UNIVERSIDADE AUTÓNOMA DE LISBOA
LUÍS DE CAMÕES

DEPARTAMENTO DE CIÊNCIAS E TECNOLOGIAS

PROJETO FINAL - TOURSPOT

Projeto final de curso

Autores: Miguel Lopes 20160087,
Margarida Duarte 20160348,
Gonçalo Neto 20160560

Orientador: Professor Doutor Daniel Silvestre

Julho de 2019
Lisboa

(Página em Branco)

1. Resumo

O rápido crescimento na utilização de smartphones e desenvolvimento de novas aplicações criou uma forma nova e diferente de viajar e de conhecer novos sítios bem como, de interação social.

Neste sentido, vimos a necessidade de desenvolvermos uma aplicação online de turismo que permita aos nossos utilizadores usarem a nossa aplicação como guias para prestar serviços ou como clientes para usarem os nossos serviços.

O nosso objetivo é facilitar e melhorar a experiência turística de um indivíduo a através da realização de tours pela cidade do destino turístico, pondo em contacto pessoas de diferentes culturas bem como, facilitar a mobilidade e o conhecimento de gastronomia da cidade tendo por base a utilização de uma única aplicação.

Palavras-chave: aplicações, turismo, viajar, tours

2. Abstract

The fast growth in the use of smartphones and the development of new applications are a new and different way of travel and to discover new maps as well as social interaction.

In this respect, we saw the need of developing an online tourism application in which it allows our customers to use our application as a guide in order to provide services or as customers to use our services.

The aim is to simplify and improve the tourist experience of an individual by conducting city tours of the tourist's destination, with the support of people from different cultures as well as easing mobility and the knowledge of the city's gastronomy experience using a single application.

Keywords: applications, tourism, travel, tours

3. Índice

4. Lista de Imagens

5. Lista de Siglas e Acrónimos

UI - User Interface

REST - Representational State Transfer

JSON - JavaScript Object Notation

API - Application Programming Interface

SQL - Structured Query Language

iOS - iPhone OS

UWP - Universal Windows Plataform

JSX - JavaScript XML

HTML - HyperText Markup Language

CSS - Cascading Style Sheets

APK - Android Application Package

6. Introdução

Desde indivíduos a organizações podem desenvolver aplicações. Sendo que, isto resultou no rápido desenvolvimento de aplicações comerciais e não comerciais para diversas finalidades [3]. Devido à crescente popularidade dos smartphones e das suas aplicações, as tecnologias móveis têm mudado a forma de viajar [1].

Com processadores eficientes, sistemas operativos modernos, acesso à Internet em banda larga, ecrãs touch com boa visualização e interfaces fáceis ao utilizador [1] os smartphones têm-se tornado grande parte da rotina de um indivíduo. Neste sentido, as tecnologias móveis são uma ferramenta que pode oferecer uma ampla gama de possibilidades para ajudar turistas no planeamento das suas viagens em casa e durante a viagem [3]. Sendo que, os smartphones têm a capacidade de ligar pessoas e informações através da troca de dados baseados na localização e informações sociais. Por isto, os smartphones são uma ferramenta essencial para o turismo [3].

Neste sentido, vimos a necessidade de desenvolver uma aplicação destinada ao uso de qualquer utilizador sempre que vá viajar para um novo lugar.

A aplicação que desenvolvemos foi chamada de TourSpot e é uma aplicação online que permite aos nossos utilizadores usarem a nossa aplicação como guias para prestar serviços ou como clientes para usarem os nossos serviços.

O guia pode prestar serviços a qualquer cliente da nossa app através de um serviço público ou privado. Sendo que, numa Tour Pública é realizada uma tour num grupo de mais turistas e visitado os principais pontos turísticos e numa Tour Privada o guia realiza uma tour apenas para o cliente que requisitou o serviço. A cada guia é autorizada a realização de apenas uma Tour Privada por cada dia.

Em relação ao cliente, este pode requisitar os nossos serviços de forma pública ou privada. Ao cliente é dada a opção de realizar uma Tour Pública num grupo de mais turistas ou uma Tour Privada com possibilidade de escolha do guia e de personalização do percurso a efetuar, por exemplo, na escolha de lugares que pretende visitar.

Na aplicação o cliente pode utilizar ainda outras secções de forma a melhorar a sua experiência turística através da secção de restaurantes que dá a conhecer diversos restaurantes de acordo com as preferências da pessoa e que variam em preço; a secção dos mapas a qual tem como objetivo facilitar a mobilidade e ainda uma seção para guardar documentos em pdf (por exemplo, bilhetes de avião).

Ao longo deste projeto vamos explicar em maior detalhe as várias fases do projeto de forma a elucidar acerca do nosso problema, a solução encontrada referindo a arquitetura e tecnologias usadas.

7. Problema

7.1 Definição do Problema

No mercado desenvolveram-se Aplicações de Viagens especificamente dirigidos a turistas como, por exemplo, Airbnb, TripAdvisor e Skyscanner bem como, aplicações usadas num contexto de viagem típica como, por exemplo, Google Map, Imoney e Instagram [2].

As aplicações que existem atualmente no mercado são focadas nas consultas e reservas de hotéis, restaurantes, voos e cruzeiros sendo que, não há nenhuma aplicação que seja útil no planeamento da viagem e ao mesmo tempo que seja útil durante a viagem.

Desta forma, definimos como problema o facto das aplicações relativas ao turismo não terem nenhuma funcionalidade que tenha como objetivo promover o conhecimento da cidade visitada e o contacto com pessoas de culturas diferentes.

Das aplicações que pesquisamos não encontramos nenhuma aplicação que desse a possibilidade ao utilizador de ter acesso a guias de forma gratuita para ir numa tour bem como, uma aplicação que além deste ponto fulcral ainda possibilitasse na mesma aplicação ter acesso ao google Maps, a restaurantes e também a guardar documentos pdf.

7.2 Estado da Arte

Há indicações de que as tecnologias móveis, especialmente através do surgimento de smarphones e apps, tornar-se-ão na próxima onda de inovação que impulsiona viagens e turismo [1].

A utilização de apps permite ao utilizador estabelecer ligação pessoal devido à sua personalização bem como, as apps também permitem serviços exclusivos altamente interativos ao utilizador [2] devido às suas características de alta conetividade, comunicação, consumo e criação de conteúdo [1] bem como, a capacidade das aplicações do smartphone de fazer download e instalar aplicações [2].

No turismo, como em outras áreas da vida, as aplicações do smartphone têm a capacidade de reformular as dimensões da vida social, incluindo viagens [3].

Como tal, o turismo pode ser visto como uma rede de pessoas, coisas e lugares [3]. Os smartphones guiam as pessoas e de acordo com os destinos e atrações ligam outras pessoas, permitem marcações temporais versáteis e negociar necessidades turísticas imediatas [3].

Para desenvolver este tipo de aplicações pode-se recorrer a dois tipos de abordagem: uma abordagem multiplataforma e uma abordagem de design orientada para a marca.

Numa abordagem multiplataforma, o designer é limitado pelas diretrizes de cada plataforma [5]. Este tipo de abordagem é vantajoso para aplicações com uma interface complexa e que tenha como principal objetivo atrair utilizadores com maior probabilidade de passar muito tempo nestas plataformas, sejam elas Android ou iOS [5].

Por outro lado, uma abordagem de design orientada para a marca leva em conta os fatores específicos das marcas em detrimento de aspetos específicos de plataforma [5]. Neste sentido, esta abordagem é vantajosa se a aplicação for acedida pelo mesmo utilizador em diferentes plataformas [5].

Existem linguagens que podem ser usadas para o desenvolvimento de aplicações nativas como Kotlin que é suportada apenas para Android ou Swift que é apenas para o sistema operativo iOS. As aplicações nativas incluem inúmeras vantagens devido às seguintes características: amplas funcionalidades devido ao uso das capacidades do dispositivo subjacente, desempenho de software rápido e responsivo e garantia de qualidade através de classificações em lojas de aplicações [7]. No entanto, as aplicações nativas incluem desvantagens tais como, múltiplas bases de código, os desenvolvedores terem de criar e contruir uma base de código para cada plataforma e o tempo despendido [7].

Por outro lado, existem outras linguagens que funcionam em múltiplas plataformas como, por exemplo, o Xamarin, o Flutter e o React Native as quais são suportadas tanto pelo sistema operativo iOS como pelo sistema operativo Android.

Em relação ao React Native este apresenta inúmeras vantagens na sua utilização tais como, oferece uma base de código única pois o mesmo código serve para múltiplas plataformas, tempo de desenvolvimento reduzido devido ao facto de não ser necessário desenvolver códigos para diferentes plataformas, fácil transição para Desenvolvedores Web devido ao desenvolvimento de apps utilizar JavaScript, é open source e, a opção de Hot Reload que facilita aos desenvolvedores ver as suas alterações em tempo real [8]. Aplicações como o Facebook, Instagram, UberEATS, Airbnb e Nubank foram desenvolvidos com o React Native [6].

7.3 Principais Concorrentes do Mercado

Trip Advisor

TripAdvisor é considerada a maior plataforma de viagens do mundo pois ajuda 490 milhões de viajantes todos os meses [4].

O seu funcionamento baseia-se em empresas divulgarem o seu local e assim utilizadores avaliam consoante a sua experiência. Tem funções de comparação de preços em hotéis, voos, cruzeiros, permite também reservar excursões existentes e atrações populares bem como reservar mesas em restaurantes. [9]

No entanto, não existe forma do próprio utilizador criar excursões personalizadas.

Expedia

Permite que os utilizadores pesquisem e reservem voos, hotéis, carros, cruzeiros, passeios e vários pacotes combinados. O site/app utiliza Sistemas de Distribuição Global (GDS) como o Amadeus ou Sabre para reservas. [10]

TripIt

Aplicação de planeamento de viagens na qual existem duas versões. Uma grátis e outra paga.

A versão standard permite criação de itinerários e a sua partilha com amigos e família. Enquanto que, a Pro envia alertas de atrasos ou cancelamentos de voos, informação sobre reembolso, opções alternativas de voos, entre outros. [11]

Google Trips

Esta aplicação é basicamente um guia personalizado que fornece a possibilidade de criar passeios personalizados com mapas, informação sobre restaurantes e monumentos no local.

Reúne todas as informações da viagem na conta gmail do utilizador e armazena-as offline, dando acesso à aplicação sem wi-fi ou dados móveis.

No entanto, esta aplicação vai ser descontinuada a partir deste verão. [12]

Klook

Esta app apresenta atrações, atividades, restaurantes, entre outros como muitas das outras aplicações.

Além disso tem a funcionalidade de procurar passeios turísticos com um guia, a diferença entre a nossa aplicação é que Klook permite apenas procurar passeios guiados já existentes [13], ao contrário da nossa que permite ao utilizador a criação personalizada com local, data e hora.

7.4 Objetivo

Neste projeto, desenvolvemos uma aplicação utilizando uma abordagem de design orientada para a marca de forma a não existir diferença entre o design UI da app em android e iOS. Dentro das linguagens existentes para o desenvolvimento de aplicações

escolhemos React Native por ser uma linguagem que possibilita com o mesmo código-fonte gerar versões nativas para iOS e Android, o que poupa muitas horas e custos com programação [6]. Existem outras tecnologias que permitem isso, porém diferente delas, o React Native apresenta uma excelente performance [6].

Assim, tivemos como grande objetivo desenvolver uma nova aplicação de turismo de acesso rápido e fácil que fosse utilizada para o planeamento e utilização durante as viagens por parte de qualquer utilizador com idade superior a 18 anos.

Nesta aplicação, é permitido ao utilizador:

1) Fazer tours públicas e gratuitas em qualquer cidade;
2) Fazer tours privadas de acordo com as preferências do utilizador;
3) Procurar restaurantes ordenados por classificação;
4) Guardar documentos em formato pdf;
5) Ter acesso ao Google Maps.

8. Solução Do Problema

Para colmatar o problema em questão, sentimos a necessidade de "atacar" os pontos fracos das aplicações atuais relacionadas ao turismo.

Nos dias de hoje as empresas relacionadas a este tema focam-se em desenvolver aplicações para captar o interesse dos utilizadores, mas a preocupação em atrair clientes e ao mesmo tempo ter lucro afasta-as das reais necessidades do utilizador comum dos dias de hoje.

Com isto, decidimos criar uma aplicação que ofereça os mesmos serviços, de graça e mais próxima do utilizador do que qualquer outra aplicação da concorrência, aqui todos os utilizadores têm a liberdade de visitar os pontos de atração turística mais famosos de cada cidade sem gastar um único cêntimo, ou pagarem para ter uma experiência mais imersiva, personalizada e pessoal, o utilizador não precisa de se limitar às aplicações em

que a informação é grátis mas qualquer tipo de experiência ou interação tem que ser cobrada, algo que na nossa aplicação se encontra a apenas um toque de distância.

8.1 Fases do Project

8.1.1 Planeamento

Inicialmente começamos por elaborar, recorrendo ao Project, uma planificação das tarefas que tínhamos de executar. Neste sentido, uma primeira parte de planeamento foi necessária para definir os requisitos do software, definição de funcionalidades e de interfaces bem como, planificação do design.

Esta primeira fase foi essencial para nós para conseguirmos fazer pesquisa acerca das aplicações já existentes no mercado e conseguirmos definir objetivos que diferenciassem a nossa aplicação de outras já existentes e também definir as interfaces para os nossos utilizadores sendo que, a nossa aplicação definiu dois tipos de interface uma para cliente e outra para guia. Inicialmente, também tínhamos pensado em definir uma interface para o administrador no entanto, durante a elaboração do projeto chegamos à conclusão que não era necessária visto que, conseguimos administrar a nossa aplicação através do Firebase e, portanto, esta seria inútil.

Esta fase permitiu-nos ainda escolher a linguagem mais adequada para conseguir atingir os nossos objetivos.

Task Name	Duration	Start	Finish	Predecessors	Resource Names
0 App	**61.5 days**	**Mon 3/18/19**	**Tue 6/11/19**		
1 Planeamento	**6.38 days**	**Mon 3/18/19**	**Tue 3/26/19**		
1.1 Requisitos do Software	**6.38 days**	**Mon 3/18/19**	**Tue 3/26/19**		
1.1.1 Objetivos do projeto	7 hrs	Mon 3/18/19	Mon 3/18/19		Gonçalo Neto,Margarida Duarte,Miguel Lopes
1.1.2 Escolha da linguagem de programação	2 hrs	Mon 3/18/19	Tue 3/19/19	3	Gonçalo Neto,Margarida Duarte,Miguel Lopes
1.1.3 Escolher o IDE a utilizar	4 hrs	Tue 3/19/19	Tue 3/19/19	4	Gonçalo Neto,Margarida Duarte,Miguel Lopes
1.1.4 Definir funcionalidades	**2.5 days**	**Tue 3/19/19**	**Fri 3/22/19**	**5**	
1.1.4.1 Utilizador	9 hrs	Tue 3/19/19	Wed 3/20/19	5	Gonçalo Neto,Margarida Duarte,Miguel Lopes
1.1.4.2 Guia	7 hrs	Wed 3/20/19	Thu 3/21/19	7	Gonçalo Neto,Margarida Duarte,Miguel Lopes
1.1.4.3 Administrador	4 hrs	Thu 3/21/19	Fri 3/22/19	8	Gonçalo Neto,Margarida Duarte,Miguel Lopes
1.1.5 Definir interfaces	**1.25 days**	**Fri 3/22/19**	**Mon 3/25/19**	**9**	
1.1.5.1 Utilizador	5 hrs	Fri 3/22/19	Fri 3/22/19	9	Gonçalo Neto,Margarida Duarte,Miguel Lopes
1.1.5.2 Guia	4 hrs	Fri 3/22/19	Mon 3/25/19	11	Gonçalo Neto,Margarida Duarte,Miguel Lopes
1.1.5.3 Administrador	1 hr	Mon 3/25/19	Mon 3/25/19	12	Gonçalo Neto,Margarida Duarte,Miguel Lopes
1.1.6 Planificação do design	**1 day**	**Mon 3/25/19**	**Tue 3/26/19**	**13**	
1.1.6.1 Criação dos Protótipos	1 day	Mon 3/25/19	Tue 3/26/19	13	Gonçalo Neto,Margarida Duarte,Miguel Lopes

Figura 1 Planeamento

8.1.2 Desenvolvimento do software

Na fase de Desenvolvimento de software indicamos as tarefas de forma pormenorizadas necessárias ao desenvolvimento da nossa aplicação sendo que, quando começamos efetivamente no desenvolvimento da aplicação não elaboramos as tarefas descritas pela mesma ordem que estão expostas no project.

Ao longo do projeto fomos fazendo alterações neste planeamento inicial e, neste sentido, houveram tarefas que nós tivemos de retirar e tarefas que tivemos de substituir por outras.

Na criação de mecanismos de autenticação não procedemos à criação do mecanismo de recuperação de dados pois a documentação existente do firebase estava desatualizada.

Task Name	Duration	Start	Finish	Predecessors	Resource Names
2 Desenvolvimento do software	**42.13 days**	**Tue 3/26/19**	**Thu 5/23/19**	**15**	
2.1 Criação da Base de Dados	**0.88 days**	**Tue 3/26/19**	**Wed 3/27/19**	**15**	
2.1.1 Setup do firebase	7 hrs	Tue 3/26/19	Wed 3/27/19	15	Miguel Lopes
2.2 Criação do mecanismo de autenticação	**4.25 days**	**Wed 3/27/19**	**Tue 4/2/19**	**18**	
2.2.1 Estabelecer ligação entre DB e app	2 days	Wed 3/27/19	Fri 3/29/19	18	Miguel Lopes,Margarida Duarte
2.2.2 Diferenciação entre utilizador e guia	1 day	Fri 3/29/19	Mon 4/1/19	20	Miguel Lopes
2.2.3 Verificação de dados de login	5 hrs	Mon 4/1/19	Mon 4/1/19	21	Miguel Lopes
2.2.4 Criação do mecanismo de recuperação de dados	5 hrs	Mon 4/1/19	Tue 4/2/19	22	Miguel Lopes
2.3 Desenvolvimento da aplicação pós-login	**37 days**	**Tue 4/2/19**	**Thu 5/23/19**	**23**	
2.3.1 Criação da barra principal	**2 days**	**Tue 4/2/19**	**Thu 4/4/19**	**23**	
2.3.1.1 Design e funcionalidade	2 days	Tue 4/2/19	Thu 4/4/19	23	Margarida Duarte,Miguel Lopes
2.3.2 Criação da página principal	**8 days**	**Thu 4/4/19**	**Tue 4/16/19**	**26**	
2.3.2.1 Utilizador	**5 days**	**Thu 4/4/19**	**Thu 4/11/19**	**26**	
2.3.2.1.1 Criação das abas: próximo destino, minhas v	2 days	Thu 4/4/19	Mon 4/8/19	26	Margarida Duarte
2.3.2.1.2 Conclusão do restante design	3 days	Mon 4/8/19	Thu 4/11/19	29	Margarida Duarte
2.3.2.2 Guia	**3 days**	**Thu 4/11/19**	**Tue 4/16/19**	**30**	
2.3.2.2.1 Criação das abas: organizar eventos e meus	1 day	Thu 4/11/19	Fri 4/12/19	30	Margarida Duarte
2.3.2.2.2 Conclusão do restante design	2 days	Fri 4/12/19	Tue 4/16/19	32	Margarida Duarte
2.3.3 Criação da página de perfil	**5 days**	**Thu 4/4/19**	**Thu 4/11/19**	**26**	
2.3.3.1 Criação dos campos	1 day	Thu 4/4/19	Fri 4/5/19	26	Gonçalo Neto
2.3.3.2 Criação da funcionalidade de edição	2 days	Fri 4/5/19	Tue 4/9/19	35	Gonçalo Neto
2.3.3.3 Conclusão do restante design	2 days	Tue 4/9/19	Thu 4/11/19	36	Gonçalo Neto
2.3.4 Criação da página opções	**3 days**	**Thu 4/4/19**	**Tue 4/9/19**	**26**	
2.3.4.1 Criação das abas: alterar password/email, FAQS,	1 day	Thu 4/4/19	Fri 4/5/19	26	Miguel Lopes
2.3.4.2 Conclusão do restante design	2 days	Fri 4/5/19	Tue 4/9/19	39	Miguel Lopes
2.3.5 Criação da página Próximo Destino	**12 days**	**Tue 4/9/19**	**Thu 4/25/19**	**40**	
2.3.5.1 Criação dos campos de procura: data, país e cida	1 day	Tue 4/9/19	Wed 4/10/19	40	Miguel Lopes
2.3.5.2 Criação das funcionalidades de eventos: presenç	3 days	Wed 4/10/19	Mon 4/15/19	42	Miguel Lopes,Margarida Duarte

Figura 2 Desenvolvimento de Software I

Na barra principal além dos ícones da home, perfil e opções adicionamos um ícone de documentos em que permite ao utilizador fazer upload de documentos em formato pdf.

Em relação à interface do cliente, a criação das páginas de Locais Úteis, Hotéis, Bem-Estar, Entretenimento, Cultura, Desporto e Transportes não foram desenvolvidas pelo facto das APIS que existem não nos autorizarem o acesso.

Apenas conseguimos utilizar a API do Zomato para a realização da página dos Restaurantes em que o cliente basta inserir o nome do país e cidade que se encontra e o tipo de comida que pretende e são-lhe mostrados por ordem crescente de classificação os melhores 20 restaurantes.

Ainda em alternativa à página dos Transportes fizemos uma página Maps em que permite ao utilizador escrever a origem ou utilizar a sua localização atual e escrever o destino para onde quer ir sendo que, depois quando o utilizador clica em OK é redirecionado para a aplicação do Google Maps com os parâmetros que foram inseridos anteriormente e é-lhe indicado o caminho mais próximo e os respetivos transportes.

Task Name	Duration	Start	Finish	Predecessors	Resource Names
2.3.5.3 Criação do botão de tour privada	**8 days**	**Mon 4/15/19**	**Thu 4/25/19**	**43**	
2.3.5.3.1 Criação da página de pagamento (dummy)	2 days	Mon 4/15/19	Wed 4/17/19	43	Miguel Lopes
2.3.5.3.2 Criação da página dos guias disponíveis	3 days	Wed 4/17/19	Mon 4/22/19	45	Miguel Lopes,Gonçalo Neto,Margarida Duarte
2.3.5.3.3 Criação da funcionalidade chat	3 days	Mon 4/22/19	Thu 4/25/19	46	Miguel Lopes,Gonçalo Neto,Margarida Duarte
2.3.6 Criação da página Minhas Viagens	**7 days**	**Thu 4/11/19**	**Mon 4/22/19**	**37**	
2.3.6.1 Criação da funcionalidade de mostragem do histó	2 days	Thu 4/11/19	Mon 4/15/19	37	Gonçalo Neto
2.3.6.2 Criação da funcionalidade de detalhes das tours	3 days	Mon 4/15/19	Thu 4/18/19	49	Gonçalo Neto
2.3.6.3 Conclusão do restante design	2 days	Thu 4/18/19	Mon 4/22/19	50	Gonçalo Neto
2.3.7 Criação da página de Locais Úteis	**13 days**	**Tue 4/16/19**	**Fri 5/3/19**	**33**	
2.3.7.1 Criação das abas: hotéis, restaurantes, bem-estar	1 day	Tue 4/16/19	Wed 4/17/19	33	Margarida Duarte
2.3.7.2 Criação da página de hotéis	**5 days**	**Wed 4/17/19**	**Wed 4/24/19**	**53**	
2.3.7.2.1 Criação dos campos de procura: país, cidade	2 days	Wed 4/17/19	Fri 4/19/19	53	Margarida Duarte
2.3.7.2.2 Criação da funcionalidade de mostragem de	3 days	Fri 4/19/19	Wed 4/24/19	55	Margarida Duarte
2.3.7.3 Criação da página de restaurantes	**4 days**	**Wed 4/24/19**	**Tue 4/30/19**	**56**	
2.3.7.3.1 Criação dos campos de procura: país, cidade	2 days	Wed 4/24/19	Fri 4/26/19	56	Margarida Duarte
2.3.7.3.2 Criação da funcionalidade de mostragem de	2 days	Fri 4/26/19	Tue 4/30/19	58	Margarida Duarte
2.3.7.4 Criação da página de bem-estar	**3 days**	**Tue 4/30/19**	**Fri 5/3/19**	**59**	
2.3.7.4.1 Criação dos campos de procura: país, cidade	1 day	Tue 4/30/19	Wed 5/1/19	59	Margarida Duarte
2.3.7.4.2 Criação da funcionalidade de mostragem de	2 days	Wed 5/1/19	Fri 5/3/19	61	Margarida Duarte
2.3.7.5 Criação da página de entretenimento	**4 days**	**Mon 4/22/19**	**Fri 4/26/19**	**51**	
2.3.7.5.1 Criação dos campos de procura: país, cidade	2 days	Mon 4/22/19	Wed 4/24/19	51	Gonçalo Neto
2.3.7.5.2 Criação da funcionalidade de mostragem de	2 days	Wed 4/24/19	Fri 4/26/19	64	Gonçalo Neto
2.3.7.6 Criação da página de cultura	**4 days**	**Fri 4/26/19**	**Thu 5/2/19**	**65**	
2.3.7.6.1 Criação dos campos de procura: país, cidade	2 days	Fri 4/26/19	Tue 4/30/19	65	Gonçalo Neto
2.3.7.6.2 Criação da funcionalidade de mostragem de	2 days	Tue 4/30/19	Thu 5/2/19	67	Gonçalo Neto
2.3.7.7 Criação da página de desporto	**4 days**	**Thu 4/25/19**	**Wed 5/1/19**	**47**	
2.3.7.7.1 Criação dos campos de procura: país, cidade	2 days	Thu 4/25/19	Mon 4/29/19	47	Miguel Lopes
2.3.7.7.2 Criação da funcionalidade de mostragem de	2 days	Mon 4/29/19	Wed 5/1/19	70	Miguel Lopes

Figura 3 Desenvolvimento de Software II

Nas Opções, em relação à página de alterar password/email apenas permitimos ao utilizador a funcionalidade de alterar password e ainda criamos uma página para a Política de Privacidade.

Task Name	Duration	Start	Finish	Predecessors	Resource Names
2.3.8 Criação da página transportes	**5 days**	**Wed 5/1/19**	**Wed 5/8/19**	**71**	
2.3.8.1 Criação dos campos de procura Origem e Destino	2 days	Wed 5/1/19	Fri 5/3/19	71	Miguel Lopes
2.3.8.2 Criação da funcionalidade de mostragem dos res	3 days	Fri 5/3/19	Wed 5/8/19	73	Miguel Lopes
2.3.9 Criação da página Organizar Evento	**4 days**	**Fri 5/3/19**	**Thu 5/9/19**	**62**	
2.3.9.1 Criação dos campos com informação sobre o eve	2 days	Fri 5/3/19	Tue 5/7/19	62	Margarida Duarte
2.3.9.2 Conclusão do restante design	2 days	Tue 5/7/19	Thu 5/9/19	76	Margarida Duarte
2.3.10 Criação da página de Meus eventos	**7 days**	**Thu 5/2/19**	**Mon 5/13/19**	**68**	
2.3.10.1 Criação da funcionalidade de mostragem do his	3 days	Thu 5/2/19	Tue 5/7/19	68	Gonçalo Neto
2.3.10.2 Criação da funcionalidade de detalhes das tours	2 days	Tue 5/7/19	Thu 5/9/19	79	Gonçalo Neto
2.3.10.3 Conclusão do restante design	2 days	Thu 5/9/19	Mon 5/13/19	80	Gonçalo Neto
2.3.11 Criação da página Alterar password/email	**5 days**	**Wed 5/8/19**	**Wed 5/15/19**	**74**	
2.3.11.1 Criação dos campos e da funcionalidade para al	2 days	Wed 5/8/19	Fri 5/10/19	74	Miguel Lopes
2.3.11.2 Criação da funcionalidade de verificação e alter	2 days	Fri 5/10/19	Tue 5/14/19	83	Miguel Lopes
2.3.11.3 Conclusão do restante design	1 day	Tue 5/14/19	Wed 5/15/19	84	Miguel Lopes
2.3.12 Criação da página de FAQs	**3 days**	**Mon 5/13/19**	**Thu 5/16/19**	**81**	
2.3.12.1 Criação da coletânea das perguntas mais freque	1 day	Mon 5/13/19	Tue 5/14/19	81	Gonçalo Neto
2.3.12.2 Conclusão do restante design	2 days	Tue 5/14/19	Thu 5/16/19	87	Gonçalo Neto
2.3.13 Criação do Portal de Ajuda	**4 days**	**Wed 5/15/19**	**Tue 5/21/19**	**85**	
2.3.13.1 Criação da funcionalidade de comunicação com	2 days	Wed 5/15/19	Fri 5/17/19	85	Miguel Lopes
2.3.13.2 Conclusão do restante design	2 days	Fri 5/17/19	Tue 5/21/19	90	Miguel Lopes
2.3.14 Criação da página dos T&C	**2 days**	**Thu 5/9/19**	**Mon 5/13/19**	**77**	
2.3.14.1 Criação do texto dos T&C	1 day	Thu 5/9/19	Fri 5/10/19	77	Margarida Duarte
2.3.14.2 Conclusão do restante design	1 day	Fri 5/10/19	Mon 5/13/19	93	Margarida Duarte
2.3.15 Criação da funcionalidade de logout	**2 days**	**Tue 5/21/19**	**Thu 5/23/19**	**91**	
2.3.15.1 Criação da funcionalidade de retorno à página c	2 days	Tue 5/21/19	Thu 5/23/19	91	Miguel Lopes
2.4 Entregável - Caderno de Testes	0 days	Thu 5/23/19	Thu 5/23/19	96	

Figura 4 Desenvolvimento de Software III

8.1.3 Monitorização e Controlo

Na fase de Monitorização e Controlo, só não procedemos à criação do manual de administrador visto que, a sua atuação é diretamente no Firebase.

3 Monitorização e Controlo	**5 days**	**Thu 5/23/19**	**Thu 5/30/19**	**97**	
3.1 Entregável - Ponto de Situação	0 days	Thu 5/23/19	Thu 5/23/19	97	
3.2 Testes Funcionais	2 days	Thu 5/23/19	Mon 5/27/19	97	Gonçalo Neto,Margarida Duarte,Miguel Lopes
3.3 Testes de Integração	2 days	Mon 5/27/19	Wed 5/29/19	100	Gonçalo Neto,Margarida Duarte,Miguel Lopes
3.4 Testes de Aceitação	1 day	Wed 5/29/19	Thu 5/30/19	101	Gonçalo Neto,Margarida Duarte,Miguel Lopes
3.5 Entregável - Resultados dos Testes	0 days	Thu 5/30/19	Thu 5/30/19	102	
4 Criação dos manuais	**4 days**	**Thu 5/30/19**	**Wed 6/5/19**	**103**	
4.1 Finalização do manual do programador	4 days	Thu 5/30/19	Wed 6/5/19	103	
4.2 Finalização do manual do guia	4 days	Thu 5/30/19	Wed 6/5/19	103	
4.3 Finalização do manual do administrador	4 days	Thu 5/30/19	Wed 6/5/19	103	
4.4 Finalização do manual de utilização	4 days	Thu 5/30/19	Wed 6/5/19	103	Gonçalo Neto,Margarida Duarte,Miguel Lopes
4.5 Entregável - Manuais	0 days	Thu 5/30/19	Thu 5/30/19	103	
5 Implementação	**0 days**	**Wed 6/5/19**	**Wed 6/5/19**	**108**	
5.1 Publicação da aplicação	0 days	Wed 6/5/19	Wed 6/5/19	108	
6 Encerramento	**4 days**	**Wed 6/5/19**	**Tue 6/11/19**	**111**	
6.1 Finalização do relatório	4 days	Wed 6/5/19	Tue 6/11/19	111	Gonçalo Neto,Margarida Duarte,Miguel Lopes
6.2 Entregável - Relatório	0 days	Tue 6/11/19	Tue 6/11/19	113	

Figura 5 Monitorização e Controlo

8.2 Arquitetura

Em relação à arquitetura a nossa é constituída por:

Aplicação

Ponto de contacto entre utilizador e guia, como também ferramenta de pesquisa e obtenção de informação para ambos. É desenvolvida em React Native e oferece uma interface simples e minimalista para que até o menos entendido em smartphones a possa utilizar antes, durante e depois das suas viagens.

Foram usadas várias bibliotecas no desenvolvimento da aplicação em React Native tanto para executar funções críticas no desempenho e funcionalidades da aplicação como também na ligação ao Firebase da Google, as mais importantes são: redux, redux-thunk, rn-fetch-blob, react-native-woodpicker e react-native-firebase.

Base de Dados

A nossa base de dados é o ponto de comunicação crítico para as atividades e funcionalidades oferecidas pela nossa aplicação, quer seja a guardar documentos,

consultar restaurantes, ou adicionar tours à lista, todas essas interações passam pela comunicação com a base de dados.

Ambas comunicam uma com a outra para proporcionar aos utilizadores uma experiência única, personalizada e livre sem qualquer custo ou preocupação.

8.3 Tecnologias Usadas

Ao desenvolver esta aplicação foram usadas várias tecnologias:

React Native

A linguagem de programação usada para desenvolver a aplicação, criada pelo Facebook e baseada em React (web) e JavaScript, tem os seus próprios componentes nativos através do uso de JavaScript em vez de usar HTML, é uma linguagem relativamente nova (2015) mas demonstra ter grande potencial para desenvolver cada vez mais e melhores aplicações.

Firebase

Firebase é uma base de dados criada pela Google, do tipo NoSQL, não trata os dados como linhas em tabelas mas sim como objetos com propriedades que podem ser acedidos através de pedidos REST para a API.

REST

Arquitetura de software que permite a aplicação comunicar com as mais diversas APIs (Firebase, Zomato, entre outras) e obter / fornecer informações pela internet sob o formato de dados em JSON.

JSON

Usado juntamente com REST é um formato de troca de dados entre sistemas independente de qualquer linguagem de programação.

Zomato API

Na nossa aplicação usamos a API do Zomato para recolher informações sobre restaurantes perto da área onde o utilizador se encontra, ou da cidade onde planeia visitar, permite ao utilizador ver a classificação, localização e contacto dos restaurantes como também procurar pelo tipo de cozinha que prefere.

Google Maps

Para ajudar os utilizadores com direções e facilitar o trabalho dos guias, utilizamos uma ligação direta com a aplicação Google Maps em que o utilizador/guia pode inserir o ponto de origem e de destino ou até mesmo usar a sua localização dentro da nossa aplicação, que recebe esses dados e os encaminha para a aplicação do Google Maps.

8.4. Utilizadores

A aplicação é destinada a todos os utilizadores que tenham mais de 18 anos. Inicialmente no ato de inscrição todos os utilizadores por defeito são do tipo cliente. Sendo que, um utilizador inscreve-se na página inicial da aplicação preenchendo um breve formulário.

8.4.1 Clientes

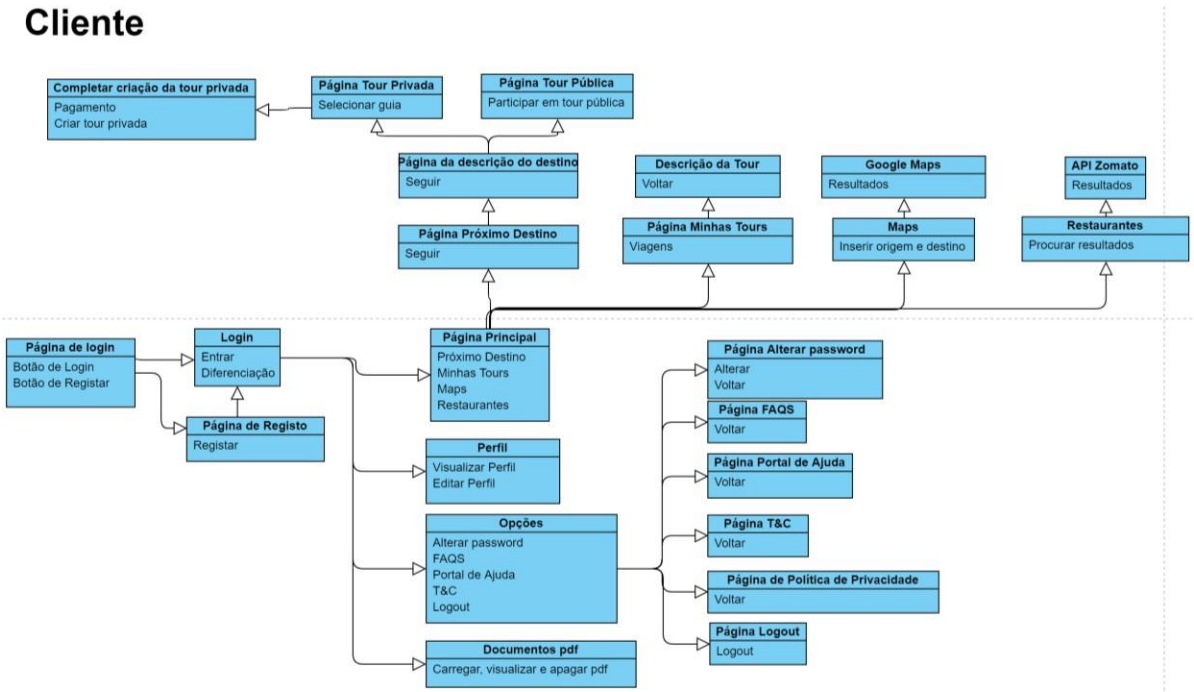


Figura 6 UML Cliente

Todos os clientes têm um perfil com informações públicas de nome, género, idioma e fotografia de perfil.

Os clientes podem escolher se querem fazer uma Tour Pública ou uma Tour Privada.

As Tours Públicas são eventos que foram previamente criados pelos guias para uma determinada data e horário.

Enquanto que uma Tour Privada é o cliente quem cria um evento com informações da tour incluindo uma descrição da tour que pretendem sendo que, esta opção de tour é paga.

8.4.2 Guias

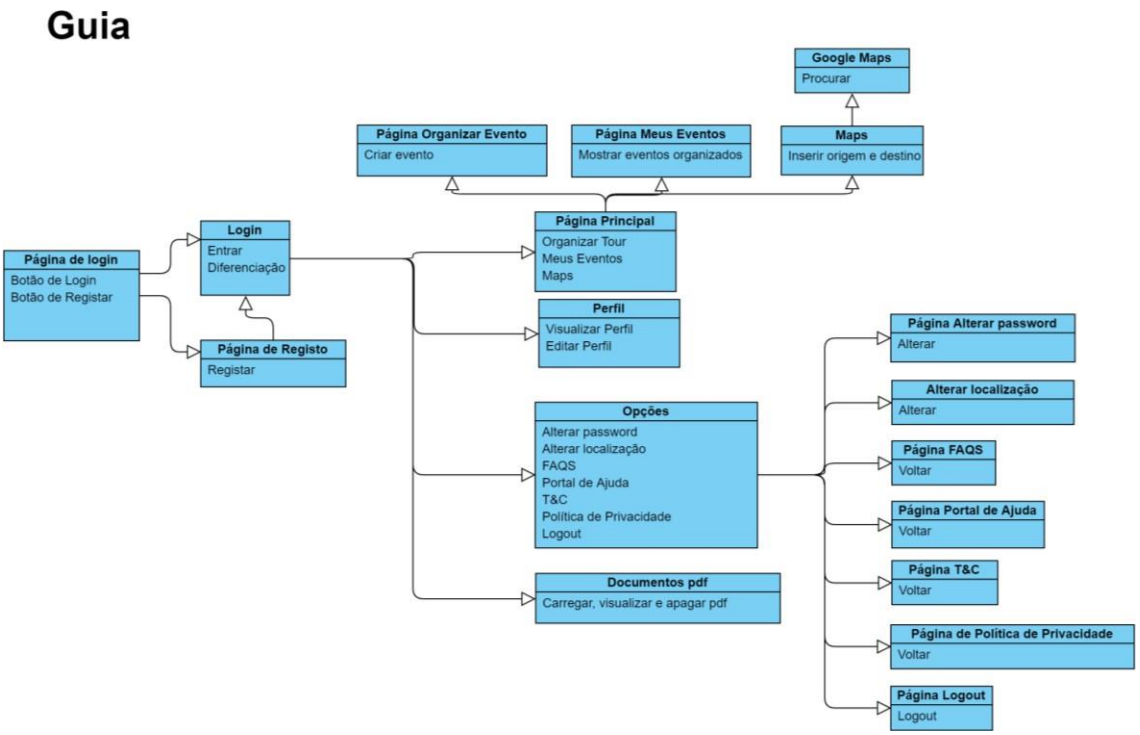


Figura 7 UML Guia

Para ser guia é necessário ter uma conta criada como utilizador, enviar uma mensagem para o nosso e-mail explicando a razão de querer ser guia, se for aceite o seu tipo de utilizador será alterado para guia.

O guia faz:

Tours Públicas que estão disponíveis para vários utilizadores que se encontrem no destino selecionado naquele período de tempo (versão grátis);

Tourss Privadas apenas para o utilizador e acompanhantes se existirem (versão paga) em que o utilizador tem a possibilidade de fazer uma descrição que inclua os sítios que gostaria de visitar.

O utilizador tem acesso ao perfil do guia o qual é composto pelas seguintes informações: Foto de Perfil, Nome, Idioma, Género.

8.5 Definições Técnicas

React Native é uma framework de aplicação móvel criada pelo Facebook que permite o desenvolvimento de aplicações nativas android, iOS e UWP [14].

A grande vantagem do React Native é que permite "cross-platform", ou seja, permite compilar o código para ambos os sistemas operativos mantendo o desempenho e aparência de aplicações nativas construídas em java ou objective-c [15].

A popularidade desta framework tem vindo a aumentar, tendo resultado numa grande comunidade de desenvolvedores [16].

É importante saber distinguir ReactJS de React Native. Enquanto que ambos são open source, o primeiro é uma biblioteca de javascript que suporta front-end web e pode ser corrido num servidor, de modo a contruir interfaces e aplicações web. O segundo é uma framework especificamente para mobile que utiliza APIs nativas para renderizar componentes em dispositivos móveis [17].

8.5.1 JSX

JSX é uma extensão da sintaxe da linguagem JavaScript que fornece uma maneira de estruturar a renderização dos componentes, que por sua vez são geralmente escritos em JSX [14].

Por exemplo, usa uma sintaxe do tipo <view><Text> Olá Mundo! </Text></View> para de seguida incorporar XML em Javascript.

É semelhante a HTML na web, no entanto em vez de ou usa componentes react.

8.5.2 Métodos Lifecycle

componentDidMount – É chamado quando o componente é "montado", ou seja, quando o componente é criado na interface do utilizador [14].

componentWillUnmount- É chamado imediatamente antes do componente ser destruido ou "desmontado" [14].

render – Único que é realmente necessário em qualquer componente. Geralmente é chamado sempre que o estado do componente seja atualizado, refletindo alterações na interface do utilizador [14].

8.5.3 Componentes

Basicamente tudo o que se vê no código é um componente, apenas é necessário que exista uma função "render" de fora a retornar JXS para renderizar [18].

Existem dois tipos de dados que controlam um componente: props e state [18].

Props

São parâmetros existentes na criação de um componente e são também definidos pelo hereditário durante todo o tempo de vida de um componente [18].

State

Trata-se do estado da aplicação e são usados em dados que eventualmente irão ser alterados [18].

Style

O estilo é semelhante ao CSS StyleSheet, mas em vez de criar um novo objeto todas as vezes, o StyleSheet permite criar objetos de estilo com um identificador de forma a poder referenciar, assim não tem de renderizá-lo novamente [18].

Desta forma, conseguimos ter um estilo base e reutilizá-lo sempre que necessário.

8.6 Bibliotecas utilizadas

Redux [19]

Trata-se de uma biblioteca que ajuda a gerir o estado da aplicação. É constituído pelos seguintes conceitos:

1. **Reducers**

Função que leva o estado anterior e uma ação como argumentos e retorna um novo estado. Estas ações são um objeto com um tipo um payload opcional.

Reducers especificam como o estado da aplicação é alterado em resposta a ações que são "despachadas" para o armazenamento. Ou seja, atualizam o estado de acordo com as ações.

2. **Actions**

São objetos javascript simples que representam payloads de informação (payloads são propriedades que contêm dados da ação de um objeto), que enviam dados da aplicação para a "storage".

Um exemplo da sua utilização é a ocorrência de um evento que é ativado por um clique de um botão ou seleção de um item. As ações são enviadas a partir de "Action Creators".

3. **Action Creators**

Função que retorna uma retorna uma ação, ou seja, retorna um objeto "action" que por sua vez é enviado a todos os diferentes "reducers" existentes na aplicação.

4. **Storage**

Objeto que junta as "actions" e os "reducers". Tem como objetivo:

- ✓ Permitir o acesso ao estado com getState();
- ✓ Permitir que o estado seja atualizado com dispatch(action);
- ✓ Manter todo o estado da aplicação.

9. Conclusões

Ao longo deste projeto conseguimos concretizar dentro do prazo que estabelecemos as tarefas maioritárias. Neste sentido, conseguimos desenvolver uma aplicação que seja útil durante o planeamento de uma viagem e também durante a mesma, integrando diversas funcionalidades.

As tarefas que foram mais difíceis para nós de realizar foram: criar a página de próximo destino pois tivemos problemas em concretizar a separação dos guias disponíveis na secção de tours privadas, criar a página do perfil em alterar as informações e adicionar uma fotografia na página de perfil, e na página de pdf a converter o pdf para base64. Apesar das dificuldades encontradas conseguimos ultrapassá-las.

Adicionalmente, encontramos algumas limitações devido à desatualização da documentação do Firebase relativamente à alteração da password e do e-mail.

Apesar de todas as dificuldades e limitações ao longo do nosso percurso conseguimos concretizar as tarefas que nos propusemos a fazer.

Na utilização do Firebase como pontos positivos destacamos a ligação acessível do React Native ao Firebase bem como, a facilidade em aceder aos dados visto que é uma base de dados não SQL.

Em conclusão, este trabalho foi muito gratificante para nós pois conseguimos aprender uma nova linguagem de programação e elaborar uma aplicação por inteiro com diversas funcionalidades que consideramos importantes e úteis para qualquer pessoa que queira viajar.

10. Bibliografia

[1] Wang, D., Xiang, Z. (2012). *The New Landscape of Travel: A Comprehensive Analyses of Smartphone Apps.* Disponível em: https://www.researchgate.net/profile/Zheng_Xiang5/publication/289511910_The_New_Landscape_of_Travel_A_Comprehensive_Analysis_of_Smartphone_Apps/links/589c6da6aca272e6cd45b62d/The-New-Landscape-of-Travel-A-Comprehensive-Analysis-of-Smartphone-Apps.pdf

[2] Lu, J., Mao, Z., Wang, M., Hu, L. (2015). *Goodbye maps, hello apps? Exploring the influential determinants of travel app adoption.* Disponível em: https://www.researchgate.net/profile/Zhenxing_Mao2/publication/277970481_Goodbye_Maps_Hello_Apps_Exploring_the_Influential_Determinants_of_Travel_App_Adoption/links/5a696222aca2728d0f5e1ba4/Goodbye-Maps-Hello-Apps-Exploring-the-Influential-Determinants-of-Travel-App-Adoption.pdf

[3] Dickinson, J.E., Ghali, K., Cherrett, T., Speed, C., Davies, N., Norgate, S. (2014). *Tourism and the smartphone app: capabilities, emerging practice and scope in the traveldomain.* Disponível em: http://eprints.bournemouth.ac.uk/21155/1/Dickinson%20et%20al%202013%20Current%20Issues.pdf

[4] TripAdvisor (sd). Acerca do TripAdvisor. Consultado em Junho 27, 2019 em: https://tripadvisor.mediaroom.com/pt-about-us

[5] Soral, R. (2019). *Five Android and iOS UI Design Guidelines for React Native.* Disponível em: https://www.infoq.com/articles/ios-android-react-native-design-patterns/

[6] Redação, D. (2019). *Por que React Native é a linguagem adequada para criação de apps?* Disponível em: https://computerworld.com.br/2019/02/12/por-que-react-native-e-a-linguagem-adequada-para-criacao-de-apps/

[7] Writer, M., S. (2019). *Native Mobile Apps Market Expanding Massively by 2020*. Disponível em: https://martechseries.com/mobile/mobile-marketing/native-mobile-apps-market-expanding-massively-2020/

[8] Srivastav, S. (2019). *Less Talked About React Native App Development Considerations.* Disponível em: https://appinventiv.com/blog/factors-to-consider-react-native-app-development/

[9] TripAdvisor (sd). Acerca do TripAdvisor. Consultado em Julho 1, 2019 em: https://tripadvisor.mediaroom.com/in-about-us

[10] Rezgo Support (sd). *What is Expedia?*. Consultado em Julho 1, 2019 em: https://www.rezgo.com/glossary/expedia

[11] Guy, P. (2014). *My Experience Using TripIt Pro to Organize My Travel*. Disponível em: https://thepointsguy.com/2014/04/my-experience-using-tripit-pro-to-organize-my-travel/

[12] Zaino, L. (2019). *The Best Apps for Travel*. Disponível em: https://thepointsguy.com/guides/best-travel-apps/

[13] Klook Travel Technology Ltd (sd). Klook: Travel Activities, Day Trips & Guided Tours. Consultado em Julho 1, 2019 em: https://play.google.com/store/apps/details?id=com.klook&hl=en

[14] Wikipedia (sd). Acerca do React Native. Consultado em Junho 29, 2019 em: https://en.wikipedia.org/wiki/React_(JavaScript_library)

[15] oreilly (sd). Acerca do React Native. Consultado em Junho 29, 2019 em: https://www.oreilly.com/library/view/learning-react-native/9781491929049/ch01.html

[16] Thinkwik (2018). *React Native: What is it? and, Why is it used?* Disponível em:https://medium.com/@thinkwik/react-native-what-is-it-and-why-is-it-used-b132c3581df

[17] Ashwini, A. (2017). *WHAT IS THE DIFFERENCE BETWEEN REACT.JS AND REACT NATIVE?* Disponível em: https://www.cognitiveclouds.com/insights/what-is-the-difference-between-react-js-and-react-native/

[18] Facebook Inc. (sd). Acerca do React Native. Consultado em Junho 29, 2019 em: https://facebook.github.io/react-native/docs/0.59/tutorial

[19] Byers, D (sd). *An Introduction to Redux's Core Concepts.* Disponível em: https://alligator.io/redux/redux-intro/

11. Anexos

11.1 Manual de Instalação

11.1.1 Utilizador

O utilizador, primeiro precisa de descarregar a nossa APK:

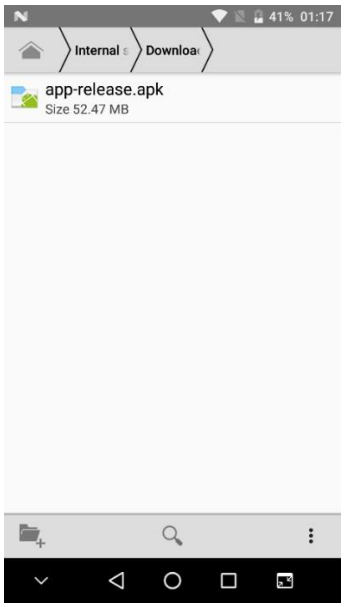


Figura 8 APK

De seguida, apenas precisa de tocar na APK e será prontamente levado à interface de instalação:

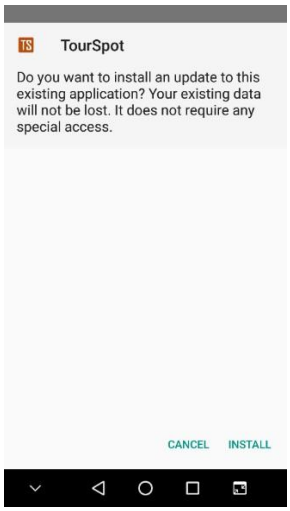


Figura 9 Interface de Instalação

O utilizador verá que rapidamente a aplicação será instalada no seu dispositivo:

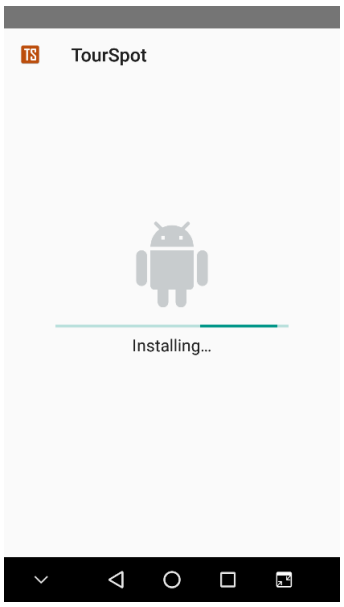


Figura 10 Instalação da aplicação

Depois de feita a instalação, já poderá aceder a todas as suas funcionalidades depois de entrar:

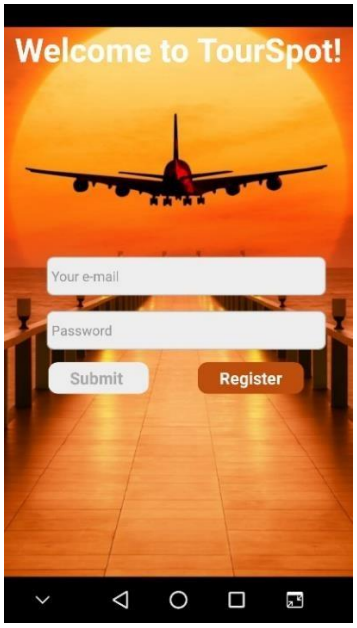


Figura 11 Aplicação TourSpot

11.1.2 Programador

O programador necessitará de certos requisitos para poder instalar a aplicação em opção de desenvolvimento. Os passos são os seguintes:

1. Instalar o **npm (NodeJS)** que é um dos elementos principais no desenvolvimento da aplicação React Native em:
   https://nodejs.org/en/

2. Instalar o git em:
   https://git-scm.com/

3. Instalar o pacote npm **react-native-cli** para poder executar operações relacionadas com o React Native:

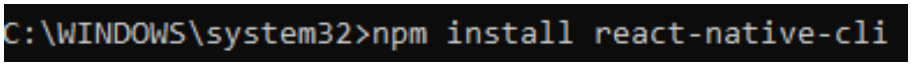


Figura 12 Instalação react-native-cli

4. Ter a versão 1.8 do Java instalada encontrada em:
   https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

5. Um IDE para compilar o código, neste caso recomendamos fortemente o Visual Studio Code em:
   https://code.visualstudio.com/

6. Alterar as variáveis de ambiente do seu computador para incluir as seguintes:

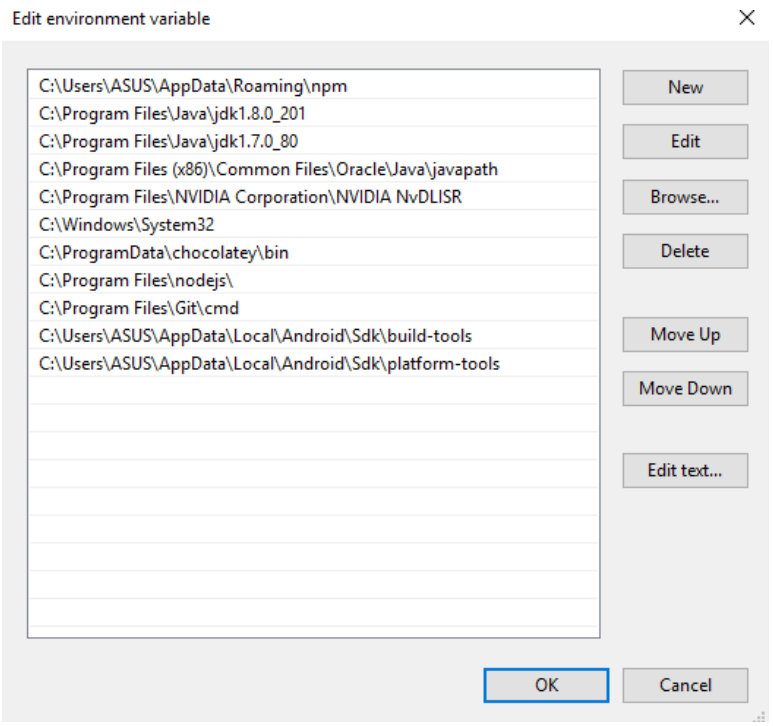


Figura 13 Variáveis de ambiente

7. Com isto feito poderá descarregar a pasta com o projeto inteiro e passar aos passos de instalação de pacotes.

Os pacotes a ser instalados são os seguintes:

moment: Necessário para a formatação de datas na aplicação.

react-native-modal: Necessário para a criação do modal quando se seleciona um objeto para obter informação mais detalhada.

react-native-firebase: Necessário para a ligação entre aplicação e Firebase (base de dados).

react-native-geocoding: Necessário para obter a localização atual do utilizador.

react-native-document-picker: Necessário para carregar ficheiros PDF (documentos) que podem ser lidos na aplicação.

react-native-image-picker: Necessário para carregar imagens para o perfil dos utilizadores.

react-native-launch-navigator: Necessário para abrir a aplicação do Google Maps com as informações geradas na nossa aplicação.

react-native-navigation: Necessário para poder navegar entre ecrãs e tabs.

react-native-vector-icons: Necessário para exibir ícones dentro da aplicação, o tema usado são os Ionicons.

react-native-view-pdf: Necessário para ler ficheiros PDF vindos do Firebase.

react-native-woodpicker: Necessário para escolher datas dentro da aplicação.

react-redux: Necessário para utilizarmos a abordagem Redux no nosso projeto.

redux-thunk: Necessário para utilizarmos o redux com HTTP requests (pedidos ao Firebase com a API REST).

rn-fetch-blob: Necessário para enviar e receber informações em base64 relativas aos documentos (PDFS)

Com estes pacotes instalados o programador está cada vez mais perto de poder desenvolver a sua própria versão da nossa aplicação.

8. Apenas precisa de ligar o seu dispositivo em modo de programador ao seu computador, abrir o VS Code e abrir uma instância do terminal, escrevendo o seguinte:

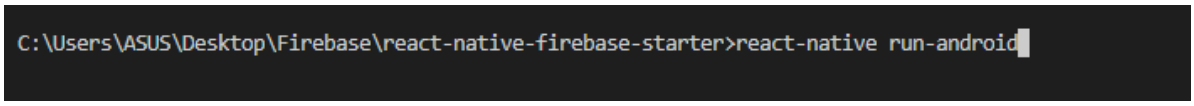


Figura 14 Execução do react-native run-android

9. Com isto, aguarda-se o tempo necessário para o packager aparecer e quando o comando estiver concluído poderá assim começar a desenvolver a sua aplicação.

11.2 Manual do Programador

Após instalar corretamente a nossa aplicação, as funções seguintes são as mais importantes no nosso código:

updateInputState()

Também muito usada em campos que precisam de validação, quer seja pelo seu formato ou pela falta de conteúdo, usamos bastante esta função quando precisamos de guardar/retirar informação do Firebase.

Um exemplo:

```
//verificar se os campos estão válidos
updateInputState = (key,value) => {
    let connectedValue = {}
    if(this.state.controls[key].validationRules.equalTo){
        const equalControl = this.state.controls[key].validationRules.equalTo
        const equalValue = this.state.controls[equalControl].value
        connectedValue = {
            ...connectedValue,
            equalTo: equalValue
        }
    }
    if(key === 'password'){
        connectedValue = {
            ...connectedValue,
            equalTo: value
        }
    }
    this.setState(prevState => {
        return{
            controls:{
                ...prevState.controls,
                confirmPassword:{
                    ...prevState.controls.confirmPassword,
                    valid: key === 'password' ? validate(prevState.controls.confirmPassword.value,
                        prevState.controls.confirmPassword.validationRules,
                        connectedValue) : prevState.controls.confirmPassword.valid
                },
                [key]:{
                    ...prevState.controls[key],
                    value: value,
                    valid: validate(value, prevState.controls[key].validationRules, connectedValue),
                    touched: true
                }
            }
        }
    })
}
```

Figura 15 Exemplo de validação de dados

Funções mais importantes por classe:

1. Auth:

authHandler

Vai processar a informação fornecida pelo utilizador e vai enviá-la através do redux para a função tryAuth (situada no auth.js na store do Redux) explicada de seguida.

```
//funçao para login / registo
authHandler = () => {

    //dados de login
    const authData = {
        email: this.state.controls.email.value,
        password: this.state.controls.password.value
    }
    if (this.state.authMode === 'login'){
        return this.props.onTryAuth(authData, this.state.authMode,null,true,this.state.controls.email.value)

    }
    else {
        console.log(this.state.date)
        let datee = Moment(this.state.date.toDateString()).format('DD/MM/YYYY')
        const regData = {
            email: this.state.controls.email.value,
            password: this.state.controls.password.value,
            name: this.state.controls.name.value,
            phone: this.state.controls.phone.value,
            gender: this.state.gender.value,
            idioma: this.state.language.value,
            DoB: datee,
            type: 'cliente'
        }
        return this.props.onTryAuth(authData, this.state.authMode,regData,false,this.state.controls.email.value)

    }

}
```

Figura 16 Função authHandler

tryAuth(obj authData, String authMode, obj regData, boolean bool, String email)

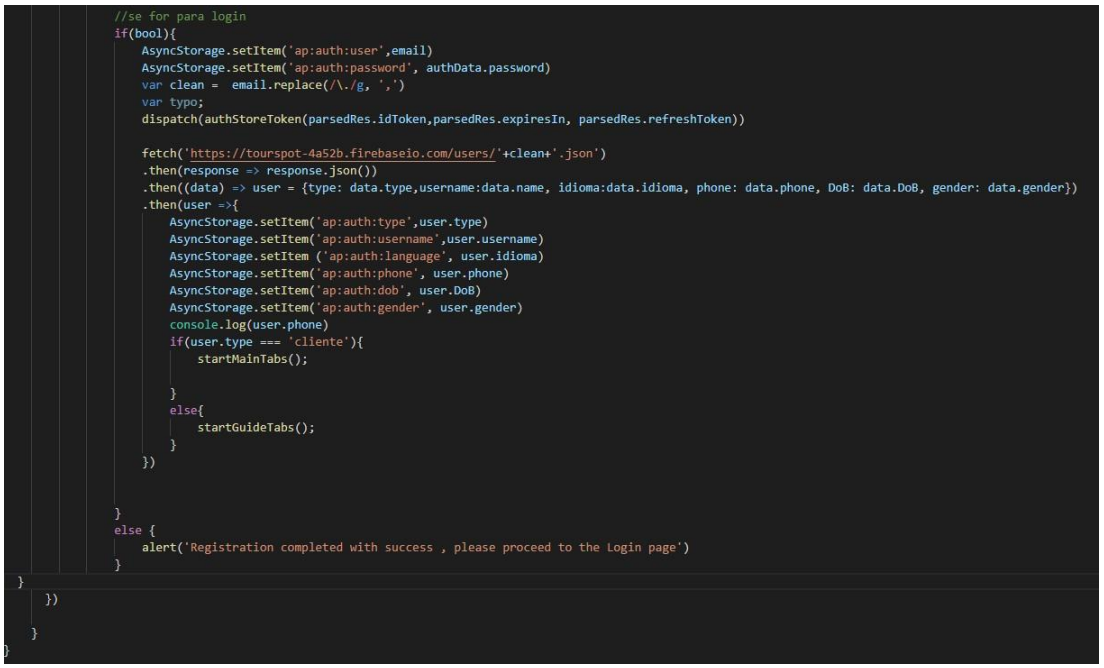
Usada para login / registo em que envia as informações para o Firebase e retorna a resposta enviada pela API REST do Firebase.

```
//ação de autenticação - login/registo
export const tryAuth = (authData, authMode, regData, bool, email) => {

    return (dispatch,getState) => {


        dispatch(uiStartLoading())
        let url = 'https://www.googleapis.com/identitytoolkit/v3/relyingparty/verifyPassword?key=AIzaSyDkoRVdOPjmx-Rxa9jwWWGYSYK7LLY7QP0'

        if( authMode === 'signup'){
            url = 'https://www.googleapis.com/identitytoolkit/v3/relyingparty/signupNewUser?key=AIzaSyDkoRVdOPjmx-Rxa9jwWWGYSYK7LLY7QP0'
        }
        //entrar / registar no firebase
        fetch(url,
        {method:'POST',
        body: JSON.stringify({
            email: authData.email,
            password: authData.password,
            returnSecureToken: true
        }),
        headers:{
            "Content-Type": "application/json"
        }
        })
        .catch(err => {
            console.log(err);
            alert(err)
            dispatch(uiStopLoading())
        })
        .then(res => res.json())
        .then(parsedRes =>{
            dispatch(uiStopLoading())
```

Figura 17 Função tryAuth I

```
        if (parsedRes.error){
            console.log(parsedRes)
            console.log(parsedRes.error)
            //alert('Authentication failed, try again')
             switch(parsedRes.error.message){
                 case 'EMAIL_EXISTS':
                     alert('This user already exists')
                     break;
                 case 'EMAIL_NOT_FOUND':
                     alert('This user doesn\'t exist')
                     break;
                 case 'INVALID_PASSWORD':
                     alert('The password is incorret, please try again')
                     break;
             }
        }
        else {
            console.log(parsedRes);
            const token = getState().auth.token
            //se existir dados de registo guarda os dados no database do firebase
            if (regData){
                const data = regData
                var cleanEmail = email.replace(/\./g, ',')
                fetch('https://tourspot-4a52b.firebaseio.com/users/'+cleanEmail+'.json', {
                    headers: { "Content-Type": "application/json; charset=utf-8" },
                    method: 'PATCH',
                    body: JSON.stringify(data)
                    })


            }
```

Figura 18 Função tryAuth II



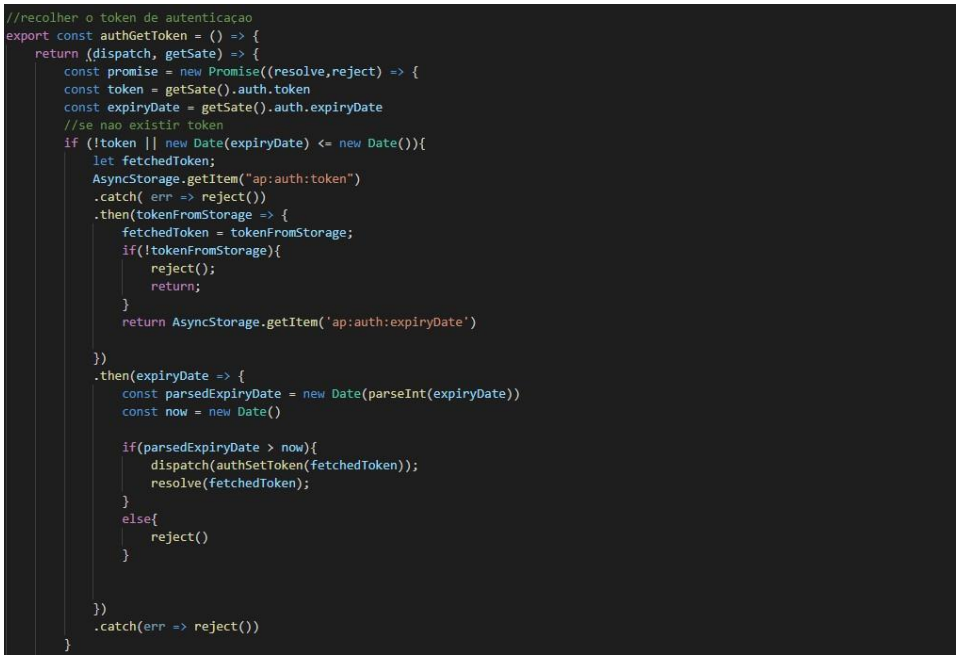
```
            //se for para login
            if(bool){
                AsyncStorage.setItem('ap:auth:user',email)
                AsyncStorage.setItem('ap:auth:password', authData.password)
                var clean =  email.replace(/\./g, ',')
                var typo;
                dispatch(authStoreToken(parsedRes.idToken,parsedRes.expiresIn, parsedRes.refreshToken))

                fetch('https://tourspot-4a52b.firebaseio.com/users/'+clean+'.json')
                .then(response => response.json())
                .then((data) => user = {type: data.type,username:data.name, idioma:data.idioma, phone: data.phone, DoB: data.DoB, gender: data.gender})
                .then(user =>{
                    AsyncStorage.setItem('ap:auth:type',user.type)
                    AsyncStorage.setItem('ap:auth:username',user.username)
                    AsyncStorage.setItem ('ap:auth:language', user.idioma)
                    AsyncStorage.setItem('ap:auth:phone', user.phone)
                    AsyncStorage.setItem('ap:auth:dob', user.DoB)
                    AsyncStorage.setItem('ap:auth:gender', user.gender)
                    console.log(user.phone)
                    if(user.type === 'cliente'){
                        startMainTabs();

                    }
                    else{
                        startGuideTabs();
                    }
                })


            }
            else {
                alert('Registration completed with success , please proceed to the Login page')
            }
  }
    })

  }
}
```

Figura 19 Função tryAuth III

Na mesma classe do tryAuth temos dois métodos bastante importantes:

authGetToken()

Serve para retornar o token de autenticação que vai servir para manter a sessão mesmo quando se desliga a aplicação.



```
//recolher o token de autenticaçao
export const authGetToken = () => {
    return (dispatch, getSate) => {
        const promise = new Promise((resolve,reject) => {
        const token = getSate().auth.token
        const expiryDate = getSate().auth.expiryDate
        //se nao existir token
        if (!token || new Date(expiryDate) <= new Date()){
            let fetchedToken;
            AsyncStorage.getItem("ap:auth:token")
            .catch( err => reject())
            .then(tokenFromStorage => {
                fetchedToken = tokenFromStorage;
                if(!tokenFromStorage){
                    reject();
                    return;
                }
                return AsyncStorage.getItem('ap:auth:expiryDate')

            })
            .then(expiryDate => {
                const parsedExpiryDate = new Date(parseInt(expiryDate))
                const now = new Date()

                if(parsedExpiryDate > now){
                    dispatch(authSetToken(fetchedToken));
                    resolve(fetchedToken);
                }
                else{
                    reject()
                }


            })
            .catch(err => reject())
        }
```

Figura 20 Função authGetToken() I

```
        else {
            resolve(token)
        }
        });
        return promise
        .catch( err => {
           return AsyncStorage.getItem('ap:auth:refreshToken')
            .then(refreshToken => {
                return fetch('https://securetoken.googleapis.com/v1/token?key=AIzaSyACT-liOWAzpa6hH8TgcrIvHM5OCFg9Qbs',{
                    method:'POST',
                    headers:{
                        'Content-Type': 'application/x-www-form-urlencoded'
                    },
                    body:"grant_type=refresh_token&refresh_token=" + refreshToken
                })
            })
            .then(res => res.json())
            .then(parsedRes => {
                if(parsedRes.id_token){
                    console.log('Refresh worked')
                    dispatch(authStoreToken(parsedRes.id_token,parsedRes.expires_in,parsedRes.refresh_token));
                    return parsedRes.id_token;
                }else{
                    dispatch(authClearStorage())
                }
            })
        })
        .then(token =>{
            if(!token){
                throw(new Error())
            }
            else{
                return token
            }

        })
    }
}
```

Figura 21 Função authGetToken() II

authAutoSignIn()

Serve para fazer o auto Sign in caso exista token válido.

```
//entrar automaticamente se ja estiver autenticado
export const authAutoSignIn = () =>{
    var typp
    AsyncStorage.getItem('ap:auth:type').then(type => typp = type)

    return dispatch => {
        dispatch(authGetToken())
      .then(token => {
        if(typp === 'cliente'){
            startMainTabs();
        }
        else{
            startGuideTabs();
        }

      })
      .catch(err => console.log("Failed to fetch token!"));


    }
}
```

Figura 22 Função authAutoSignIn()

2. **ChangeLocation**:

onSubmit()

Envia para o Firebase a informação relativa à localização atual do guia.

```
//função para submeter os dados no firebase
onSubmit = () =>{

    AsyncStorage.getItem('ap:auth:user')
    .then(user => {
        var clean =  user.replace(/\./g, ',')

        dat ={
            location: this.state.location
        }
        console.log(this.state.location)
        fetch('https://tourspot-4a52b.firebaseio.com/users/'+clean+'.json', {
                        headers: { "Content-Type": "application/json; charset=utf-8" },
                        method: 'PATCH',
                        body: JSON.stringify(dat)
                        })
    })

    alert('Location changed with success!!!')

}
```

Figura 23 Função onSubmit()

3. ChangePass:

changePass()

Envia as informações relativas à palavra passe do utilizador para o Firebase de modo a alterar a palavra passe.

```
changePass = () => {
    console.log(this.state.changed)
        AsyncStorage.getItem('ap:auth:user')
        .then(email => {
            this.setState({
                email: email
            })
        })
        .then(() => {
            AsyncStorage.getItem('ap:auth:password')
        .then(pass => {
            this.setState({
                password: pass
            })
        }).then(() => {

            console.log(this.state.email + '    EMAIL')
    console.log(this.state.password + '    PASSWORD')
    firebase.auth().signInWithEmailAndPassword(this.state.email, this.state.password)
    .then(user =>{
        console.log(user)
       firebase.auth().currentUser.updatePassword(this.state.controls.password.value)
        AsyncStorage.setItem('ap:auth:password', this.state.controls.password.value)
        this.setState({changed:true})
    })
    .then(() => {
        var clean =  this.state.email.replace(/\./g, ',')

        dat ={
            password: this.state.controls.password.value
        }
        console.log(this.state.password)
        fetch('https://tourspot-4a52b.firebaseio.com/users/'+clean+'.json', {
                        headers: { "Content-Type": "application/json; charset=utf-8" },
                        method: 'PATCH',
                        body: JSON.stringify(dat)
                        })
    })
    .then(() => {
        alert('Password changed with success!!!')
    })
    .catch(err => console.log(err))

        })
        })
}
```

Figura 24 Função changePass()

4. DocumentsDisplay:

pdf()

Abre o document picker para selecionar o pdf pretendido.

```
//exibir janela de upload de pdf
pdf = () =>{
    this.setState({didIt:true})
    DocumentPicker.show({
        filetype: [DocumentPickerUtil.pdf()],
    }, (error, response) => {
        if(response){
            let path = (Platform.OS == 'ios')
            ? response.uri.replace('file://', '')
            : response.uri;
        RNFetchBlob.fs.readFile(path, 'base64')
       .then((encoded) => {
           this.setState({image: {uri:"data:application/pdf;base64,"+encoded}, b64:encoded})
           console.log(encoded)
       })
       .catch((error) => console.error(error));
        }

   });
}
```

Figura 25 Função pdf()

submit()

Envia o documento convertido em base64 para o Firebase.

```
//submeter os dados do pdf em base64 no firebase e atualizar a lista de pdfs
submit = () => {
    data = {
        base64: this.state.b64
    }
 fetch('https://tourspot-4a52b.firebaseio.com/users/'+this.state.user+'/documents/'+this.state.pdfname+'.json', {
     headers: { "Content-Type": "application/json; charset=utf-8" },
     method: 'PATCH',
     body: JSON.stringify(data)
     }).then(() => {
         let tempDocs = []
         for(let i in this.state.docs){


                 tempDocs.push(this.state.docs[i])

         }
         tempDocs.push({name: this.state.pdfname, base64:this.state.b64})
         this.setState({docs: tempDocs})
     })
     this.setState({ isModalVisible: !this.state.isModalVisible });
     this.setState({ addModal: !this.state.addModal });


}
```

Figura 26 Função submit()

componentDidMount()

Carrega os documentos do utilizador quando o componente inicia.

```
   //funçao quando o componente inicia e que vai buscar todos os documentos do dado utilizador ativo
componentDidMount(){
    AsyncStorage.getItem('ap:auth:user')
    .then(item =>{
        let clean =  item.replace(/\./g, ',')
        this.setState({user:clean})
        return clean
    }).then(clean =>{
        console.log(clean+ ' CLEAN')


        fetch('https://tourspot-4a52b.firebaseio.com/users/'+clean+'/documents.json')
        .catch(err => console.log(err))
        .then(res => res.json())
        .then(parsedRes => {
            let docs = []
            console.log(parsedRes)
            for(let ev in parsedRes){
                docs.push({
                    ...parsedRes[ev],
                    name: ev
                })

            }

                        return this.handleEv(docs)
        })


    })

}
```

Figura 27 Função componentDidMount()

toggleDelete()

Apaga o pdf selecionado.

```
//funçao para apagar um documento e atualizar a informaçao no firebase
toggleDelete = () => {

    console.log(this.state.pdf.name && !this.state.addModal  + '   OI')
    fetch('https://tourspot-4a52b.firebaseio.com/users/'+this.state.user+'/documents/'+this.state.pdf.name+'.json', {
    headers: { "Content-Type": "application/json; charset=utf-8" },
    method: 'DELETE'
    }).then(() => {
        let tempDocs = []
        for(let i in this.state.docs){
            if(this.state.docs[i].name === this.state.pdf.name){
                console.log('entrouuu')

            }
            else{
                tempDocs.push(this.state.docs[i])
            }
        }
        this.setState({docs: tempDocs})
    })

    this.setState({ isModalVisible: !this.state.isModalVisible });
}
```

Figura 28 Função toggleDelete()

5. MyEvents (GUIA):

componentDidMount()

Carrega os eventos de um determinado guia quando o componente inicia.

```
//função quando se inicia um componente e vai buscar a lista de todos os eventos
componentDidMount(){
    let events;
    fetch('https://tourspot-4a52b.firebaseio.com/events.json')
    .catch(err => console.log(err))
    .then(res => res.json())
    .then(parsedRes => {
        let event = []

        for(let ev in parsedRes){
            event.push({
                ...parsedRes[ev],
                name: ev
            })

        }
        console.log(event)
        events = event
                    return this.handleEv(event)
    })

    AsyncStorage.getItem('ap:auth:username')
    .then(item =>{
        this.setState({user:item})
    })


}
```

Figura 29 Função componentDidMount() do guia

6. NewTour (GUIA):

submitEvent()

Envia a informação relativa ao evento criado pelo guia no Firebase.

```
    //submeter tour no firebase
submitEvent = () =>{
    let name =  this.state.controls.name.value

    //definir a data do campo e transformar no formato de DD MM YYYY
    let datee = Moment(this.state.date.toDateString()).format('DD/MM/YYYY')

    //ir buscar o email ao asyncstorage
    AsyncStorage.getItem('ap:auth:user').then(user => {
        this.setState({user:user})
    })

    //guardar a informação do evento no firebase
    AsyncStorage.getItem('ap:auth:username').then(item => {

      return  data = {
       country: this.state.controls.country.value,
       city: this.state.controls.city.value,
       rendezvous: this.state.controls.rendezvous.value,
       description: this.state.controls.description.value,
       time: this.state.controls.time.value,
       type: 'public',
       date: datee,
       language:this.state.language,
       guide: item,
       guideID: this.state.user
    }}).then( dat => {
       console.log(dat)
       fetch('https://tourspot-4a52b.firebaseio.com/events/'+name+'.json', {
                  headers: { "Content-Type": "application/json; charset=utf-8" },
                  method: 'PATCH',
                  body: JSON.stringify(dat)
                  })
                }
    )
      alert('Tour created with success!!!')
}
```

Figura 30 Função submitEvent()

7. MapScreen:

getLocation()

Utiliza o GPS do dispositivo para fornecer coordenadas à aplicação.

```
//obter a localizaçao do dispositivo
getLocation = () => {
      var that =this;
      this.setState({touchedLoc: true, touchedStart: true})
//Checking for the permission just after component loaded
if(Platform.OS === 'ios'){
 this.callLocation(that);
}else{
 async function requestLocationPermission() {
  try {
   const granted = await PermissionsAndroid.request(
    PermissionsAndroid.PERMISSIONS.ACCESS_FINE_LOCATION,{
     'title': 'Location Access Required',
     'message': 'This App needs to Access your location'
    }
   )
   if (granted === PermissionsAndroid.RESULTS.GRANTED) {
    //To Check, If Permission is granted
    that.callLocation(that);
   } else {
    alert("Permission Denied");
   }
  } catch (err) {
   alert("err",err);
   console.warn(err)
  }
 }
 requestLocationPermission();
}



 }
```

Figura 31Função getLocation()

submitEvent()

Envia as informações de localização/origem e destino para a aplicação do Google Maps.

```
//submeter os dados para o google maps
submitEvent = () => {
 if(Platform.OS === "android") LaunchNavigator.setGoogleApiKey("AIzaSyCSw4H2lXmqzCSokVt0RWi63fpnOTqd9iA");

 if(this.state.touchedLoc){
  LaunchNavigator.navigate(this.state.destination, {
   start: this.state.currentLocation
 })
    .then(() => console.log("Launched navigator"))
    .catch((err) => console.error("Error launching navigator: "+err));
 }
 else{

  LaunchNavigator.navigate(this.state.destination, {
   start: this.state.start
 })
    .then(() => console.log("Launched navigator"))
    .catch((err) => console.error("Error launching navigator: "+err));

 }

 }
```

Figura 32 Função submitEvent()

8. NextDestination:

submitEvent()

Guarda as informações iniciais de uma tour no AsyncStorage de forma a poder usá-las na página seguinte e abre a página de tours privadas ou públicas consoante a escolha do utilizador.

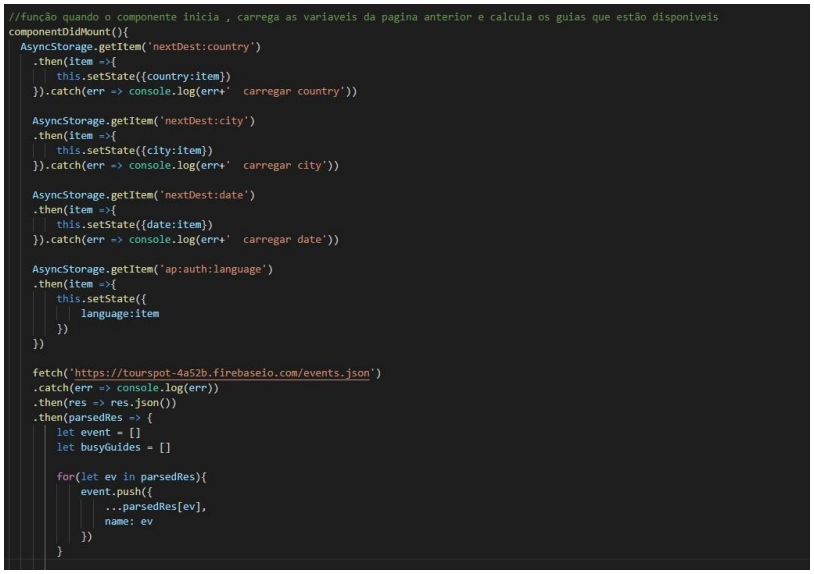
```
    //submeter os primeiros parametros do evento
submitEvent = () =>{
    let country =  this.state.country
    let city = this.state.city
    let datee = Moment(this.state.date.toDateString()).format('DD/MM/YYYY')
    let type = this.state.type

    AsyncStorage.setItem('nextDest:country',country)
    .catch(err => console.log('erro country'))
    AsyncStorage.setItem('nextDest:city',city)
    .catch(err => console.log('erro city'))
    AsyncStorage.setItem('nextDest:date',datee)
    .catch(err => console.log('erro date'))

    //abre a pagina consoante for public ou private
    if(this.state.type === 'public'){
        this.props.navigator.push({
            screen: 'awesome-places.NextDestinationPublic',
            title: 'Results'
        })
    }
    else{
        this.props.navigator.push({
            screen: 'awesome-places.NextDestinationPrivate',
            title: 'P'
        })
    }


}
```

Figura 33 Função submitEvent()

9. NextDestinationPrivate:

componentDidMount()

Exibe todos os guias disponíveis naquela data e local através da recolha de informações do Firebase com um pedido fetch (REST API), filtra os que correspondem aos parâmetros inicias e exibe-os numa lista.



```
//função quando o componente inicia , carrega as variaveis da pagina anterior e calcula os guias que estão disponiveis
componentDidMount(){
  AsyncStorage.getItem('nextDest:country')
    .then(item =>{
        this.setState({country:item})
    }).catch(err => console.log(err+'  carregar country'))

    AsyncStorage.getItem('nextDest:city')
    .then(item =>{
        this.setState({city:item})
    }).catch(err => console.log(err+'  carregar city'))

    AsyncStorage.getItem('nextDest:date')
    .then(item =>{
        this.setState({date:item})
    }).catch(err => console.log(err+'  carregar date'))

    AsyncStorage.getItem('ap:auth:language')
    .then(item =>{
        this.setState({
            language:item
        })
    })

    fetch('https://tourspot-4a52b.firebaseio.com/events.json')
    .catch(err => console.log(err))
    .then(res => res.json())
    .then(parsedRes => {
        let event = []
        let busyGuides = []

        for(let ev in parsedRes){
            event.push({
                ...parsedRes[ev],
                name: ev
            })
        }
```

Figura 34 Função componentDidMount() para exibir os guias disponíveis I

```
            for (let ev in event){
              if(event[ev].date === this.state.date){
                let guide = event[ev].guideID
                if(busyGuides.includes(event[ev].guideID)){

                }else{
                  busyGuides.push(guide)
                }
              }
            }
                                return busyGuides
        })
        .then(busyGuides =>{
          console.log(busyGuides)
          let guids = []
          fetch('https://tourspot-4a52b.firebaseio.com/users.json')
        .catch(err => console.log(err))
        .then(res => res.json())
        .then(parsedRes => {
          for(let ev in parsedRes){
            if(parsedRes[ev].type ==='guia'){
              guids.push({
                ...parsedRes[ev]
            })
            }
        }
        let availableGuides = []
        for(let ev in guids){
          if(!busyGuides.includes(guids[ev].email)){
            availableGuides.push(guids[ev])
          }
        }
          return this.handleEv(availableGuides)
        })

        })
        AsyncStorage.getItem('ap:auth:user')
        .then(item =>{
            this.setState({user:item})
        })
    }
```

Figura 35 Função componentDidMount() para exibir os guias disponíveis II

submitGuide()

Guarda as informações do guia escolhido no AsyncStorage.

```
  //submeter o resto das variaveis para a proxima pagina
submitGuide = () =>{

AsyncStorage.setItem('nextDest:guideName', this.state.guide.name)
AsyncStorage.setItem('nextDest:guideID', this.state.guide.email)

 alert('You have chosen '+this.state.guide.name+' as your guide!')
 this.setState({chosen: true})
}
```

Figura 36 Função submitGuide()

10. NextDestinationPrivate2:

componentDidMount()

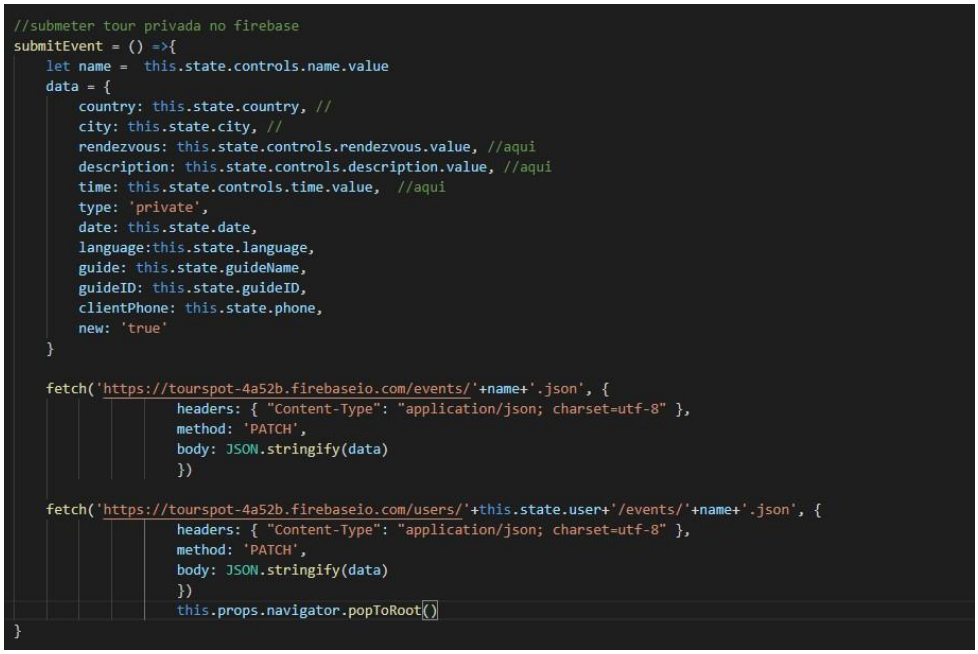
Carrega todas as variáveis anteriores necessárias para a criação de uma tour.

```
//função que corre quando o componente inicia carrega as variaveis necessarias para criar a tour privada
componentDidMount(){
    AsyncStorage.getItem('ap:auth:language').then(item =>  {
        console.log(item)
        this.setState({
            language: item
        })
    })
    AsyncStorage.getItem('nextDest:country').then(item =>  {
        console.log(item)
        this.setState({
           country: item
        })
    })
    AsyncStorage.getItem('nextDest:date').then(item =>  {
        console.log(item)
        this.setState({
            date: item
        })
    })
    AsyncStorage.getItem('nextDest:city').then(item =>  {
        console.log(item)
        this.setState({
            city: item
        })
    })
    AsyncStorage.getItem('nextDest:guideName').then(item =>  {
        console.log(item)
        this.setState({
            guideName: item
        })
    })
    AsyncStorage.getItem('nextDest:guideID').then(item =>  {
        console.log(item)
        this.setState({
            guideID: item
        })
    })
    AsyncStorage.getItem('ap:auth:user').then(user => {
        let clean =  user.replace(/\./g, ',')
        this.setState({user:clean})
    })
    AsyncStorage.getItem('ap:auth:phone').then(item => {
        console.log(item)
        this.setState({phone:item})
    })
  }
```

Figura 37 Função componentDidMount() para a criação de uma tour

submitEvent()

Envia para o firebase as informações e cria uma tour.



```
//submeter tour privada no firebase
submitEvent = () =>{
    let name =  this.state.controls.name.value
    data = {
        country: this.state.country, //
        city: this.state.city, //
        rendezvous: this.state.controls.rendezvous.value, //aqui
        description: this.state.controls.description.value, //aqui
        time: this.state.controls.time.value,  //aqui
        type: 'private',
        date: this.state.date,
        language:this.state.language,
        guide: this.state.guideName,
        guideID: this.state.guideID,
        clientPhone: this.state.phone,
        new: 'true'
    }

    fetch('https://tourspot-4a52b.firebaseio.com/events/'+name+'.json', {
                    headers: { "Content-Type": "application/json; charset=utf-8" },
                    method: 'PATCH',
                    body: JSON.stringify(data)
                    })

    fetch('https://tourspot-4a52b.firebaseio.com/users/'+this.state.user+'/events/'+name+'.json', {
                    headers: { "Content-Type": "application/json; charset=utf-8" },
                    method: 'PATCH',
                    body: JSON.stringify(data)
                    })
                    this.props.navigator.popToRoot()
}
```

Figura 38 Função submitEvent()

11. NextDestinationPublic

componentDidMount()

Carrega todos os eventos do Firebase que mais tarde vão ser filtrados pelo render para serem exibidos apenas os que corresponderem aos parâmetros de pesquisa do utilizador.

```
//função que corre quando  o componente inicia
componentDidMount(){
    fetch('https://tourspot-4a52b.firebaseio.com/events.json')
    .catch(err => console.log(err))
    .then(res => res.json())
    .then(parsedRes => {
        let event = []

        for(let ev in parsedRes){
            event.push({
                ...parsedRes[ev],
                name: ev
            })

        }
        console.log(event)
        events = event
                    return this.handleEv(event)
    })

    AsyncStorage.getItem('ap:auth:user').then(item =>{
        this.setState({user:item})
    })

    AsyncStorage.getItem('nextDest:country').then(item =>{
        this.setState({country:item})
    }).catch(err => console.log(err+'  carregar country'))

    AsyncStorage.getItem('nextDest:city').then(item =>{
        this.setState({city:item})
    }).catch(err => console.log(err+'  carregar city'))

    AsyncStorage.getItem('nextDest:date').then(item =>{
        this.setState({date:item})
    }).catch(err => console.log(err+'  carregar date'))

    AsyncStorage.getItem('ap:auth:language').then(item =>{
        this.setState({
            language:item
        })
    })
}
```

Figura 39 Função componentDidMount() para carregar eventos

submitEvent()

Adiciona no Firebase na secção do utilizador atual o evento que ele decidiu adicionar.

```
//submeter evento e guardar no firebase

submitEvent = () =>{
  var clean
  AsyncStorage.getItem('ap:auth:user').then(item => {
      clean =  item.replace(/\./g, ',')
    return  data = {
     country: this.state.event.country,city: this.state.event.city,
     rendezvous: this.state.event.rendezvous,description: this.state.event.description,
     time: this.state.event.time,type: 'public',
     date: this.state.event.date,language:this.state.language,
     guide: this.state.event.guide
  }}).then( dat => {
     console.log(dat+'  user: '+clean+'   name: '+this.state.event.name)
     fetch('https://tourspot-4a52b.firebaseio.com/users/'+clean+'/events/'+this.state.event.name+'.json', {
                 headers: { "Content-Type": "application/json; charset=utf-8" },
                 method: 'PATCH',
                 body: JSON.stringify(dat)
                 })
             })
       }
```

Figura 40 Função submitEvent() para adicionar eventos

12. Profile:

componentDidMount()

Carrega as informações de perfil do utilizador e exibe-as nos campos correspondentes.

```
//funçao que corre quando o componente inicia e exibe as informações do utilizador atual
componentDidMount(){
    AsyncStorage.getItem('ap:auth:user')
    .then(item =>{
        let clean =  item.replace(/\./g, ',')
        this.setState({user:clean})
        return clean
    }).then(() => {

    fetch('https://tourspot-4a52b.firebaseio.com/users/'+this.state.user+'.json')
    .then(response => response.json())
    .then((data) => user = {username:data.name, phone: data.phone, DoB: data.DoB, gender: data.gender, base64:data.image})
    .then(user =>{

        if(typeof user.base64 === 'undefined' || user.base64 === '' || user.base64 === null){
            console.log('exist')
            this.setState({oldName:user.username, oldDate:user.DoB,oldGender:user.gender,oldPhone:user.phone})
        this.setState({
            pickedImaged: {uri: '
        })
        }
        else{
            this.setState({oldName:user.username, oldDate:user.DoB,oldGender:user.gender,oldPhone:user.phone, oldBase64:user.base64})
        this.setState({
            pickedImaged: { uri: ''+user.base64}
        })
        }
    })
    })
    if(!this.state.pickedImaged){
        this.setState({pickedImaged:{uri: '
    }
}
```

Figura 41 Função componentDidMount() para carregar informações do perfil

pickImageHandler()

Abre o ImagePicker que permite ao utilizador carregar uma imagem através dos ficheiros ou da câmara que vai servir como foto de perfil.

```
// Image Picker
pickImageHandler = () => {
    ImagePicker.showImagePicker({ title: "Pick an Image" }, res => {
        if (res.didCancel) {
            console.log("User cancelled!");
        } else if (res.error) {
            console.log("Error", res.error);
        } else {
            const encodedData = res.data;
            console.log(res.data);
            this.setState({
                pickedImaged: { uri: ''+res.data },
                base64:res.data // response.uri
            });
        }
    });
}
```

Figura 42 Função pickImageHandler() para carregar a foto de perfil

toggleEdit()

Troca os campos que exibem a informação por inputs para atualizar a mesma, e ao ser executada de novo guarda-os no Firebase.

```
//editar as informações
toggleEdit = () => {
    if(this.state.edit){
        let datee
        let namee
        let genderr
        let phonee
        let basee64
        if(!this.state.date){
            datee = this.state.oldDate
        }
        else{
            datee = Moment(this.state.date.toDateString()).format('DD/MM/YYYY')

        }

        if(this.state.gender === ''){
            genderr = this.state.oldGender
        }
        else{
            genderr = this.state.gender
        }

        if(!this.state.name){
            namee = this.state.oldName
        }
        else{
            namee = this.state.name
        }

        if(!this.state.phone){
            phonee = this.state.oldPhone
        }
        else{
            phonee = this.state.phone
        }

        if(!this.state.base64){
            basee64 = this.state.oldBase64
        }
        else{
            basee64 = this.state.base64
        }
```

Figura 43 Função toggleEdit() I

```
        data = {
            gender:genderr,name:namee,phone:phonee,DoB:datee,image: basee64
        }
    fetch('https://tourspot-4a52b.firebaseio.com/users/'+this.state.user+'/.json', {
        headers: { "Content-Type": "application/json; charset=utf-8" },
        method: 'PATCH',
        body: JSON.stringify(data)
        }).then(
            item => {
                fetch('https://tourspot-4a52b.firebaseio.com/users/'+this.state.user+'.json')
    .then(response => response.json())
    .then((data) => user = {username:data.name, phone: data.phone, DoB: data.DoB, gender: data.gender})
    .then(user =>{
        this.setState({oldName:user.username, oldDate:user.DoB,oldGender:user.gender,oldPhone:user.phone})
    })
            }
        )
    }
    this.setState({edit: !this.state.edit})
}
```

Figura 44 Função toggleEdit() II

13. Restaurants:

submitEvent()

Envia para a API do Zomato os parâmetros de pesquisa fornecidos pelo utilizador e envia o utilizador para a página seguinte, com os resultados da sua pesquisa.

```
//submeter os parametros para encontrar qual a cidade e guardar o id e as keywords no firebase
submitEvent = () => {
    console.log(this.state.city)
 fetch('https://developers.zomato.com/api/v2.1/cities?q='+this.state.city,
 {
     headers:{
         "Accept": "application/json",
         "user-key": "b13aab7ce62605797 7a18608f18c396e"
     }
 }).then(res => res.json())
 .then(parsedRes =>{

     for(let i in parsedRes.location_suggestions){
         if(parsedRes.location_suggestions[i].country_name === this.state.country && parsedRes.location_suggestions[i].name.includes(this.state.city)){
             console.log(parsedRes.location_suggestions[i].id + ' THIS IS THE KEY')
             this.setState({
                 id: parsedRes.location_suggestions[i].id.toString()
             })
             return parsedRes.location_suggestions[i].id.toString()
         }
     }

 }).then(id => {

 fetch('https://tourspot-4a52b.firebaseio.com/zomato/.json', {
     headers: { "Content-Type": "application/json; charset=utf-8" },
     method: 'PATCH',
     body: JSON.stringify({
         id: id,
         keys: this.state.keywords
     })
     })

 }).then(n => {
     this.props.navigator.push({
         screen: 'awesome-places.Restaurants2',
         title: 'Find Restaurants'
     })
 })
```

Figura 45 Função submitEvent() para resultados de restaurantes

14. Restaurants2:

componentDidMount()

Carrega os 20 primeiros resultados recolhidos da API do Zomato e guarda-os numa lista no state.

```
//funçao que corre quando o componente inicia e procura os 20 melhores restaurantes na zona escolhida pelo cliente
componentDidMount(){

    fetch('https://tourspot-4a52b.firebaseio.com/zomato/.json', {
                    headers: { "Content-Type": "application/json; charset=utf-8" },

                    }).then(res => res.json()).then(parsedRes =>{
                        this.setState({id:parsedRes.id, keywords:parsedRes.keys})
                        return data = {id: parsedRes.id, keywords:parsedRes.keys}
                    })
                    .then(data =>{
                        console.log(data.id+ ' THIS IS ID')
                        console.log(data.keywords+ ' THIS IS KEY')
                        fetch('https://developers.zomato.com/api/v2.1/search?entity_id='+data.id+'&entity_type=city&q='+data.keywords+'&start=0&count=20&sort=rating&order=desc',
        {
            headers:{
               "Accept": "application/json",
                "user-key": "b13aab7ce626057977a18608f18c396e"
            }
        }).then(res => res.json())
        .then(parsedRes => {

            let event = []
            console.log(parsedRes)

            for (let i in parsedRes.restaurants){
                event.push({name: parsedRes.restaurants[i].restaurant.name,
                    avgCost2: parsedRes.restaurants[i].restaurant.average_cost_for_two, cuisines: parsedRes.restaurants[i].restaurant.cuisines,
                    contact: parsedRes.restaurants[i].restaurant.phone_numbers,
                rating: parsedRes.restaurants[i].restaurant.user_rating.aggregate_rating,
                image: { uri: parsedRes.restaurants[i].restaurant.featured_image}, address: parsedRes.restaurants[i].restaurant.location.address
                })
            }
            console.log(event)

                        return this.handleEv(event)
        })
                    })
}
```

Figura 46 Função componentDidMount() para mostrar os resultados dos restaurantes

15. UserEvents:

componentDidMount()

Carrega todos os eventos relacionados ao utilizador.

```
//corre a função quando o componente se inicia e carrega os eventos do utilizador
componentDidMount(){

    AsyncStorage.getItem('ap:auth:user').then(user => {
        let clean =  user.replace(/\./g, ',')

        this.setState({user:clean})
        return clean
    })
    .then(clean => {
        fetch('https://tourspot-4a52b.firebaseio.com/users/'+clean+'/events.json')
    .catch(err => console.log(err))
    .then(res => res.json())
    .then(parsedRes => {

        let event = []

        for(let ev in parsedRes){
            event.push({
                ...parsedRes[ev],
                name: ev
            })

        }

        events = event
                        return this.handleEv(event)
    })
    })
```

Figura 47 Função componentDidMount()para carregar eventos do utilizador

11.3 Manual de Utilizador

11.3.1 Cliente

1. Ao abrir a APP o cliente insere os dados de login nos campos correspondentes.

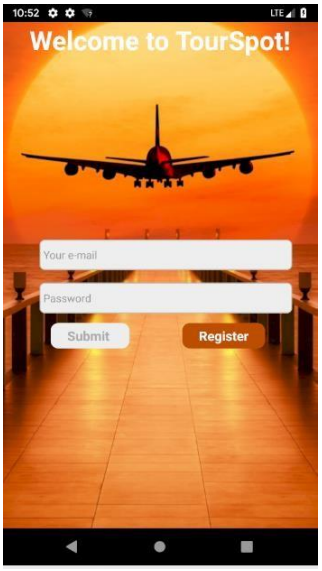


Figura 48 Página de Login

2. Se não tiver conta criada, clica no botão “Register”, insere os seus dados e cria a sua conta.

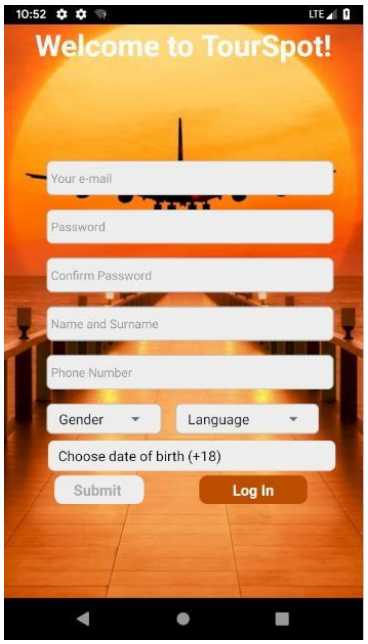


Figura 49 Página de Inscrição

3. Depois, ao entrar, na barra principal tem várias secções por onde escolher sendo a primeira a “Home”.

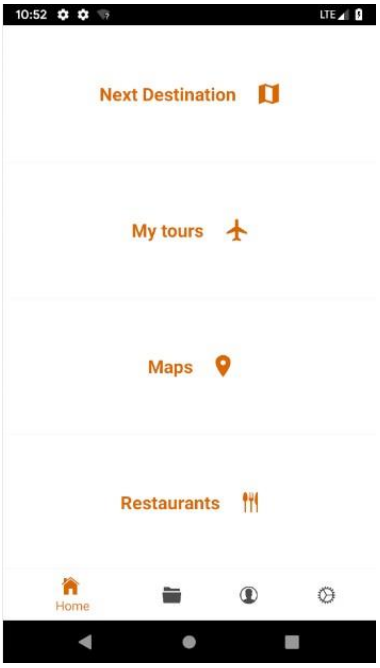


Figura 50 Página Home

4. Ao clicar em “Next Destination” é-lhe apresentada uma página com os campos relativos às tours, em que o cliente preenche e seleciona se pretende participar numa tour pública ou numa tour privada.

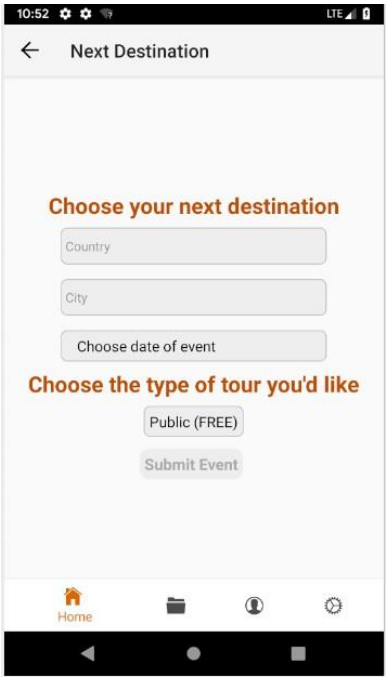


Figura 51 Página Next Destination

5. Se optar por uma tour privada, serão apresentados os guias que se encontram no local que pretende visitar, que falam o mesmo idioma que o cliente e que estão disponíveis nesse dia.

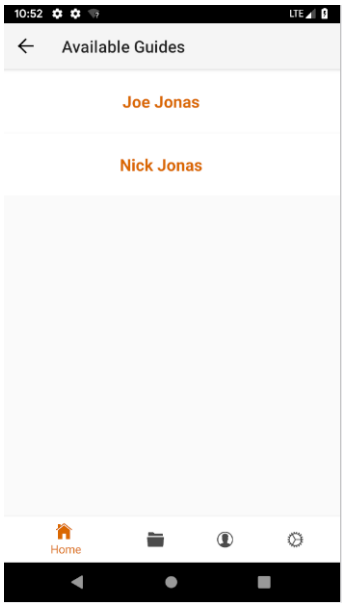


Figura 52 Página dos guias disponíveis

6. Ao selecionar o guia poderá ver em detalhe mais informações sobre o mesmo, poderá selecioná-lo com o botão "Choose this guide!" e depois em "Next" que irá aparecer ao selecionar o anterior e será reencaminhado para a próxima página ou se não o quiser pode fechar com o botão "Close" e escolher outro.

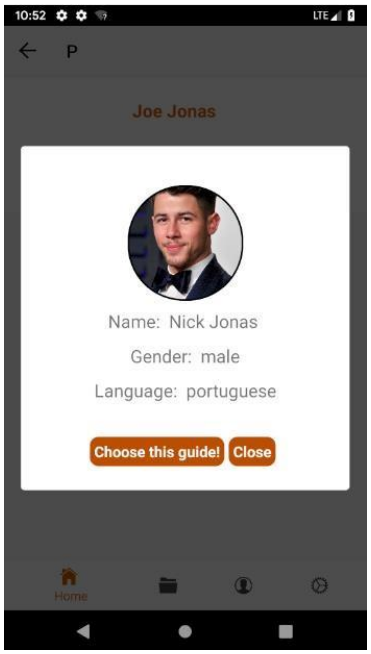


Figura 53 Página de informações do guia

7. Depois de selecionar o guia será encaminhado para a página seguinte em que o cliente tem que inserir os restantes parâmetros necessários à criação de uma tour privada, depois de inseridos deve proceder ao pagamento selecionando o botão "Payment" e apenas quando o pagamento tiver sido efetuado poderá selecionar o botão "Submit Event".

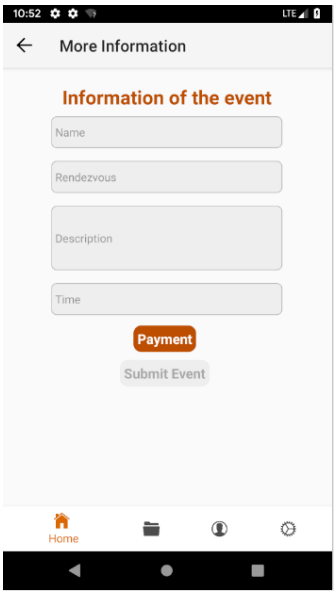


Figura 54 Página de Informação do evento

8. Se optar por tours públicas irão aparecer as tours que serão realizadas no dia e local escolhidos.

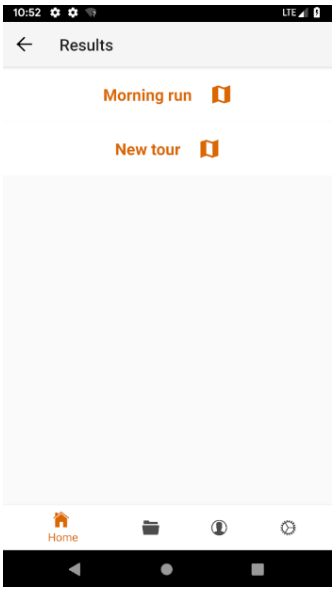


Figura 55 Página das tours públicas

9. Ao selecionar um evento o cliente obterá mais detalhes sobre ele, podendo adicioná-lo aos seus eventos com “Add to My Tours” ou fechar com “Close”.

Figura 56 Informação da tour

10. Voltando à página “Home” o cliente pode visualizar as suas tours ao selecionar a opção “My Tours”.

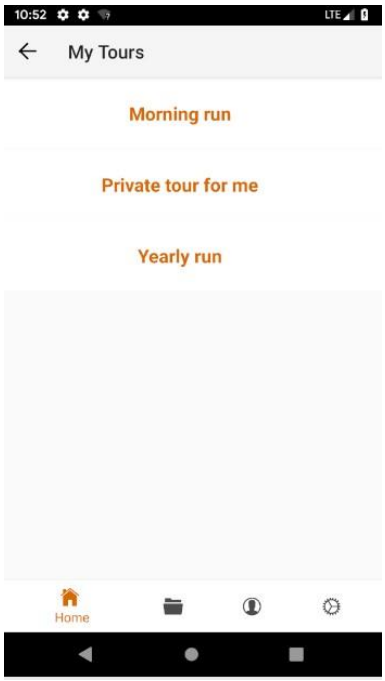


Figura 57 Página das minhas tours

11. Ao selecionar um evento pode obter mais detalhes.

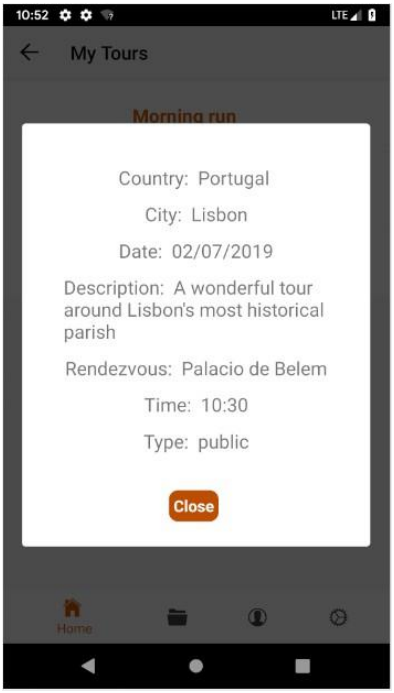


Figura 58 Página com detalhe da tour

12. Voltando à "Home" o cliente pode escolher a opção "Maps" para se orientar ou encontrar os mais diversos transportes através da nossa aplicação em parceria com o Google Maps. Primeiro insere a origem ou pode selecionar o botão "Get Current Location" e insere o destino, ao selecionar o botão "OK" será reencaminhado para o Google Maps com os parâmetros fornecidos pela nossa aplicação.

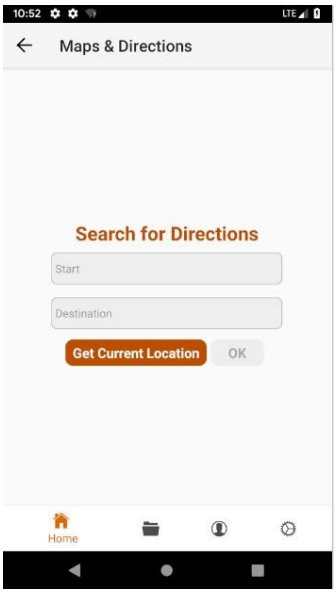


Figura 59 Página dos Maps

13. Podemos observar as várias opções existentes para o cliente explorar.

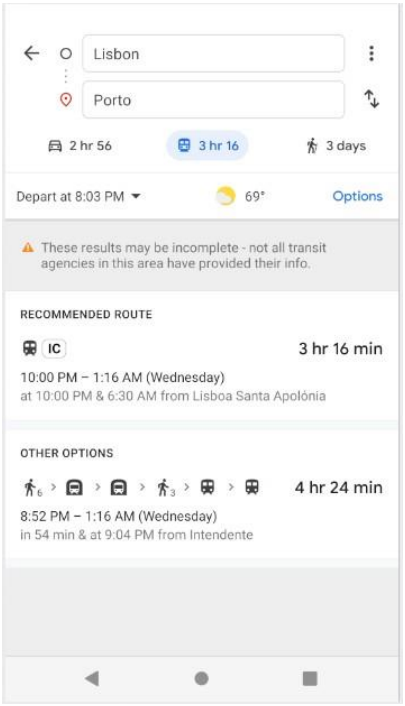


Figura 60 Página do Google Maps

14. Também na "Home" o cliente pode selecionar a opção "Restaurants", uma colaboração entre a TourSpot e o Zomato para proporcionar ao cliente a melhor experiência gastronómica quando for viajar. Nesta página o cliente apenas precisa de inserir a cidade e o país e o tipo de comida (italiano, sushi, etc.)

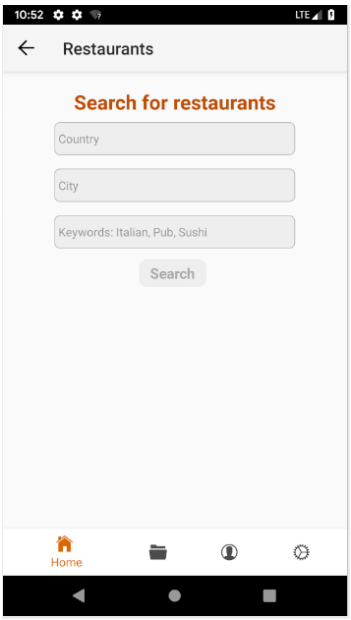


Figura 61 Página para procura de Restaurantes

15. Ao procurar por restaurantes italianos em Lisboa, temos como resultado os 20 melhores restaurantes, com base no rating do Zomato que correspondam aos parâmetros de pesquisa.

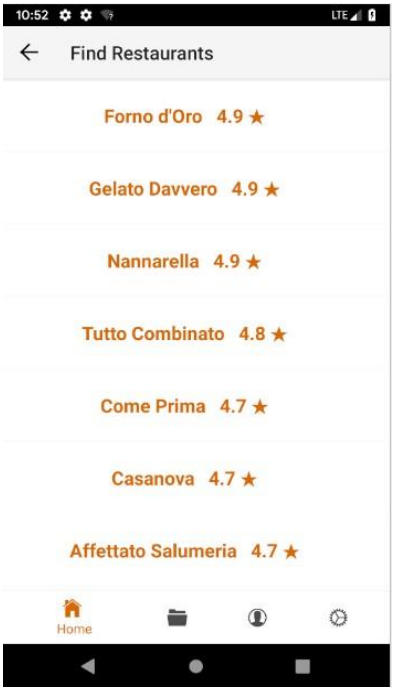


Figura 62 Página dos resultados dos restaurantes

16. Ao selecionar um deles, obtém mais informações.

Figura 63 Página de informação do restaurante

17. Depois na barra principal temos a secção “Documents” que permite carregar e visualizar documentos PDF (bilhetes de avião, cinema, comboio, etc.) de forma dinâmica e prática.

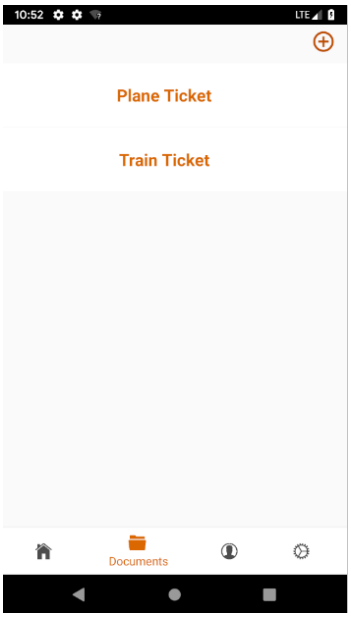


Figura 64 Página dos documentos

18. Pode adicionar um Documento ao selecionar o ícone “+” no canto superior direito do ecrã e irá abrir uma página para adicionar o novo documento. Com isto, metemos o nome do documento selecionamos “Upload” e será aberta uma página com o explorador de ficheiros do dispositivo para selecionar o documento pretendido, depois de o carregar, seleciona o botão “Submit”.

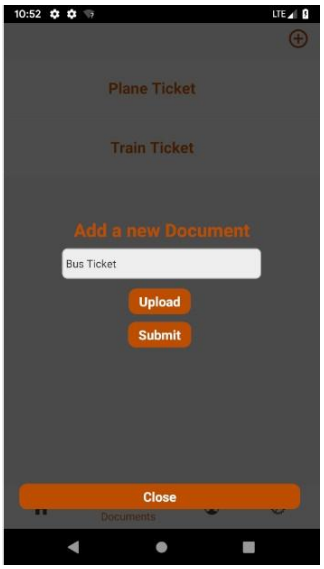


Figura 65 Página de upload de documentos

19. De seguida observará que o documento já se encontra adicionado.

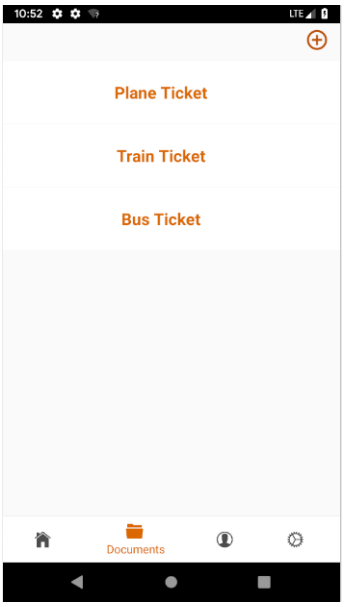


Figura 66 Página com exemplos de documentos carregados

20. O cliente pode visualizar o documento ao selecioná-lo e apagá-lo ao selecionar o botão do ícone do lixo vermelho no canto superior direito ou fechar a visualização ao selecionar “Close”.

Figura 67 Visualização do pdf

21. De seguida, na barra principal na secção “Profile” contém as informações sobre o cliente.

Figura 68 Página do perfil

22. Ao selecionar o ícone de edição no canto superior direito o cliente pode editar as suas informações bem como, adicionar uma foto ao selecionar a imagem da fotografia.

Figura 69 Página de editar perfil

23. Na barra principal passando para a secção das “Options”, o cliente tem várias informações sobre a aplicação, para ajuda e alteração de dados pessoais como também para efetuar o “Logout” da aplicação.

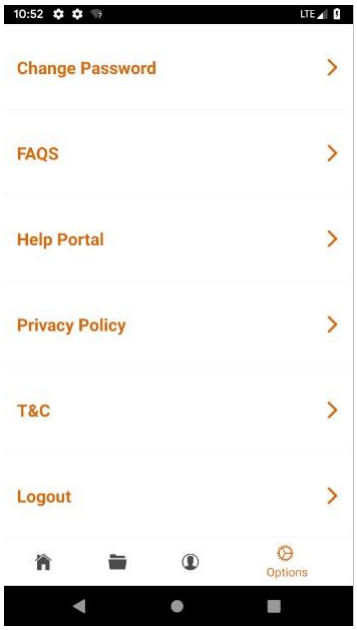


Figura 70 Página das opções

24. Ao selecionar a opção “Change Password” o utilizador pode mudar a sua password inserindo a nova password e confirmando, de seguida seleciona o botão “Change password”.

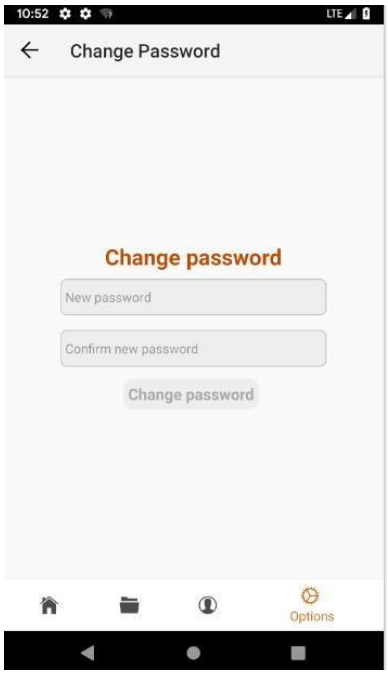


Figura 71 Página de alterar password

25. De volta à secção “Options” temos a página das “FAQs” que contém as perguntas mais frequentes relacionadas com a aplicação.

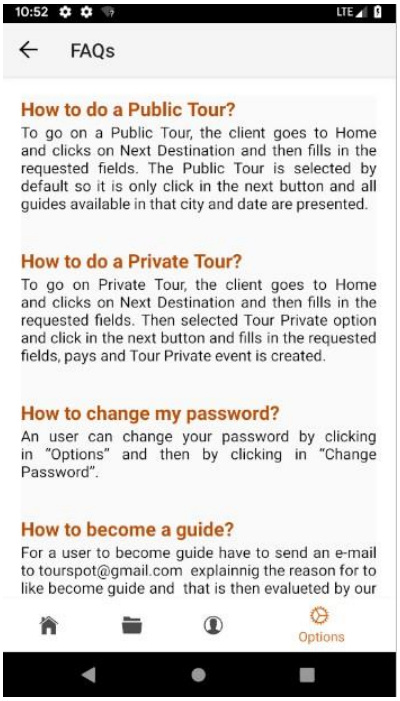


Figura 72 Página das FAQs

26. O cliente pode também selecionar a página “Help Portal” nas “Options” que contém a informação relativa ao suporte e aos contactos da TourSpot.

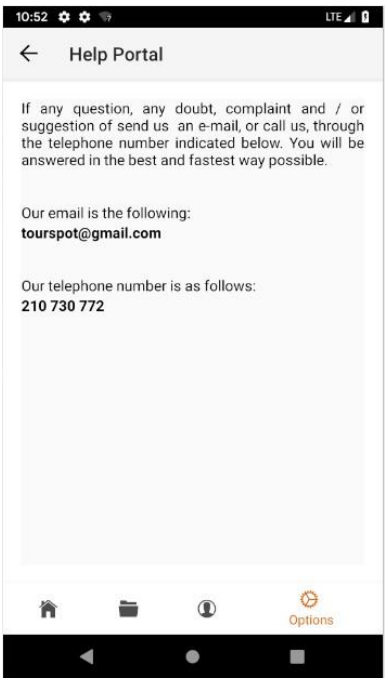


Figura 73 Página do Portal de Ajuda

27. Nas “Options” temos também a página de “Privacy Policy” que contém a informação acerca das políticas de privacidade.

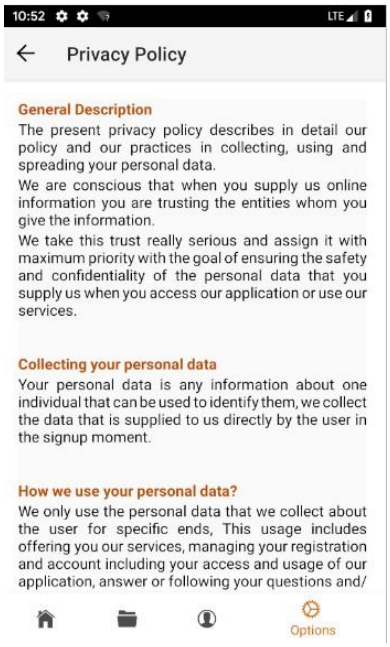


Figura 74 Página Política de Privacidade

28. Também nas “Options” o cliente pode ler os termos e condições em “T&C”.

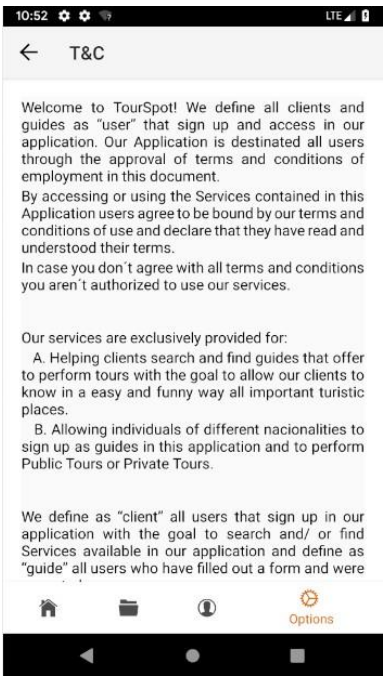


Figura 75 Página Termos e Condições

29. Quando desejar sair e fechar a sessão, o cliente pode selecionar a opção “Logout” nas Options voltando ao ecrã inicial.

11.3.2 Guia

A interface do guia comparada à do cliente difere nas seguintes páginas e funcionalidades:

1. A secção “Home” do guia é diferente da do cliente.

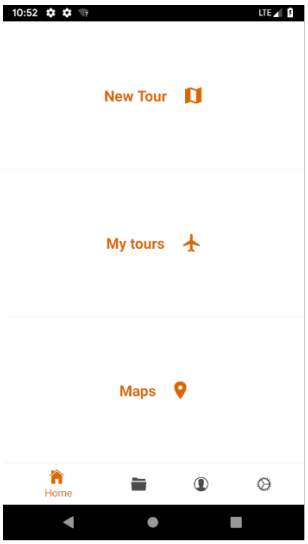


Figura 76 Página Home do guia

2. Ao selecionar “New Tour” o guia vai ser encaminhado para a página de criação de tours públicas em que terá que inserir os dados necessários à criação da tour. De seguida, seleciona “Submit event” para publicar a tour na aplicação.

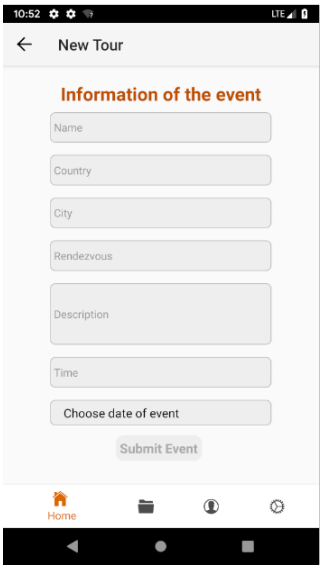


Figura 77 Página de criar uma tour pública

3. Voltando à “Home”, pode selecionar a opção “My Tours” em que visualiza todas as tours realizadas por ou requisitadas ao guia em que as novas tours privadas aparecem com *NOVO* à frente de forma a notificar o guia, desaparecendo quando o guia selecionar essa tour para obter mais informação.

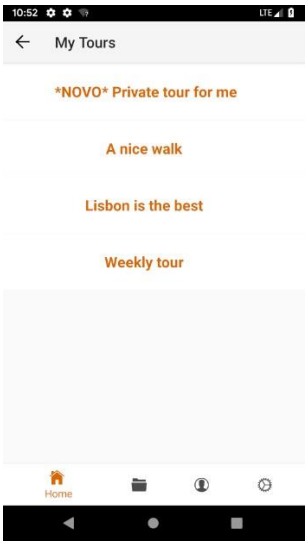


Figura 78 Página My Tours

4. Ao selecionar uma dessas tours é aberta uma página com informação mais detalhada acerca das tours.

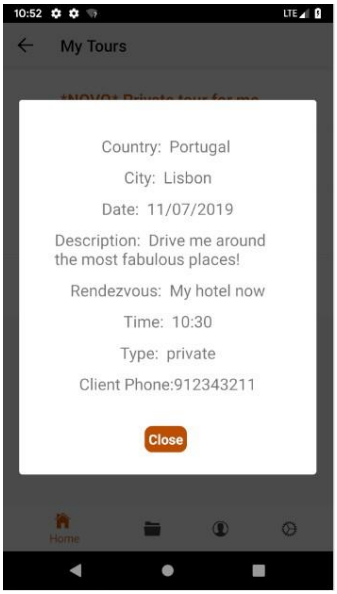


Figura 79 Página de informações da tour

5. Na secção das “Options” o guia dispõe de uma opção extra comparado ao cliente, que é a de “Change Guide Location” na qual o guia pode alterar a sua localização atual em que está a realizar tours. Ao selecionar o botão “Change” o guia altera a sua localização.

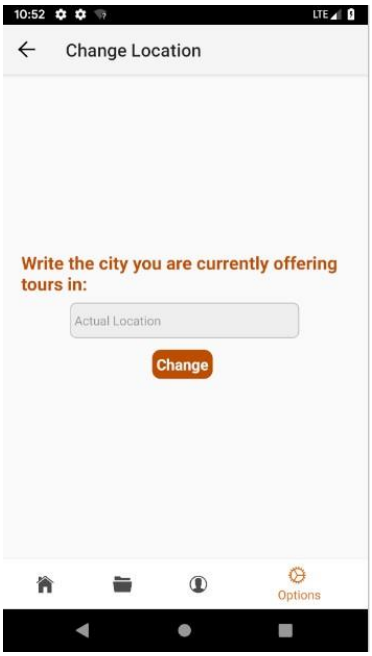


Figura 80 Página de Alteração de Localização