checkOut}>
38.             <label htmlFor="checkOut">Check-Out</label>
39.             <div className='md-grid'>
40.               <DatePicker
41.                 className="md-cell"
42.                 id="checkOut"
43.                 name="checkOut"
44.                 defaultValue={data.checkOut}
45.                 onChange={this.onChangeCheckOut}
46.               />
47.             </div>
48.           </Form.Field>
49.
50.           <Form.Field error={!!errors.country}>
51.             <label htmlFor="country">Select Country</label>
52.             <Dropdown
53.               id='country'
54.               name='country'
55.               placeholder='Select Country'
56.               fluid
57.               search
58.               selection
59.               defaultValue={data.countryCode}
60.               options={countries}
61.               onChange={this.onChangeDropdown}
62.             />
63.             {errors.country && <InlineError text={errors.country} />}
64.           </Form.Field>
65.
66.           <Form.Field error={!!errors.feedback}>
67.             <label htmlFor="feedback">Feedback</label>
68.             <TextArea
69.               id='feedback'
70.               name='feedback'
71.               placeholder='Write a feedback'
72.               defaultValue={data.feedback}
73.               onChange={this.onChange}
74.             />
75.             {errors.feedback && <InlineError text={errors.feedback} />}
```

```
          </Form.Field>

          <Button primary loading={loading}>Add Travel</Button>

        </Form>

      </div >
    );
  }
```

A.3.7. Implementação de um mapa mundo

O código abaixo, apresenta a forma como o mapa foi implementado.

```
render() {

        console.log(this.state.data);

        const { paintMap } = this;

        const { loading } = this.state;

        const MapRender = () => {
            return (
                <ComposableMap
                    projectionConfig={{
                        scale: 205,
                        rotation: [-11, 0, 0],
                    }}
                    width={980}
                    height={551}
                    style={{
                        width: "100%",
                        height: "auto",
                    }}
                >
                    <ZoomableGroup center={[0, 20]} disablePanning>
                        <Geographies geography={worldMap}>
                            {(geographies, projection) => geographies.map((geograp
hy, i) => geography.id !== "ATA" && (
                                <Geography
                                    key={i}
                                    geography={geography}
                                    projection={projection}
                                    style={{
                                        default: {
                                            fill: paintMap(geography.properties.IS
O_A2),
                                            stroke: "#607D8B",
                                            strokeWidth: 0.75,
                                            outline: "none",
                                        },
                                        hover: {
                                            fill: "#808080",
                                            stroke: "#607D8B",
                                            strokeWidth: 0.75,
                                            outline: "none",
                                        },
                                        pressed: {
                                            fill: "#808080",
                                            stroke: "#607D8B",
                                            strokeWidth: 0.75,
```

```
                                                outline: "none",
                                            },
                                        }}
                                    />
                                ))}
                            </Geographies>
                        </ZoomableGroup>
                    </ComposableMap>
                )
            }

            return (
                <div>
                    {!loading && <MapRender />}
                </div>
            );
        }
```