UNIVERSIDADE
LUSÓFONA

Buddy Abroad

Trabalho Final de curso

Diogo Azevedo:21702801
Mário Oliveira: 21703872
Rui Ribeiro

Trabalho Final de Curso | LIG | Data

Direitos de cópia

Indice

Indice de Figuras

Resumo

Buddy Abroad é uma plataforma mobile constituída por uma aplicação que pretende renovar a forma de olhar para o turismo convencional que até hoje temos visto, proporcionando ao utilizador uma nova experiência mais interativa e descontraída.

A ideia é procurar oferecer experiências mais genuínas e atrativas aos turistas sendo acompanhados por habitantes nativos, em percursos personalizados e horários flexíveis a preços atrativos.

O grande desafio será conquistar turistas que procurem uma estadia mais enriquecedora, promovendo interações socioculturais ao mesmo tempo que aprofundam o conhecimento sobre a história local

Abstract

Buddy Abroad is a mobile platform constituted by one app and intends to innovate the way we look at the conventional tourism practices alongside with providing a more laid- back experience to the user.

Offering a genuine and attractive experience to its user is the main premise and it will try do it by uniting tourists with local residents in guided activities with flexible schedules and attractive prices.

1. Identificação de problema

O mercado onde é pretendido que esta plataforma atue caracteriza-se pela ausência de concorrência direta e pela falta de soluções no mercado que proporcionem aos turistas uma experiência mais autêntica, personalizável, com horários flexíveis e preços atrativos.

Tal como demonstra o questionário com a amostra de 147 turistas, acerca das suas motivações e preferências turísticas efetuado por Júlia Melo e Helga Antunes (Mestradas em Turismo, Gestão de Empresas Turísticas):

- 74,8% dos turistas não participam em excursões pré-definidas, tendo como principais motivos a dependência de rotas e horários pré-definidos e generalista no que toca a atrações
- 50,3 % referem ter muita vontade, de interagir com a cultura e com os nativos de forma mais direta.

A perspetiva do turismo tem-se alterado bastante com o decorrer dos anos, a maioria dos viajantes hoje em dia pretende mais do que conhecer as paisagens, monumentos e peças históricas envolver-se com a cultura e os seus respetivos nativos. Em paralelo às alterações à forma de viajar, consequentemente evoluiu também a necessidades de aplicações que simplifiquem toda a burocracia envolvente, logo mais do que um desejo, esta a aplicação está-se a tornar uma necessidade.

Pretende-se então a criação de uma plataforma que visa suprir essa procura turística por experiências autênticas e interação com os locais, de modo a alocar "Buddys" mais compatíveis com os turistas, tendo por base as especificações linguísticas.

Através de um estudo realizado em 2017 podemos perceber o impacto e a necessidade de aplicações mobile no panorama atual do turismo.

Quantas vezes você já utilizou um aplicativo de turismo durante sua viagem?	Frequência	Porcentagem %
Nunca	0	0%
De 1 a 5 vezes	29	48%
De 6 a 10 vezes	11	18%
De 11 a 15 vezes	4	7%
De 16 a 20 vezes	1	2%
De 21 a 25 vezes	0	0%
Mais de 25 vezes	15	25%

Figura 1:Utilização de Aplicações no Turismo

Quanto à quantidade de vezes que os entrevistados utilizam aplicações de turismo nas suas viagens, verifica-se que 48% utilizam de 1 a 5 vezes, 18% de 6 a 10 vezes, 7% de

11 a 15 vezes, 2% de 16 a 20 vezes, e 25% mais de 25 vezes. Isso mostra que no geral, a utilização deste tipo de aplicações durante as viagens ainda é razoável.

Vamos então verificar que tipo de aplicações os utilizadores mais procuram:

Quais aplicativos de turismo você utilizou durante sua viagem?	Frequência	Porcentagem %
Google Maps	53	88%
Facebook	45	75%
Google Translate	28	47%
Foursquare	26	43%
Skype	24	40%
Booking.com	18	30%
Tripadvisor	17	28%
Conversor de Moedas	13	22%
Tradutor de Viagem	4	7%
Infraero Vôo Online	2	3%
Lista de Viagem	1	2%
aMetro	1	2%
Easy Currency	1	2%

Figura 2: Aplicações mais procuradas

Podem ser observadas as aplicações que os entrevistados mais utilizam nas suas viagens. A maioria recorre ao Google Maps [88%], ao Facebook [75%] e ao Google tradutor [47%]. Os demais aplicativos citados foram: o Foursquare [43%]; o Skype [40%]; o Booking.com [30%]; o TripAdvisor [28%], o Conversor de Moedas [22%], o Tradutor de Viagem [7%], Infraero Vôo Online [3%], o aMetro [2%], o Easy Currency [2%]. Há 7% que utilizam outras aplicações, dentre eles o Decolar.com, Local, Instagram, GPS, ficando estes com 0%.

2. Levantamento de requisitos

Neste ponto vamos falar dos requisitos que foram implementados totalmente, bem como aqueles que sofreram alterações com base nas sugestões feitas pelos utilizadores que testaram a aplicação. Por motivos de simplificação do relatório, vamos apenas apresentar os *userstories* e vamos disponibilizar o acesso ao *easybacklog* nos **8.2Anexos** para poder ver os critérios de aceitação de todos os requisitos.

2.1 Requisitos Implementados na totalidade

Backlog: Buddy Abroad Tema: Login	LOG1
Como **Utilizador** Eu quero **fazer login** Para que **aceder á aplicação**	

Backlog: Buddy Abroad Tema: Registo	RE11
Como **Utilizador** Eu quero **registar-me** Para que **possa criar uma conta**	

Backlog: Buddy Abroad Tema: Recuperar Passe	REP1
Como **Utilizador** Eu quero **recuparar a minha palavra passe** Para que **possa voltar a entrar na app**	

Backlog: Buddy Abroad Tema: Home	HOM1
Como **Utilizador** Eu quero **ver visitas populares** Para que **possa ter acesso rapido as visitas populares**	

Backlog: Buddy Abroad Tema: Home	HOM2
Como **Utilizador** Eu quero **ver visitas proximas** Para que **possa ter acesso rapido as visitas proximas de mim**	

Backlog: Buddy Abroad Tema: Search	SEA1
Como **Utilizador** Eu quero **poder pesquisar visitas** Para que **possa fazer escolhas com base em criterios**	

2.1 Requisitos Implementados na totalidade (continuação)

Backlog: Buddy Abroad	RVE1
Tema: Reenviar verification Email	
Como **Utilizador** Eu quero **receber o email de verificação novamente** Para que **possa confirmar o meu registo**	

2.2 Requisitos modificados

Backlog: Buddy Abroad	HOM2
Tema: Home	
Como **Utilizador** Eu quero **ver visitas proximas** Para que **possa ter acesso rapido as visitas proximas de mim**	

Backlog: Buddy Abroad	SEA1
Tema: Search	
Como **Utilizador** Eu quero **poder pesquisar visitas** Para que **possa fazer escolhas com base em criterios**	

Backlog: Buddy Abroad	APP1
Tema: Alterar palavra passe	
Como **Utilizador** Eu quero **receber um email** Para que **poder alterar a minha palavra passe**	

2.3 Requisitos não implementados

Backlog: Buddy Abroad Tema: Profile	PRO3
Como **Utilizador** Eu quero **editar buddy info** Para que **poder apresenta-la a outros clientes**	

Backlog: Buddy Abroad Tema: Profile	PRO4
Como **Utilizador** Eu quero **ver metodos de pagamento** Para que **decidir se quero adicionar ou remover algum deles**	

Backlog: Buddy Abroad Tema: Profile	PRO5
Como **Utilizador** Eu quero **ver contas de recebimento** Para que **decidir se quero adicionar ou remover alguma delas**	

Backlog: Buddy Abroad Tema: Definitions	DEF1
Como **Utilizador** Eu quero **aceder as definições** Para que **possa editar as definições da aplicação**	

Backlog: Buddy Abroad Tema: Definitions	DEF2
Como **Utilizador** Eu quero **ter notificações** Para que **possa saber ser avisado de atividades na aplicação**	

Backlog: Buddy Abroad Tema: Definitions	DEF3
Como **Utilizador** Eu quero **ligar/desligar notificações** Para que **possa optar por ter, ou não ter notificações**	

Backlog: Buddy Abroad Tema: Definitions	DEF4
Como **Utilizador** Eu quero **aceder as definições** Para que **possa mudar a palavrapasse**	

Backlog: Buddy Abroad Tema: My Bookings	MYB1
Como **Visitante** Eu quero **Aceder aos meus bookings** Para que **ver os bookings que tenho feitos**	

Backlog: Buddy Abroad **MYB2**
Tema: My Bookings

Como **Visitante**
Eu quero **selecionar um booking**
Para que **possa ver os seus detalhes e editala**

Backlog: Buddy Abroad **MYB3**
Tema: My Bookings

Como **Visitante**
Eu quero **selecionar um booking**
Para que **possa editar o seu agendamento**

Backlog: Buddy Abroad **MYB4**
Tema: My Bookings

Como **Visitante**
Eu quero **selecionar um booking**
Para que **possa mudar o tamaho do seu grupo**

Backlog: Buddy Abroad **MYV1**
Tema: My visits

Como **Buddy**
Eu quero **aceder as minhas visitas**
Para que **possa selecionar visitas criadas e ver a visita ativa**

Backlog: Buddy Abroad **YOT1**
Tema: Your tourists

Como **Buddy**
Eu quero **selecionar os meus turistas**
Para que **possa ver os detalhes de um determinado agendamento**

Backlog: Buddy Abroad **BOP1**
Tema: Booking Page

Como **Buddy**
Eu quero **selecionar o meu grupo**
Para que **possa limitar o grupo**

Backlog: Buddy Abroad **PRO1**
Tema: Profile

Como **Utilizador**
Eu quero **aceder ao perfil**
Para que **possa ver e editar as minhas informações**

Backlog: Buddy Abroad **PRO2**
Tema: Profile

Como **Utilizador**
Eu quero **editar personal info**
Para que **possa manter a minha informação atualizada**

Backlog: Buddy Abroad **MYV2**
Tema: My visits

Como **Buddy**
Eu quero **aceder as minhas visitas**
Para que **possa selecionar a visita que quero editar**

Backlog: Buddy Abroad **MYV3**
Tema: My visits

Como **Buddy**
Eu quero **ver os agendamentos feitos a minha visita**
Para que **possa ver os seus detalhes**

Backlog: Buddy Abroad **MYV4**
Tema: My visits

Como **Buddy**
Eu quero **aceder as minhas visitas**
Para que **possa criar uma visita**

Backlog: Buddy Abroad **TOP1**
Tema: Tour Page

Como **Utilizador**
Eu quero **aceder a tour page**
Para que **possa ver os detalhes da tour**

Backlog: Buddy Abroad **TOP2**
Tema: Tour Page

Como **Utilizador**
Eu quero **selecionar uma visita**
Para que **possa ver as suas datas disponiveis**

Backlog: Buddy Abroad **TOP3**
Tema: Tour Page

Como **Utilizador**
Eu quero **ver datas possiveis**
Para que **possa agendar uma visita**

Backlog: Buddy Abroad **TOP4**
Tema: Tour Page

Como **Utilizador**
Eu quero **editar o tamanho do grupo**
Para que **se possa definir preços**

Backlog: Buddy Abroad **PAM1**
Tema: Payment method

Como **Utilizador**
Eu quero **visualizar os detalhes da compra**
Para que **possa finalizar e verificar o check-out**

Backlog: Buddy Abroad **CRT2**
Tema: Create Tour

Como **Buddy**
Eu quero **adicionar linguagem**
Para que **possa definir o meu publico**

Backlog: Buddy Abroad **CRT3**
Tema: Create Tour

Como **Buddy**
Eu quero **escolher calendario**
Para que **possa definir datas**

Backlog: Buddy Abroad **CRT4**
Tema: Create Tour

Como **Buddy**
Eu quero **selecionar calendario**
Para que **alterar a data**

Backlog: Buddy Abroad **CRT5**
Tema: Create Tour

Como **Buddy**
Eu quero **selecionar ecra photos**
Para que **possa adicionar fotos do local**

Backlog: Buddy Abroad **PAM2**
Tema: Payment method

Como **Utilizador**
Eu quero **escolher metodo de pagamento**
Para que **selecionar um cartão que já tenha adicionado anteriormente ou adicionar um cartão novo**

Backlog: Buddy Abroad **PAM3**
Tema: Payment method

Como **Utilizador**
Eu quero **adicionar cartao**
Para que **que possa ser utilizado como forma de pagamento futuramente.**

Backlog: Buddy Abroad **PAM4**
Tema: Payment method

Como **Utilizador**
Eu quero **adicionar conta bancaria**
Para que **possa fazer um pagamento**

Backlog: Buddy Abroad **CRT1**
Tema: Create Tour

Como **Buddy**
Eu quero **escolher localizaçao**
Para que **possa definir a localizaçao da visita**

3.Viabilidade e pertinência

Antes do que tudo, será necessário olhar para o panorama do mercado atual e verificar a importância do mobile, numa perspetiva de turismo.

Segundo um estudo realizado pela Travelport no ano de 2018 acerca da evolução do m-tourism, onde foram questionados 16.000 viajantes de 25 países diferentes, podemos analisar a importância e crescimento do mesmo.

Temos então que a maior procura reside nas aplicações que permitem procurar e agendar voos (68%), ver itinerários (67%), alertas sobre voos ao longo da sua jornada (64%).

Em média, um utilizador usa 10-12 aplicações durante o processo de pesquisa, sendo que, como seria de esperar, aquando questionados se prefeririam uma aplicação única que conjugasse todas, a maior parte respondeu que sim.

Referência bibliográficas: 2018 Global DigitalTraveler Research, Travelport

Uma vez que o buddy que vai fazer de guia é nativo, terá maior conhecimentos sobre a cultura nacional, o que será útil de forma a fornecer ao utilizador dicas e sugestões que lhe serão uteis na sua viagem, como por exemplo:

Planeia a experiência

Dicas de como aproveitar melhor o tempo

Sugestões de locais

Forma de fazer amigos

A pertinência deste projeto enquadra-se na medida em que olhando para a evolução constante do mercado do turismo em paralelismo com a necessidade de redução de todos os passos “chatos” e toda a burocracia envolvente, leva a que uma aplicação deste género seja necessária de forma a colmatar essa necessidade de um nicho de mercado com uma prospeção evolutiva alta.

A evolução dos tempos provou que o que há 25 anos seria muito difícil, arriscando mesmo a dizer que utópico, se está a tornar cada vez mais real no turismo atual, as pessoas cada vez mais gostam de viajar sozinhas e já não olham para isso como tabu, cada vez mais pretendem envolver-se com a cultura muito mais do que tirar um selfie com a torre eiffel no panorama para apenas provar que realmente esteve no local, contudo a plataforma não pretende excluir de todo este tipo de pessoas, contudo a probabilidade de uma pessoa que apresente este biótipo mental pretender uma interação mais humana, enriquecedora e interativa é sensivelmente baixa. Resumindo podemos então concluir que a realização de uma plataforma deste nível é bastante viável.

Como termo de comparação, nada melhor do que uma aplicação tão próxima de nós, como a Zarco, startup portuguesa cujo modelo de negócio , à semelhança da Airbnb e da Uber, é do tipo plataforma multilateral, visto que liga pessoas interessadas em conhecer a cidade, de uma forma mais personalizada, a pessoas disponíveis para o fazer, dentro das sua(s) área(s) de interesse. Um arquiteto portuense conseguirá fazer com que o turista que o escolheu tenha uma experiência única, porque lhe mostrará a cidade com um outro olhar, mais pormenorizado, sempre de encontro às paixões que ambos partilham. Quem nasce e cresce no local conhece recantos que a maioria dos guias turísticos tradicionais desconhece. (Ferreirinha, 2017)

A Zarco cobra uma comissão de 40% sobre o valor de cada visita guiada feita. Num passeio de 2 horas, com o custo de 40€ para o cliente, a empresa fica com 16€ e o guia turístico com os restantes 24€. Contudo, há um ponto interessante que deve ser referido: os pagamentos são feitos na aplicação, quando agendado o encontro. Desta forma, cabe à Zarco processá-lo e transferir a percentagem certa ao guia. Isto, para além de evitar problemas jurídicos, garante um maior conforto, tanto ao viajante como ao próprio agente turístico, que não tem a necessidade de ter dinheiro em sua posse, evitando riscos desnecessários, como por exemplo, assaltos. (Ferreirinha, 2017)

Comparando então esta aplicação com os seus concorrentes indiretos temos o seguinte:

TABELA COMPARATIVA AIRBNB, UBER E ZARCO - SEGMENTOS DE MERCADO

	AIRBNB	UBER	ZARCO
turismo	×	×	×
pessoas que gostam de viajar	×		×
pessoas que não têm carro		×	
pessoas interessadas em conhecer a cidade	×		×
pessoas que pretendem um serviço personalizado	×	×	×

Figura 3:Estudo de Mercado

O interesse em conhecer a cidade, por parte de um turista, vai mais de encontro à Airbnb e à Zarco. A primeira, completou o seu serviço de alojamento com um serviço de experiências em que o viajante pode escolher uma série de programas a fazer na cidade que visita. A ideia é semelhante ao que a segunda apresenta como serviço, com a diferença de que o "guia" é o próprio anfitrião, o que poderá ser uma desvantagem em relação à Zarco, porquanto os guias desta são certificados ou têm interesses provados nas áreas a que se candidatam. (Ferreirinha, 2017)

Quanto à proposta de valor:

TABELA COMPARATIVA AIRBNB, UBER E ZARCO - PROPOSTA DE VALOR

	AIRBNB	UBER	ZARCO
serviço lowcost	×	×	×
segurança	×	×	×
política de cancelamento	×	×	desconhecido
facilidade no processo de pagamento	×	×	×

Figura 4: Proposta de Valor

Quanto à proposta de valor, conforme é visível na figura 3, estas empresas apresentam uma solução lowcost dos serviços que prestam, os quais são seguros e cómodos. Há facilidade de pagamento em todas, uma vez que este é efetuado através da aplicação, com recurso a processadores de pagamento. A Airbnb e a Uber detêm políticas de cancelamento. Relativamente à Zarco desconhece-se tal informação. Todas prezam satisfazer as necessidades e desejos do(s) cliente(s). (Ferreirinha, 2017)

Referência: O papel das aplicações móveis no turismo – o caso Zarco, Marta Sofia

(Ferreirinha, 2017)

4. Solução Desenvolvida

4.1 Front-End

A solução desenvolvida e aplicada para o *front end* da aplicação tem como base os storyboards dos vários níveis da aplicação utilizando a ferramenta Axure. Estes storyboards têm como objetivo criar uma ideia geral do funcionamento que a aplicação terá, e de estabelecer um tema geral que a aplicação deve seguir em termos visuais.

Olhando agora para o desenvolvimento do front-end propriamente dito, tomámos a decisão de utilizar a framework Ionic que tem por base a framework Angular mantida pela Google, que por sua vez tem as suas bases em HTML, CSS e TypeScript. A escolha desta framework veio com o intuito de seguir o trabalho realizado anteriormente pelo nosso colega, e por esta ter uma boa documentação e disponibilizar a capacidade de desenvolvimento em paralelo para Android e IOS.

Olhando para a forma de fazer comunicações entre o servidor e a aplicação decidimos seguir o que o nosso colega Fábio Bexiga recomendou, e estamos a usar uma REST API na qual as comunicações são feitas com a ajuda da classe "HttpClient" disponibilizada pela Angular, sendo através desta classe que realizamos todos os pedidos Http necessários.

4.2 Back-End

Olhando agora para o back-end, seguimos as ideias da solução anterior e estamos a utilizar uma baze de dados feita em Postgresql, utilizando a ferramenta PgAdmin para criar e editar a nossa base de dados.

Para a comunicação entre base de dados e aplicação, criámos um web server em javascript no qual usamos a Node.js, que nos permite utilizar realizar comunicações entre a base de dados e o servidor através do NodePg, sendo isto uma coleção de módulos que nos permitem interagir com a nossa base de dados, nomeadamente escrever scripts SQL diretamente no código do nosso servidor.

Para a comunicação entre servidor e aplicação estamos a usar a usar Node Express que é uma web framework que nos vai permitir receber pedidos REST tratá-los e responder-lhes. Para além disto o express ajuda-nos a estruturar o trabalho através do Router e traz consigo enumeras outras ferramentas que podemos utilizar mediante as necessidades que os serviços que pretendemos disponibilizar exigem.

Para envio de emails estamos a usar a api da Mailgun, que nos permite enviar até 5000 emails por mês de forma gratuita e tem vários planos de pagamento que facilitam a escalabilidade da aplicação se necessário.

Por fim, para o armazenamento das imagens decidimos que seria melhor guardar as imagens na cloud da firebase ao invés de as guardar diretamente na base de dados. Desta forma na base de dados teremos apenas um link para a imagem guardada em cloud. Isto torna-se também uma vantagem para o transporte das imagens do servidor para o utilizador por estarmos apenas a enviar um link, em vez de ser enviada a imagem convertida para base 64.

4.3 Arquitetura

Depois de enumeradas e explicadas as interações entre as ferramentas podemos então apresentar a arquitetura utilizada na aplicação:

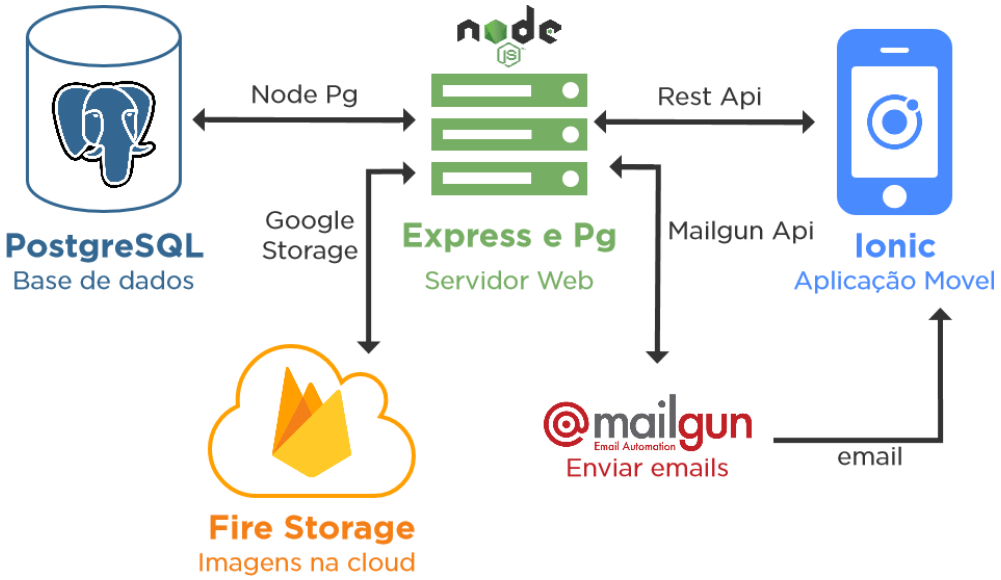


Figura 5: Arquitetura da solução desenvolvida

4.4 Níveis da aplicação

Neste ponto criámos os vários níveis da aplicação para conseguirmos ter uma *overview* da aplicação na sua totalidade.

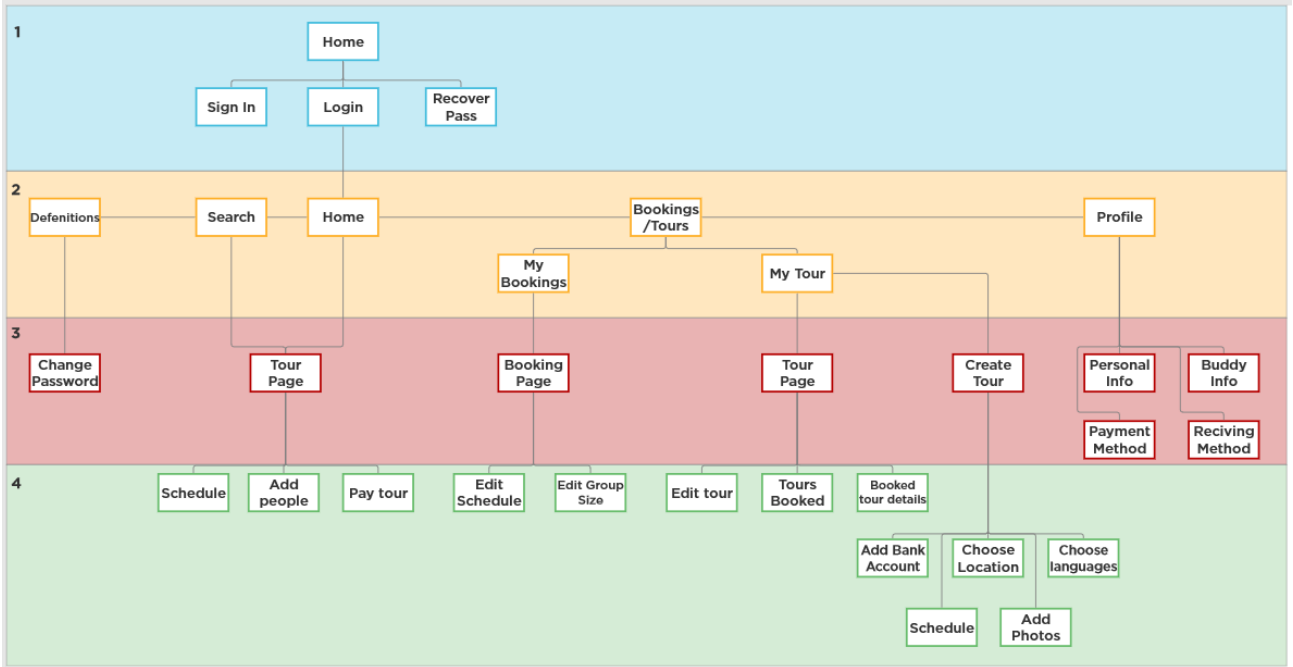


Figura 6: Niveis da Aplicação

4.6 Base de dados

Olhando então para o modelo relacional desenvolvido e implementado em PostgreSQL, temos o seguinte:

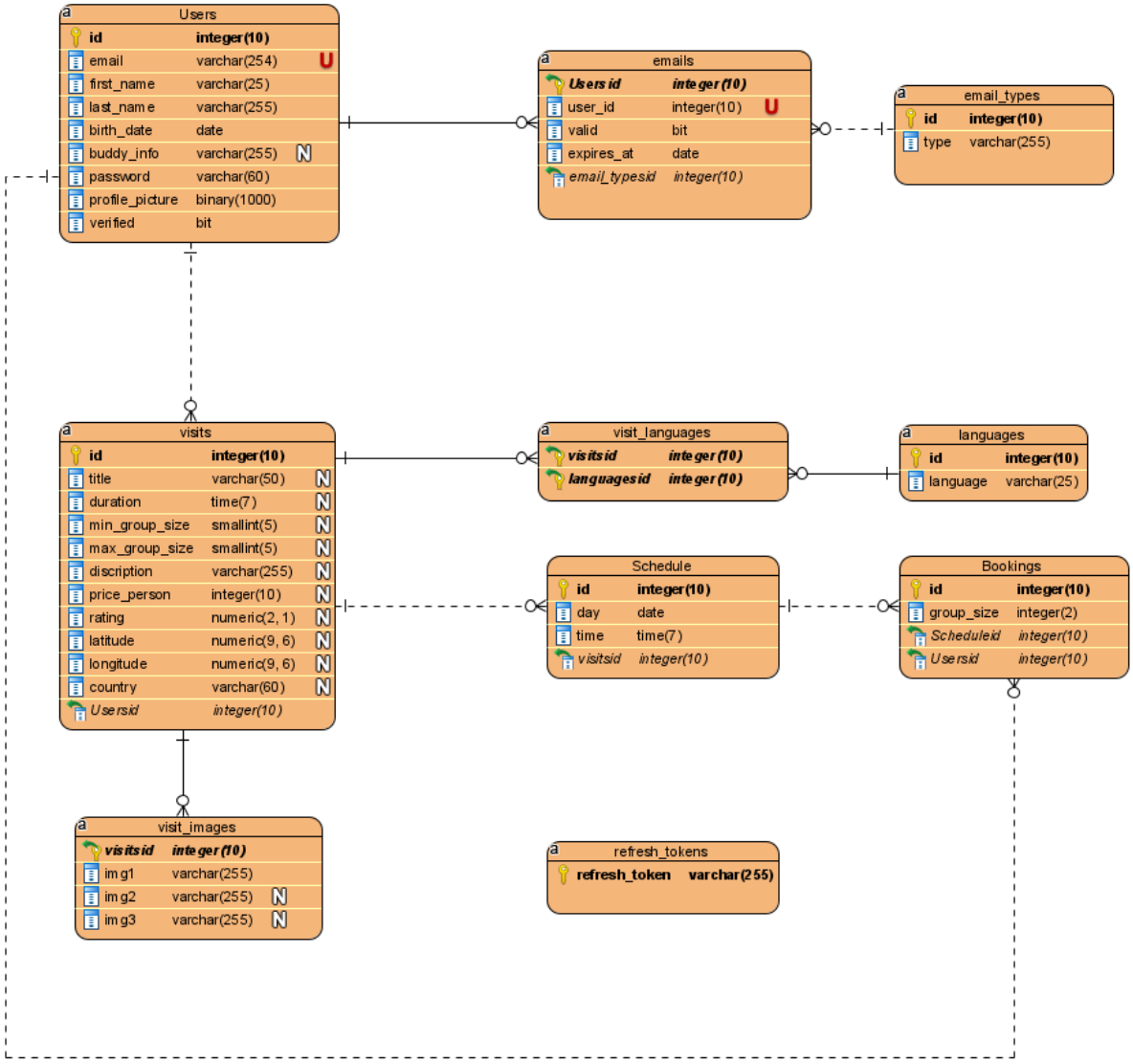


Figura 7: Modelo Relacional

Para além da implementação do modelo, implementamos uma função capaz de calcular a distância em unidades do sistema métrico (Ex: distancia em quilómetros) entre duas geolocalizações.

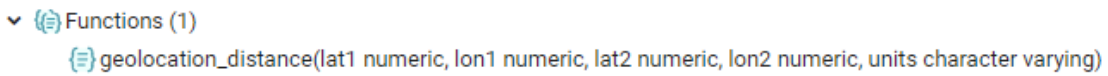


Figura 8: Função geolocation_distance

4.7 Web Server

Para explicar o servidor desenvolvido vamos identificar os serviços que este fornece, a estrutura do servidor, e por fim relacionar as partes do servidor com os seus respetivos serviços.

O servidor desenvolvido oferece 4 serviços distintos (que podem facilmente ser migrados cada um para o seu servidor dedicado), estes serviços são:

- Responder a pedidos feitos pela aplicação
- Autenticar pedidos feitos pela aplicação
- Disponibilizar a página web para alteração de palavra passe
- Enviar emails aos utilizadores

A estrutura do servidor está organizada da seguinte forma:

- *Config*: Contem configurações gerais, como o porto do servidor e chaves para *API* da Mailgun.
- Database: Contêm a *API* que faz *queries* a base de dados e um *error handler* para lidar com erros desses *queries*.
- Jwt: Contêm o validador de pedidos feitos ao servidor.
- Mailer: Contém a *API* que envia emails e valida emails.
- Routes: Contém o código para lidar com os diferentes pedidos de cada ecrã da aplicação movél.
- Static: Contém os recursos utilizados pela página web de recuperação de palavra-passe.
- Views: Contém o html da página web de recuperação de palavra-passe.
- Env: Contém chaves secretas utilizadas para criação de tokens Jwt, e o domain do servidor.
- Index.js: A raiz do servidor. Os pedidos são todos recebidos aqui e redirecionados para sua respetiva *route*

Podemos então finalizar relacionando os serviços com os módulos do servidor:

- Responder a pedidos feitos pela aplicação: Routes
- Autenticar pedidos feitos pela aplicação: Jwt
- Disponibilizar a página web para alteração de palavra passe: Static, Views
- Enviar emails aos utilizadores: Mailer

5. Benchmarking

5.1 Analise SWOT

Forças

- Ausência de concorrência direta
- Reputação inovadora
- Cash-flow simples
- Poucos ativos tangíveis
- Service on the fly

Fraquezas

- Pode ser considerado uma aplicação sazonal, tendo picos de utilização nas alturas de maior afluência.
- Não temos total controlo sobre o pessoal, mesmo verificando registo criminal.
- Alta dependência de uma boa adesão à app

Oportunidades

- Evolução do turismo numa perspetiva de interação sociocultural.
- Colmatar a necessidade de um nicho de mercado.
- Evolução dos hábitos de consumo de necessidade instantânea.

Ameaças

- Número elevado de concorrentes indiretos.
- Concorrência com forte poder negocial e promocional.
- Dificuldade em garantir uma imagem de marca forte.

5.1 Analise Comprativa

A nossa maior concorrência será a AirBnb e a Zarco sendo que a proposta que o Buddy Abroad sendo que as três se propõem a fornecer serviços bastante semelhantes, a Citymapper é também uma possível concorrente visto que esta é ainda mais indireta, contudo apenas permite calcular da melhor forma possível o trajeto entre possíveis meios de locomoção não alocando a este um serviço de orientação feita por um guia.

Apenas a título de curiosidade podemos também olhar para a Snapcity aplicação produzida por uma Startup portuguesa cuja ideia era ligar nativos a turistas através de chat, sendo que estes iriam colocar duvidas e pedir sugestões, às quais os nativos poderiam responder através do canal de comunicação acima descrito.

A nossa proposta final reúne a maioria funcionalidades da nossa concorrência, sendo que o nosso posicionamento passa por sempre que se pensa em turismo moderno se pense em Buddy Abroad,

Uma vez que a aplicação não esta concluída é impossível comparar o funcionamento da nossa proposta em relação à concorrência.

6. Método de planeamento

Abordando o método de planeamento de trabalho seguido no desenvolvimento do projeto tentamos dividi-lo em duas partes. No primeiro semestre deparamo-nos com facto de não termos bases para dar continuação ao trabalho realizado anteriormente, porque nos faltavam bases de conhecimentos web, tanto teóricos como práticos, e por este motivo dedicamos parte do desenvolvimento para a aprendizagem destes elementos.

Para o calendário do primeiro relatório tínhamos:

- 02/12/2019 - Definição de Níveis e storyboards das aplicações
- 02/12/2019 a 13/12/2019 - 1ª Iteração de protótipos das aplicações
- 13/12/2019 - User testing dos protótipos
- 15/12/2019 - Analise Heurísticas
- 16/12/2019 a 29/12/2019 – 2ª Iteração de protótipos
- 30/12/2019 a 12/1/2020 – Verificação junto de users

Para o primeiro calendário conseguimos cumprir todos os pontos dentro das datas estabelecidas e terminamos com os storyboards e requisitos para cada um deles estabelecidos, mas em termos de aprendizagem das linguagens web (Javascript, Html, Css) ficamos a quem do que tínhamos planeado e por este motivo entramos no segundo semestre sem saber utilizar bem as linguagens para dar continuação ao trabalho. Por este motivo decidimos recomeçar o trabalho de raiz, para podermos compreender o que estávamos a desenvolver ao invés de continuar a interpretar sem sucesso o trabalho realizado anteriormente.

Para o calendário do segundo relatório tínhamos:

- 24/01/2020 – Reunião com Fábio Bexiga (autor do TFC anterior)
- 28/01/2020 a 08/02/2020 – Compreensão dos conceitos apresentados na reunião
- 09/02/2020 a 15/02/2020 – Definir modelo da base de dados e implementá-lo
- 22/02/2020 – Reunião com Fábio Bexiga para tirar duvidas
- 23/02/2020 a 07/03/2020 – Criar base de web server e conseguir fazer queries a base de dados
- 08/03/2020 a 21/03/2020 – Criar base de aplicação móvel
- 22/03/2020 a 04/04/2020 – Criar serviço REST
- 05/04/2020 a 11/04/2020 – Corrigir falhas apontadas na avaliação anterior
- 12/04/2020 a 25/04/2020 – Elaborar novos pontos do relatório e definir novo calendário

Para esta segunda entrega, com o conhecimento que adquirimos com as aulas de programação web e o facto de podermos esclarecer dúvidas a cerca de especificidades das línguas que estávamos a utilizar, a nossa capacidade para desenvolver código aumentou bastante e foi nesta entrega que começamos a fazer progresso no desenvolvimento propriamente dito.

Para o último calendário estabelecemos:

- 1ª semana – Implementar email *confirmation*
- 2ª semana – Implementar Login, começar implementação de Recuperar password
- 3ª semana – Terminar Recuperar password
- 4ª semana – Começar Ecrã Home
- 5ª semana – Terminar Ecrã Home
- 6ª semana – Começar Ecrã Search
- 7ª semana – Terminar Ecrã Search
- 8ª semana – Testes com utilizadores

Para esta última entrega tivemos problemas com a realização do que tínhamos proposto por termos de resolver alguns problemas pessoais devido a pandemia que decorre nos tempos atuais, mas com o adiamento de uma semana de entrega acabou por encaixar tudo nos prazos semanais que havíamos estabelecido.

Os problemas principais para esta fase foram o *setup* do emulador móvel e o facto de não termos conseguido resolver todos os problemas apontados depois dos testes com utilizadores pelo facto da semana de entrega coincidir com a semana de duas frequências finais e de os problemas terem sido mais que os esperados.

7. Resultados

7.1 Testes de Utilizadores

Ao contrário do que planeámos anteriormente, devido à situação em que vivemos, decidimos testar apenas com 5 utilizadores (nossos familiares), situando-se os mesmos entre os 25 e os 50 anos, e como já sabemos com estes números de utilizadores é possível encontrar pelo menos 75% dos problemas.

Realizamos então os testes de primeira instância presente no plano de testes desenvolvido para o relatório anterior, e podemos verificar os seguintes resultados:

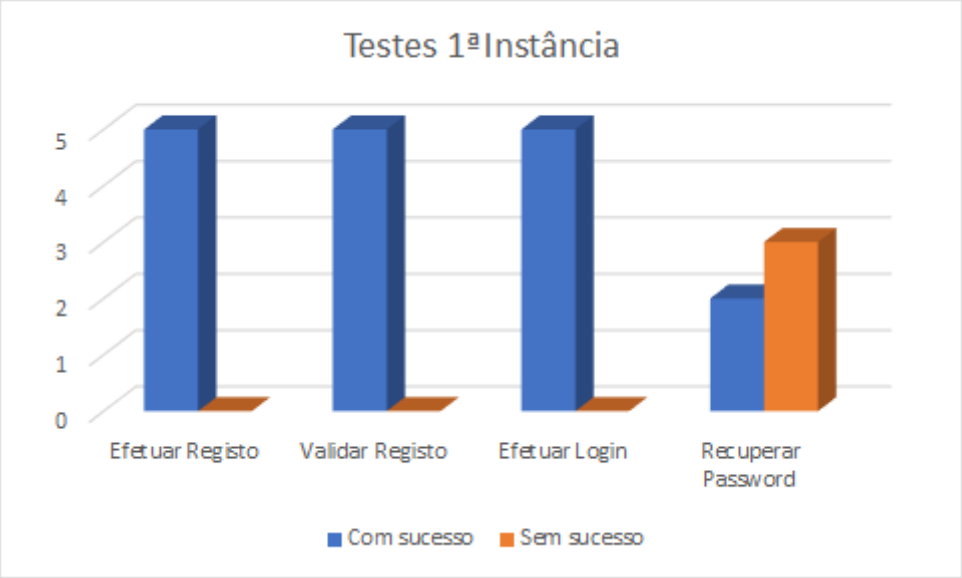


Figura 9: Resultados de testes de 1ª Instância

Nesta primeira instância os testes foram bastante básicos, sendo que apenas tivemos problemas a recuperar password, contudo apenas os 3 primeiros a testar não conseguiram, o que nos levou a tentar perceber porquê.

Depois de uma breve analise do código da página para alterar a palavra-passe reparamos que tínhamos o *domain* do servidor errado. Tivemos apenas alterar o *domain localhost* para o *IP* do servidor. Assim, com o problema resolvido, os dois últimos testers foram bem-sucedidos.

O primeiro pedido de alteração para os ecrãs destes testes foi que, quando a alteração da palavra passe é feita no dispositivo móvel, ao clicar no link enviado no email, o utilizador devia ser redirecionado para alterar a palavra passe dentro da aplicação, em vez de ser redirecionado para o browser. Com uma breve pesquisa averiguamos que isto poderia ser feito com *"universal linking"* e criamos um novo requisito para satisfazer esta alteração.

De seguida fomos testar a UX da nossa aplicação, sendo que a interface tem que corresponder às expectativas do utilizador e não do *developer*, podemos então auferir os resultados no grafico seguinte:

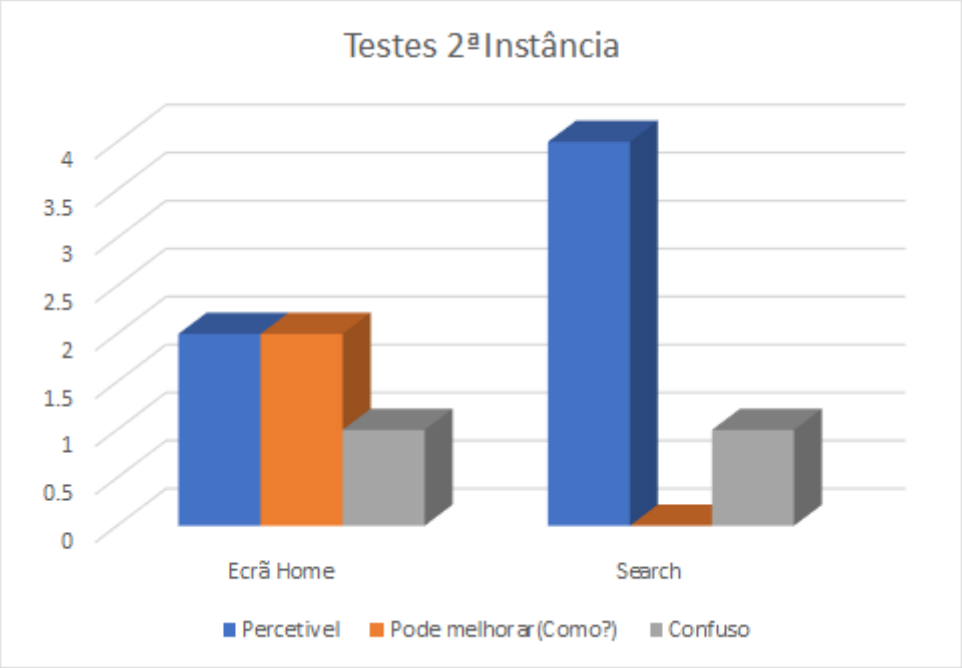


Figura 10: Resultados da User Experience

Nos testes de segunda instância tivemos pedidos de alterações para o ecrã home. Os utilizadores queixavam-se do tamanho das imagens e texto de cada visita. Por este motivo alterámos o layout da home page para apresentar as visitas em cartões maiores.

Para a search page criamos uma tab extra para dar a opção de o utilizador poder visualizar as visitas em cartões idênticos aos da home page, e mantivemos a opção de poder visualizar as visitas num formato mais compacto tal como o que tínhamos apresentado no protótipo.

7.2 Ecrãs finais

Como resultados finais, depois de realizarmos as alterações possíveis, ficamos com os seguintes ecrãs:

7.2.1 *Login Page*

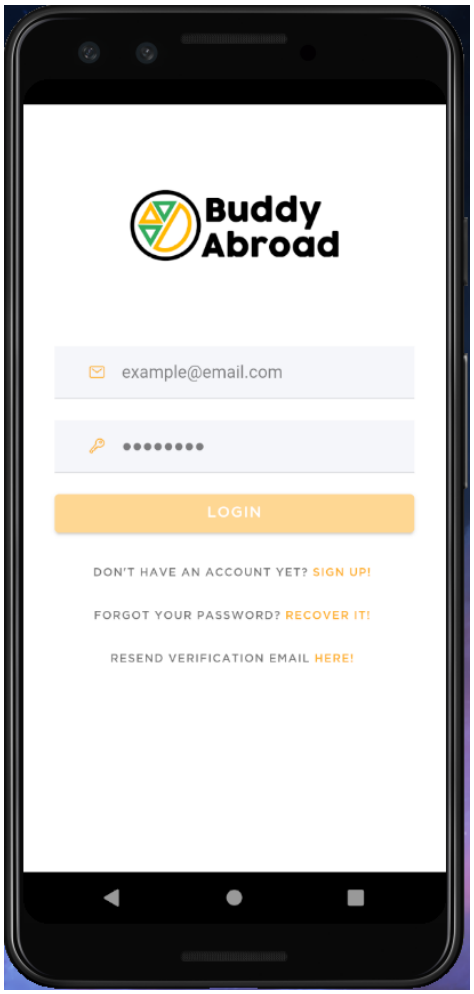


Figura 11:Ecrã Login final

Com esta página satisfazemos os requisitos **LOG1**

7.2.2 Sign Up Page

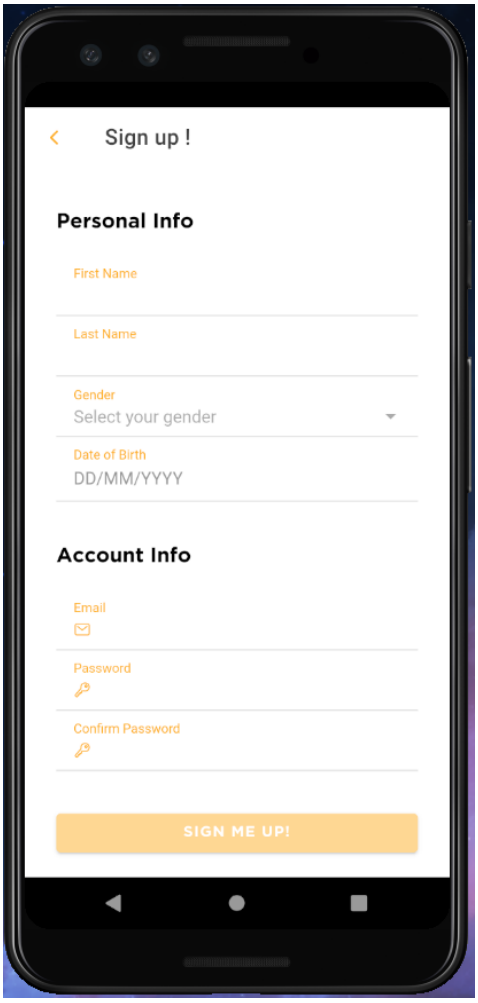


Figura 12:Sing Up page final

Com esta página satisfazemos os requisitos **RE11**

7.2.3 Recover Pass

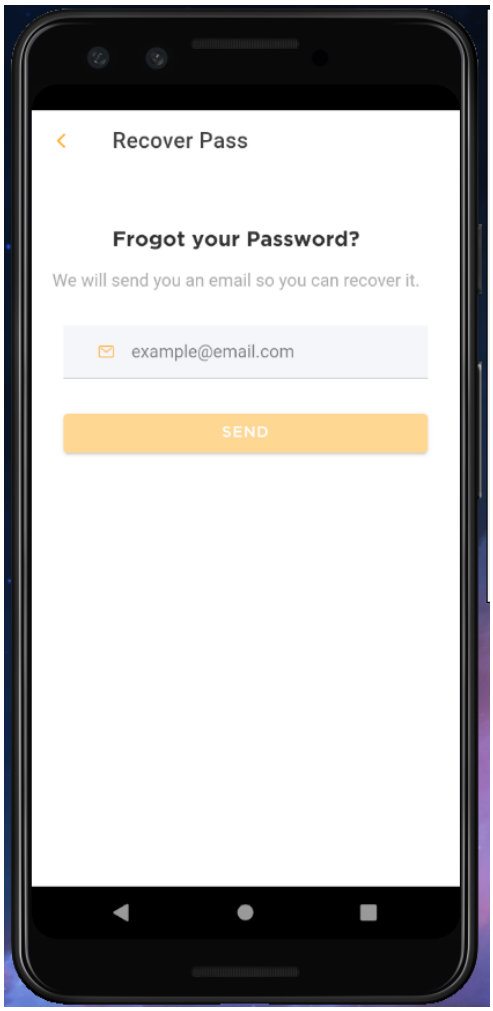


Figura 13: Recover pass fianl

Com esta página satisfazemos os requisitos **RE1**

7.2.4 Verify Account

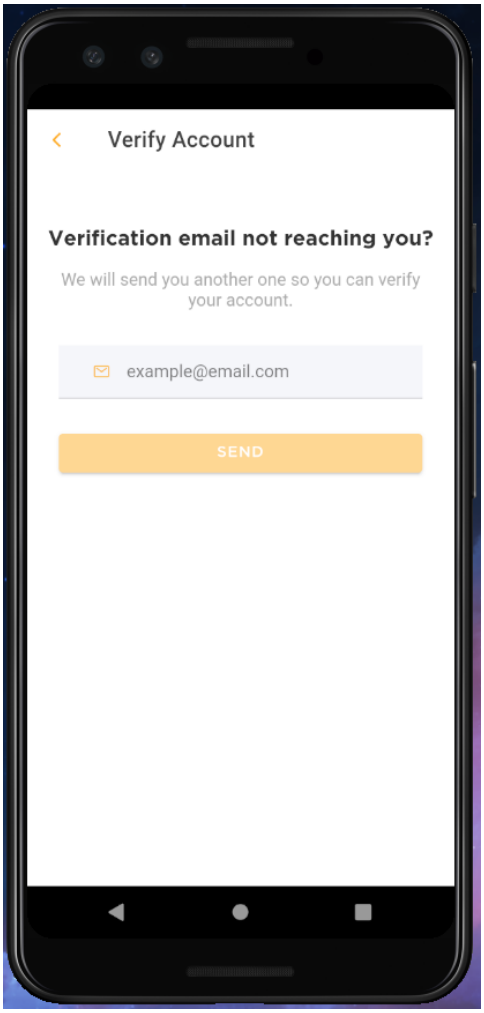


Figura 14: Ecrã verify account final

Com esta página satisfazemos os requisitos **RE1**

7.2.4 Sign up Successful

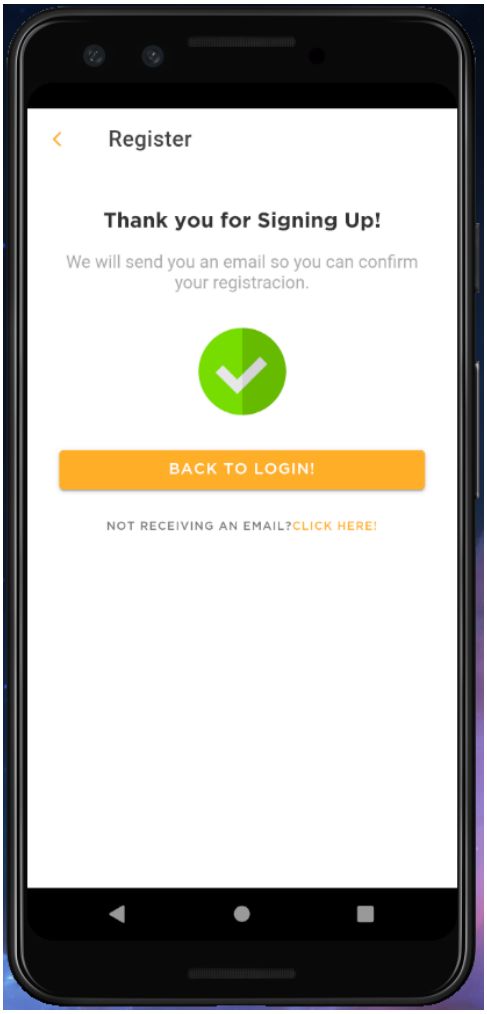


Figura 15: Register Successfull final

Com esta página satisfazemos os requisitos **RE11**

7.2.5 Home

Figura 16: Ecrã home final

Com esta página satisfazemos os requisitos **HOM1, HOM2**

7.2.5 Search

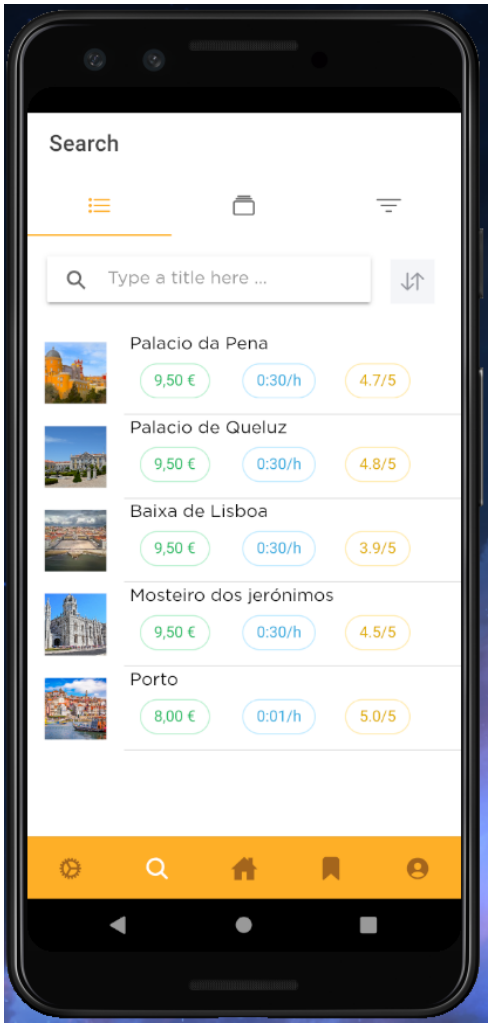


Figura 17: Ecrã search final

Com esta página satisfazemos os requisitos **SEA1**

7.2.6 *Change Password web page*

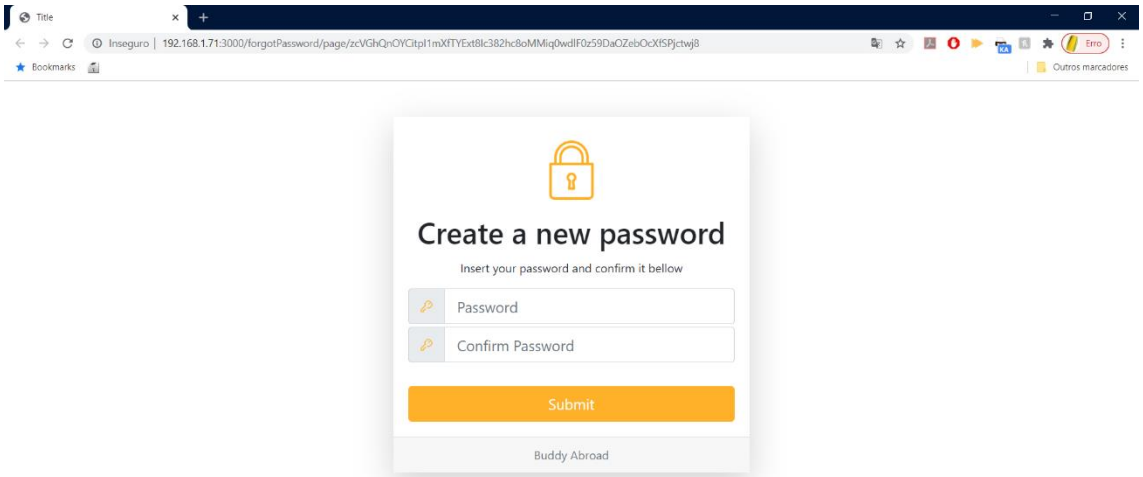


Figura 18: Change Password web page final

Com esta página satisfazemos os requisitos **APP1**

8. Conclusão e trabalhos futuros

Podemos concluir, depois desta última entrega, que a dimensão e complexidade deste trabalho eram maiores do que as que esperávamos quando nos propusemos para a realização deste TFC. Para além disto foi complicado começar a desenvolver o código propriamente dito, visto que só tivemos as cadeiras base para o fazer no segundo semestre, e os nossos conhecimentos só ficaram solidificados perto do final do semestre. Por este motivo, conseguimos agora apontar falhas a nível de código, que não fomos capazes de entender no início do seu desenvolvimento.

Foi possível também apercebermo-nos das enumeras soluções possíveis para arquiteturas de aplicações/plataformas móveis, e foi bastante interessante compreender como os vários conteúdos que aprendemos ao longo do curso se ligam quando realizamos projeto desta dimensão. Com a execução do trabalho foi possível compreender a importância de engenharia de software e do planeamento solido das bases do projeto. Conseguimos aplicar os conceitos de base de dados não de uma forma isolada, mas sim como uma peça que comunica com outras peças dentro da aplicação. Pusemos em uso os conhecimentos de *frameworks* que usamos nas cadeiras de Programação Web e Sistemas Moveis Empresariais.

Aprendemos e aplicamos conceitos que não foram postos em prática noutras cadeiras como a comunicação de App-Servidor através de uma *REST API*.

Olhando para trabalhos futuros, na nossa opinião, achamos extremamente importante que quem irá dar seguimento já tenha tido a cadeira de programação web, pois é nesta que são aprendidos conceitos base que usamos neste trabalho e é nela que o aluno se familiariza com as linguagens que usamos para o desenvolvimento tanto do *front* como do *back end.*

Caso isto não seja possível, aconselhamos que os nossos colegas façam o curso de angular e vejam os vídeos disponíveis nos **8.2.3 Recursos.**

No que toca onde começar, recomendamos os nossos colegas a começar por criar uma função no servidor, capaz de armazenar imagens na firestorage, receber o link para a imagem e guardá-la na base de dados.

(Uma possível solução disponível em **8.2.3 Recursos)**

Com isto feito podem passar a criar o ecrã de criação de visita que depois de terminado há de significar que os nossos colegas conseguiram compreender as bases de como fazer comunicações entre App-Servidor e fazer comunicações Server-Base. Com estes conceitos compreendidos, os nossos colegas vão conseguir desenvolver o resto da aplicação com menos dificuldades.

8.1 Bibliografia

"Aplicativos Móveis e Turismo: Um Estudo Quantitativo Aplicando a Teoria do Comportamento Planejado" (Filho, Batista, Cacho, & Soares, 2017)

Link: https://www.researchgate.net/profile/Luiz_Mendes-Filho/publication/316248519_Aplicativos_Moveis_e_Turismo_Um_Estudo_Quantitativo_Aplicando_a_Teoria_do_Comportamento_Planejado/links/58f77fca0f7e9b9a95d53cf1/Aplicativos-Moveis-e-Turismo-Um-Estudo-Quantitativo-Aplicando-a-Teoria-do-Comportamento-Planejado.pdf?origin=publication_detail

"Aplicativos Móveis e Turismo: Um Estudo Quantitativo Aplicando a Teoria do Comportamento Planejado / *Mobile Applications and Tourism: A Quantitative Study Applyng the Theory of Planned Behavior*" (Filho, Batista, Cacho, & Soares, 2017)

Link: http://www.ucs.br/etc/revistas/index.php/rosadosventos/article/view/4787

"Referência: O papel das aplicações móveis no turismo – o caso Zarco, Marta Sofia" (Ferreirinha, 2017)

Link:https://recipp.ipp.pt/bitstream/10400.22/11193/1/martaferreirinhaMMD2017.pdf

"*Lost in translation? Booking.com research exposes surprising gap between travel ambitions and reality*" (Booking.com, 2018)

Link: https://globalnews.booking.com/bookingcom-research-exposes-surprising-gap-between-travel-ambitions-and-reality/

8.2 Anexos

Instruções para instalação dos artefactos.

Link: https://drive.google.com/open?id=1VCuw4jz4sOFgRDXzy5rF6ve3XAlFvpBF

Repositório da aplicação movel criada em *Ionic* (*buddyAbroadApp*).

Link: https://github.com/a21702801/buddyAbroadApp

Repositório do *web server* criado em *node.Express* e *javascript* (*webServer*).

Link: https://github.com/a21702801/webServer

Solução possível para guardar imagens na cloud da *firestorage.*

Link: https://medium.com/@stardusteric/nodejs-with-firebase-storage-c6ddcf131ceb

Link: https://firebase.google.com/docs/storage/web/upload-files

Storie boards Pdf e Axure.

Link:https://drive.google.com/file/d/1fv3kzp7b_U5UxZmL8mUCy7n8JAz9K6o/view?usp=sharing

Keys para a API mailer

Link: https://drive.google.com/file/d/10rKSKUyb9Ou89SxcE-8n9urGW5R7XxfB/view?usp=sharing

Requisitos EasyBacklog:

Link:https://drive.google.com/file/d/1fvR2ePFv5W4KX4kJ41MYRB1memet1D/view?usp=sharing

8.2.2 Código

Index.js

Começando pelo documento "index.js" podemos dizer que é a raiz do nosso servidor, portanto, é aqui que o servidor trata de receber os pedidos feitos pela aplicação. É aqui que fazemos referência as middlewares utilizadas pelo servidor, bem como em forma semelhante, a referência às routes dos serviços que o servidor disponibiliza.

Para tratar das routes da aplicação estamos a utilizar o express.Router( ), que nos permite redirecionar os pedidos para as diretorias corretas. Um exemplo:

```
//Routing para pedidos de "auth"
const auth = require('./routes/auth');
app.use('/auth',auth);
```

Outra *middleware* que usamos é o *body-parser.json( )*, que nós é útil para analisar os corpos dos pedidos feitos pela aplicação (*Ionic*) ao servidor.

```
router.use(bodyParser.json());
```

Outra coisa importante a referir é que, para todas os pedidos feitos ao servidor, fazemos a seguinte alteração aos seus *headers*, para resolver o erro *"Cross-Origin Request Blocked"*:

```
app.use(function(req, res, next) { res.header("Access-
    Control-Allow-Origin", "*"); res.header("Access-
    Control-Allow-Headers", "*"); next();

});
```

PgErrHand.js

Este documento serve de auxílio para o debugging das comunicações feitas entre base de dados e servidor, sendo constituído por uma lista de erros:

```
let errorCodes = {
    "08003": "connection_does_not_exist",
    "08006": "connection_failure",
    "2F002": "modifying_sql_data_not_permitted",
    "57P03": "cannot_connect_now",
    "42601": "syntax_error",
    "42501": "insufficient_privilege",
    "42602": "invalid_name",
    "42622": "name_too_long",
    "42939": "reserved_name",
    "42703": "undefined_column",
    "42000": "syntax_error_or_access_rule_violation",
    "42P01": "undefined_table",
    "42P02": "undefined_parameter",
    "23505": "violates_unique_constraint"
};
```

Depois disto temos um condigo simples que vês se o código do erro obtido corresponde a algum erro na lista, e que dá print para a consola, da sua respetiva mensagem.

```
if (err === undefined) {
    console.log("No errors returned from Postgres")
}
else {
    console.log('Error code details:', errorCodes[err.code])
}
```

PgApi

Este documento contém o código que se encarrega de realizar queries a base de dados.

A primeira coisa a apontar é que, neste documento são feitas comunicações entre servidor e base de dados. Portanto, aqui utilizamos o Node.pg mencionado anteriormente. O Node.pg oferece duas classes para realizar comunicações entre server e base de dados, estas são: Client e Pool. Como é esperado que aplicação faça pedidos concorrentes ao servidor que por sua vez terá de fazer pedidos concorrentes á base de dados, optamos por utilizar a classe Pool que em suma, é um conjunto de clientes conectados a base de dados, através dos quais o servidor poderá fazer os seus pedidos de queries.

Para implementar a pool escrevemos o seguinte código:

```
//Importar a classe Pool
const {Pool} = require('pg');


//Criar Client para fazer conexão com a pg database
const pool = new Pool({
    user: "DiogoAzevedo",
```

```
    password: "1234",
    host: "localhost",
    port: 5432,
    database: "buddyAbroad",
    max: 20,
    _connectionTimeoutMillis: 0,
    idleTimeoutMillis: 10000
});
```

Depois da pool definida, para realizar queries temos apenas de escrever o SQL necessário, atribuí-lo a uma variável, e por fim, pedir a pool para realizar o query com SQL e valores que entendermos:

```
async function registerUser(email,firstName,lastName,
                            dateOfBirth,gender,password,callBack) {
    const sql = 'INSERT INTO users(email, first_name, last_name,
gender, birth_date, password) VALUES ($1, $2, $3, $4, $5, $6)';
    pool.query(sql,[email,firstName,lastName,gender,dateOfBirth,
        password], (err, res) => {
            callBack(err,res)
    })
}
```

Podemos verificar que após realizarmos o query chamamos uma função callback com argumentos err (error) e res (response). Isto serve para, depois de realizarmos o query, podermos realizar ações com o resultado do mesmo. Se este der um erro, a variável err vai ser do tipo Object e a variável res será do tipo unsigned. Caso não haja nenhum erro a varável err será do tipo unsigned e a variável res do tipo objeto.

Com esta informação podemos responder a aplicação se ocorreu um erro no servidor quando ela fez o seu pedido, ou se correr tudo bem, enviar a resposta para aplicação com o que for necessário. Por exemplo:

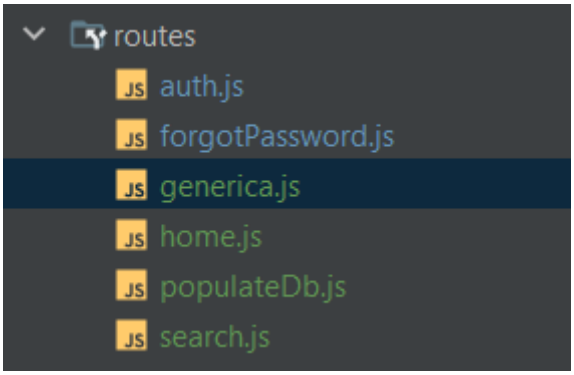
```
await findUserByEmail(registerRequest.email, async (err,result) =>{


    if(err !== undefined) {
        res.status(500).send('Server error.');
        return;
    } else {
    res.status(400).send(result)
}
```

Route genérica

Para criar uma route começamos por instanciar no **índex.js** qual é a route propriamente dita, e dizemos a *app* para a usar como *middleware*.

```
//Routing para pedidos de "populateDb"
const generica = require('./routes/generica');
app.use('/generica',generica);
```

Depois criamos um novo documento **.js**, no caso, chamado "genérica" e colocamo-lo dentro do folder "routes".



Agora, dentro da nova route restanos importar o router para receber os pedidos http feitos ao servidor e importar a api do *postgres* que criámos para fazer queries á base de dados.

```
const express = require('express');
let router = express.Router();

//Importar a api para fazer queries á base de dados
const pgApi = require('../database/pgApi.js');
const pgErrorHandler = require('../database/pgErrHand');
```

Um exemplo de código para receber um pedido "get" a esta route poderia ser:

```
router.route('/').get( async (req,res) => {
   res.send('Acedeste á route genérica :D !')
});
```

Mailer

Antes de mais, para que o mailer funcione, os nossos colegas teram de criar uma conta na https://www.mailgun.com/ e seguir as instruções do link seguinte:

https://documentation.mailgun.com/en/latest/quickstart-sending.html#how-to-start-sending-email

Depois da nova conta estar setup para enviar um email, podemos utilizar a mailer_api.js que nós criámos.

Primeiro precisamos de importar o mailer para a route em que o vamos utilizar:

```
//Importar a api para enviar emails
const mailerApi = require('../mailer/mailer_api');
```

Depois para enviar o email só temos chamar a funçar "sendMailViaApi" e fornecer os argumentos: from, to, subject, text, userId, emailType.

Um exemplo de código para enviar um email seria:

```
router.route('/mail').get( async (req,res) => {

   res.send('A enviar email!')

   const from = 'Buddy Abroad <BuddyAbroad@mailgun.org>'
   const email = 'oTeuEmail@mail.com'
   const subject = 'Mail Generico'
   const text = 'Email enviado com a API!'
   const userId = 156
   const emailType = 1
   mailerApi.sendMailViaApi(from, email, subject, text, userId, emailType)

});
```

Jwt autentication

A autentificação está a ser feita da seguinte forma: O servidor cria tokens de acesso com 10 minutos de validade envia-os para a aplicação quando o utilizador faz login.

```
//Gerar Access token
function generateAccessToken(user) {
   return jwt.sign(user, process.env.ACCESS_TOKEN_SECRET, { expiresIn: '10m' });
}
```

Quando o utilizador faz um pedido, a aplicação te um "HttpInterceptor" que intercepta o pedido e antes do enviar, cria um header "Authorization" que contém um array com o "access_token" e o "refresh_token".

```
const cloned = req.clone({
   headers: req.headers.set('Authorization', [token.access_token, token.refresh_token])
});
console.log([token.access_token, token.refresh_token]);
return next.handle(cloned);
```

Depois, no servidor, quando nos queremos criar um pedido que precisa de ser autenticado, precisamos apenas de importar o "jwtAuth.js" e usar a função "autenticateToken".

Nesta função, se o token estiver valido, nada acontece, se o token estiver expirado, a função verifica se o refresh token desse mesmo pedido se encontra guardao na base de dados, se estiver, a função cria automaticamente um token de acesso e o pedido prossegue normalmente, se não estiver guardado na base de dados o pedido é rejeitado. O mesmo acontece para tokens inválidos/inexistentes.

Um código exemplo de use desta função pode ser:

```
router.route('/jwt').get( async (req,res) => {
   jwtAuth.autenticateToken(req,res, () => {
      res.status(200).send({info: 'Accesso concedido :D!'})
   })
});
```

8.2.3 Recursos

Aqui disponibilizamos alguns recursos que considerámos serem bons pontos de começo para quem terá de dar continuidade ao trabalho

Angular, getting started: Aqui somos introduzidos ao angular, a framework na qual o ionic tem as suas bases. No link disponibilizado, os autores proporcionam um tutorial de como criar uma aplicação em angular e explicam os essenciais da framework.

Link: https://angular.io/tutorial

PostgreSQL + Javasript: Nestes vídeos somos introduzidos a como realizar comunicações entre um uma base de dados em PostgreSQL e um webserver em javascript. No primeiro link, o autor explica como fazer uma ligação do tipo Client e a realizar queries. Também são abordados temas como Observables e Promisses, o que achamos que pode ajudar a introdução dos procimos colegas as classes utilizadas no projeto, bem como os seus conceitos. No segundo link o autor do vídeo explica como se cria uma Pool de clientes para comunicar com a base de dados. Visto que esta é a forma como comunicamos entre o servidor e a base de dados achamos que este vídeo também pode ser uma boa ajuda á compreensão da classe que utilizamos.

1ºLink: https://www.youtube.com/watch?v=ufdHsFClAk0&t=0s

2ºLink: https://www.youtube.com/watch?v=GTeCtIoV2Tw&t=0s

8.2.4 Glossário

Framework: Neste caso, quando falamos de framework, como por exemplo o Ionic, estamos a falar de uma "caixa de ferramentas" de software, que usamos para realizar o trabalho.

REST API: Uma REST API é apenas algo que nos permite comunicar entre cliente e servidor de uma forma standard. Um vídeo que pode clarificar isto em detalhe, bem como explicar de uma forma breve como funciona a classe que usamos (HttpClient) pode ser encontrado no seguinte.

Link: https://www.youtube.com/watch?v=_05v0mrNLh0&t=20s (Fireship,2017)