Universidade da Beira Interior
Departamento de Informática

PlanEvents

Elaborado por:

Maria João Marques Pais, Nº45564

Orientador:

Professor Doutor Paulo Fazendeiro

19 de julho de 2023

Agradecimentos

A conclusão deste trabalho, bem como a maior parte da minha vida académica, não seria possível sem o apoio da minha família e amigos.

Este trabalho representa o culminar de três anos de esforço e muita dedicação, dos quais sinto muito orgulho pelo percurso que tracei.

Agradeço em especial à minha família, que me ofereceu apoio incondicional, incentivo, amor, carinho e paciência nos momentos mais difíceis. Obrigada por acreditarem sempre em mim. Sem cada um de vós, não estaria onde estou hoje.

Deixo também um agradecimento especial ao meu orientador de projeto, Professor Doutor Paulo Fazendeiro, pois sem a sua orientação e imprescindível ajuda, a conclusão deste trabalho não teria sido possível. A sua disponibilidade e os seus conselhos foram fundamentais para a realização deste projeto.

Gostaria ainda de agradecer a todos os meus amigos pelo encorajamento e apoio e dado ao longo da minha vida e, em especial, durante a elaboração deste trabalho.

Conteúdo

Lista de Figuras

Lista de Excertos de Código

Acrónimos

RSVP *Répondez S'il Vous Plaît*

CRM *Customer Relationship Management*

API *Application Programming Interface*

SDK *Software Development Kit*

UML *Unified Modeling Language*

Capítulo

1

Introdução

1.1 Enquadramento

O presente documento, associado ao trabalho intitulado **Projeto**, foi realizado no âmbito da Unidade Curricular de Projeto que se enquadra no terceiro ano da Licenciatura em Engenharia Informática na Universidade da Beira Interior.

1.2 Motivação

Numa era cada vez mais tecnológica, de maneira a reduzir o consumo de papel esta aplicação seria a perfeita solução. Ao ter os eventos e os lugares de cada evento no dispositivo móvel, não seria necessário o uso de papel tanto para a publicidade do evento, como para no decorrer do evento ser possível a realização das tarefas. E ainda acrescenta o facto de ser possível a procura dos lugares e eventos no mapa. O interesse por este projeto surgiu pela realização de eventos semelhantes ao que a aplicação fornece e na realização destes senti uma certa necessidade de ajuda na procura dos lugares e com esta aplicação por meio de um simples mapa este procura torna-se mais intuitiva.

1.3 Objetivos

Este projeto visa desenvolver uma aplicação para acompanhamento de equipas em atividades de *Team Building* no exterior, em que é possível ter acesso à localização do utilizador durante o período do evento. Para esse fim, é necessário completar as seguintes etapas:

1. Criação dos eventos;

2. Inscrição do utilizador num evento;
3. Deteção da localização do utilizador;
4. Interface que permita a visualização da localização do utilizador;
5. Interface que permita a visualização da localização dos eventos e os seus respetivos lugares.

1.4 Organização do Documento

De modo a refletir o trabalho feito, este documento encontra-se estruturado da seguinte forma:

1. Capítulo1 – **Introdução** – Neste capítulo é elaborada uma pequena introdução sobre os principais objetivos deste trabalho com a motivação para a realização do mesmo;
2. Capítulo2 – **Estado de Arte** – Nesta secção faz-se referência ao que já se tem descoberto sobre o assunto em questão, evitando investigações redundantes, otimizando assim o uso do tempo. Além disso, auxilia na melhoria e desenvolvimento de novos produtos;
3. Capítulo3 – **Engenharia de Software** – Nesta secção são apresentados os requisitos da aplicação e as tabelas de *Unified Modeling Language* (UML);
4. Capítulo4 – **Tecnologias e Ferramentas Utilizadas** – Neste capítulo são apresentadas todas as tecnologias utilizadas como também a sua interação com o trabalho;
5. Capítulo5 – **Implementação e Testes** – Neste capítulo são apresentadas as principais classes utilizadas na realização da aplicação e os serviços implementados;
6. Capítulo6 – **Conclusões e Trabalho Futuro** – Neste capítulo, são descritos possíveis incrementos a realizar na aplicação de modo a otimizar a experiência do utilizador.

Capítulo

2

Estado da Arte

2.1 Introdução

Neste capítulo são apresentadas diferentes aplicações móveis para *smartphones*. Todas as aplicações apresentadas enquadram-se na gestão de eventos, e contêm o maior número de funcionalidades em comum com a aplicação que se planeia desenvolver.

2.2 Aplicações Modelo

2.2.1 *RSVPify*

A RSVPify, *Répondez S'il Vous Plaît* (RSVP) é uma plataforma online de gestão de eventos que simplifica o processo de envio de convites, recolha de RSVPs e gestão de listas de convidados para vários tipos de eventos. Quer esteja a planear um casamento ou um evento empresarial a RSVPify dispõe de funcionalidades para simplificar o processo de planeamento de eventos. Principais características:

1. Convites personalizados para eventos;
2. Gestão de RSVP;
3. Gestão da lista de convidados;
4. Acompanhamento e notificações de RSVP;
5. Organização dos utilizadores pelos lugares do evento;
6. Emissão de bilhetes e pagamentos;

7. Integração e personalização;
8. Análises e relatórios.

2.2.2 Bitrix24

O Bitrix24 é uma plataforma de gestão e colaboração empresarial que dispõe de ferramentas e funcionalidades concebidas para melhorar a produtividade, a comunicação e a gestão de projetos nas organizações. Fornece um espaço de trabalho centralizado onde as equipas podem colaborar, comunicar e gerir vários aspetos do seu trabalho. Principais características:

1. Colaboração e Comunicação;
2. Gestão de projetos;
3. *Customer Relationship Management* (CRM);
4. Gestão de documentos;
5. Gestão de funcionários e ferramentas de Recursos Humanos;
6. Automatização do fluxo de trabalho;
7. Acesso à nuvem;
8. Integração e *Marketplace*.

2.2.3 *Accelevents*

A *Accelevents* é uma plataforma de eventos virtuais e híbridos que fornece um conjunto de ferramentas e funcionalidades para auxiliar os organizadores a criar eventos em linha, de carácter envolvente e interativo. Quer se trate de uma conferência ou um evento de angariação de fundos, a *Accelevents* oferece funcionalidades para melhorar a experiência dos participantes e simplificar a gestão do evento. Principais características:

1. Plataforma de eventos virtuais e híbridos;
2. *Networking* e envolvimento;
3. Registo e emissão de bilhetes;
4. Gestão de patrocínios e expositores;
5. Análises e relatórios;

6. Personalização e *branding*;
7. Integração e *Application Programming Interface* (API).

2.3 Conclusões

Este capítulo teve como principal objetivo apresentar as diferentes aplicações móveis presentes no mercado e evidenciar as suas principais características. É de realçar que nenhuma das aplicações referidas oferece o acompanhamento virtual da localização dos utilizadores. Tendo o acompanhamento virtual é possível a monitorização de todos os participantes, podendo saber se tudo corre como planeado, tornando assim o acompanhamento virtual uma grande vantagem relativamente a outras plataformas.

Capítulo

3

Engenharia de Software

3.1 Introdução

Engenharia de software é uma área da engenharia e da computação voltada à especificação, desenvolvimento, manutenção e criação de software visando a organização, produtividade e qualidade. Atualmente, essas tecnologias e práticas englobam linguagens de programação, base de dados, ferramentas, plataformas, bibliotecas, padrões de projeto de software, processo de software e qualidade de software. Além disso, a engenharia de software deve oferecer mecanismos para planear e gerir o processo de desenvolvimento de um sistema computacional de qualidade que atenda às necessidades de um consumidor.

3.2 Requisitos Funcionais

Requisitos funcionais são todas as características ou funcionalidades esperadas que podem ser atendidas pela aplicação. De forma geral, representam o que o sistema deve fazer.

1. A aplicação deve apresentar as opções de 'Login' e 'Registo';
2. A aplicação deve permitir a criação de eventos e a edição dos mesmos;
3. A aplicação deve permitir a criação de lugares do evento e a edição das mesmas;
4. A aplicação deve permitir o acompanhamento dos participantes dos eventos;

5. A aplicação deve permitir a inscrição do utilizador num evento;

6. A aplicação deve permitir a alteração da palavra-passe, em caso de esquecimento da mesma;

7. A aplicação deve permitir a visualização dos eventos no mapa;

8. A aplicação deve permitir a visualização dos lugares do evento no mapa.

3.3 Requisitos Não Funcionais

Os requisitos não funcionais definem as propriedades e as restrições da aplicação.

1. A aplicação deve ser intuitiva, de modo a permitir uma fácil utilização por parte de pessoas com pouca experiência com dispositivos tecnológicos;

2. A aplicação deve apresentar uma interface limpa com uma palete de cores agradáveis;

3. A aplicação deve ter acesso aos serviços de localização disponibilizados pelo dispositivo;

4. A aplicação deve permitir o armazenamento das informações dos utilizadores e eventos na base de dados;

5. A aplicação deve ser estável para garantir que o utilizador não encontre *bugs* ou falhas que ponham em causa a utilização das funcionalidades dessa aplicação.

6. A aplicação deve ter acesso à internet;

3.4 Casos de Uso

Estes diagramas demonstram o sistema, quem interage com este, e apresenta de maneira simples o que o sistema faz.

Na figura 3.1 mostram-se os casos de uso para o Login e Registar, nas figuras 3.2,3.3 e 3.4 mostram-se os casos de uso para o menu de administrador, menu de monitor e menu de utilizador, respetivamente.

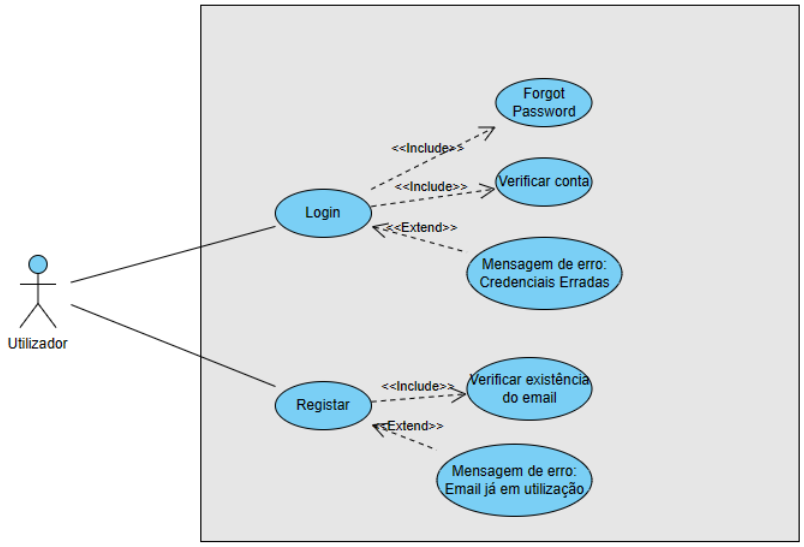

Figura 3.1: Diagrama de caso de uso do *Login* e Registar.

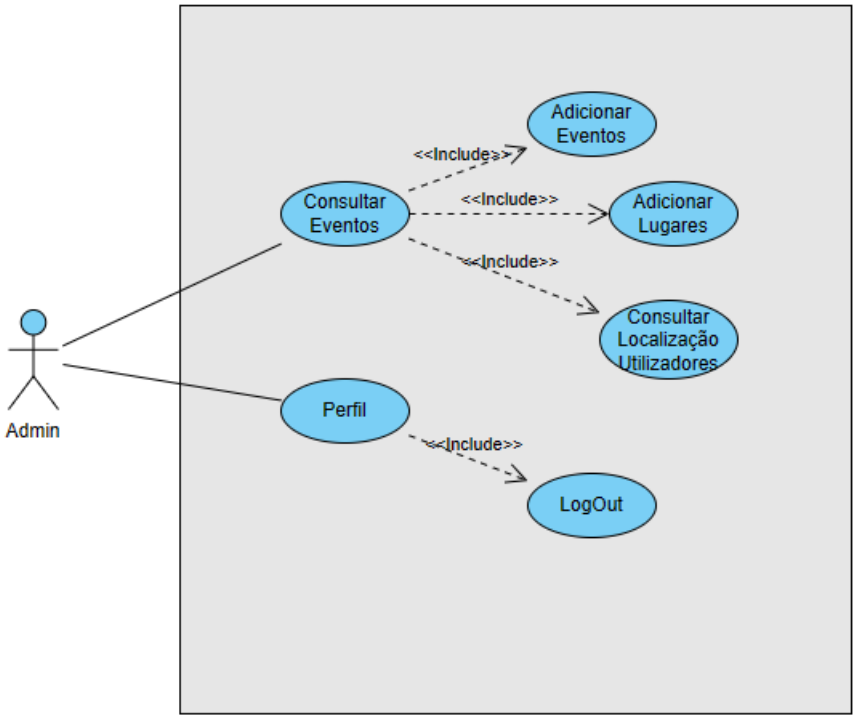

Figura 3.2: Diagrama de caso de uso do menu administrador.

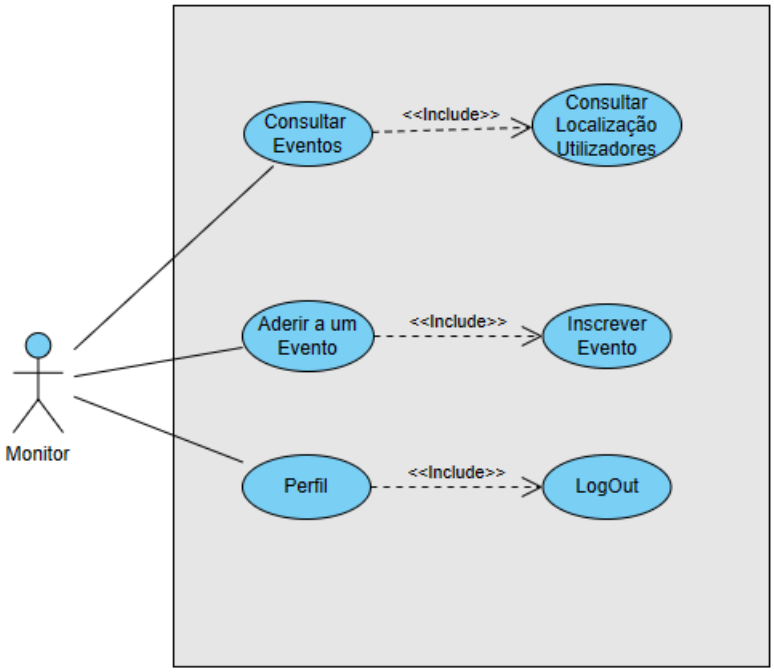

Figura 3.3: Diagrama de caso de uso do menu monitor.

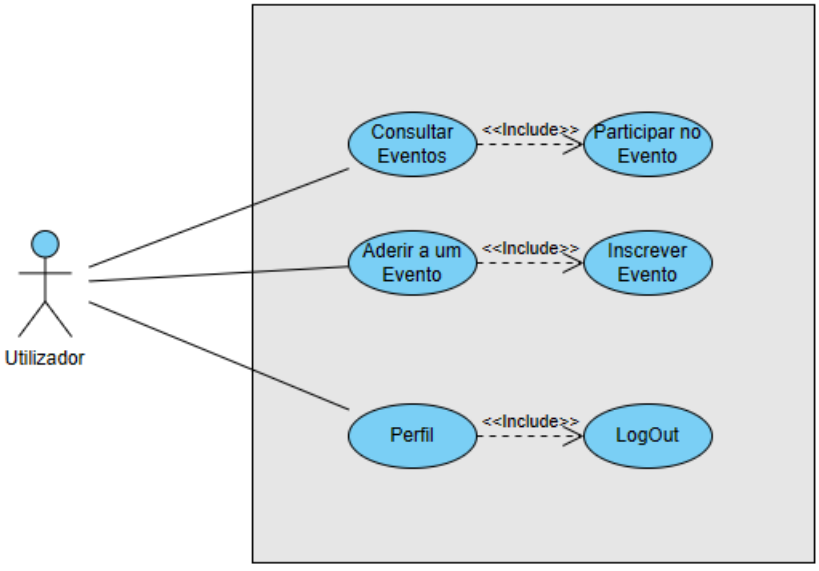

Figura 3.4: Diagrama de caso de uso do menu utilizador.

3.5 Diagramas de Atividade

Um diagrama de atividade é um gráfico que mostra o fluxo de controlo necessário para realizar uma ação. A grande vantagem deste diagrama é que mostra a lógica por detrás da ação.

Nas figuras 3.5 e 3.6 são apresentados o diagrama de atividade relativamente ao login e ao registo, respetivamente.

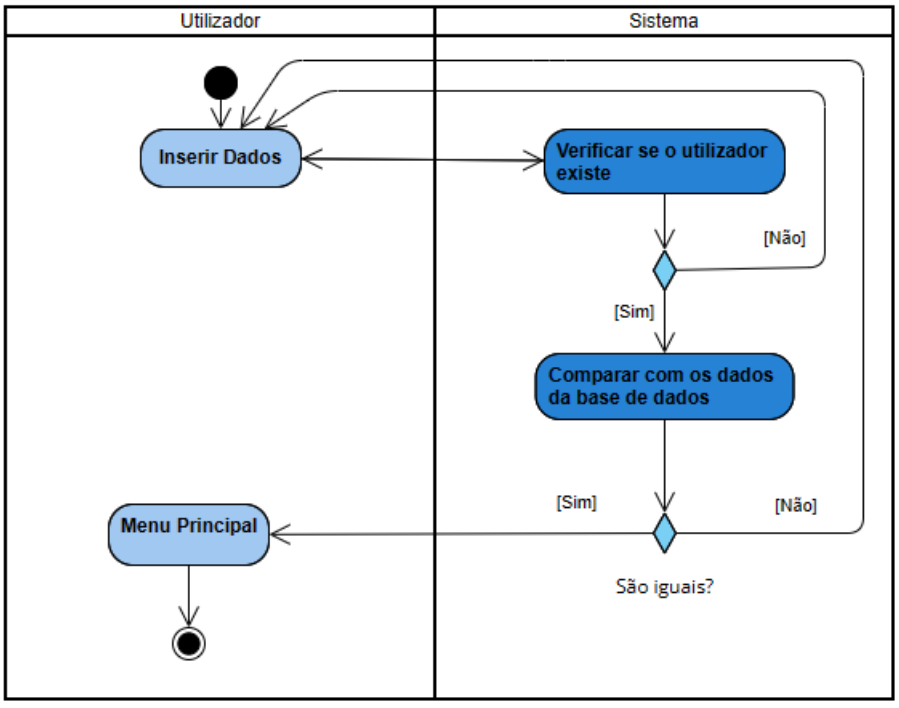

Figura 3.5: Diagrama de atividade do *Login*.

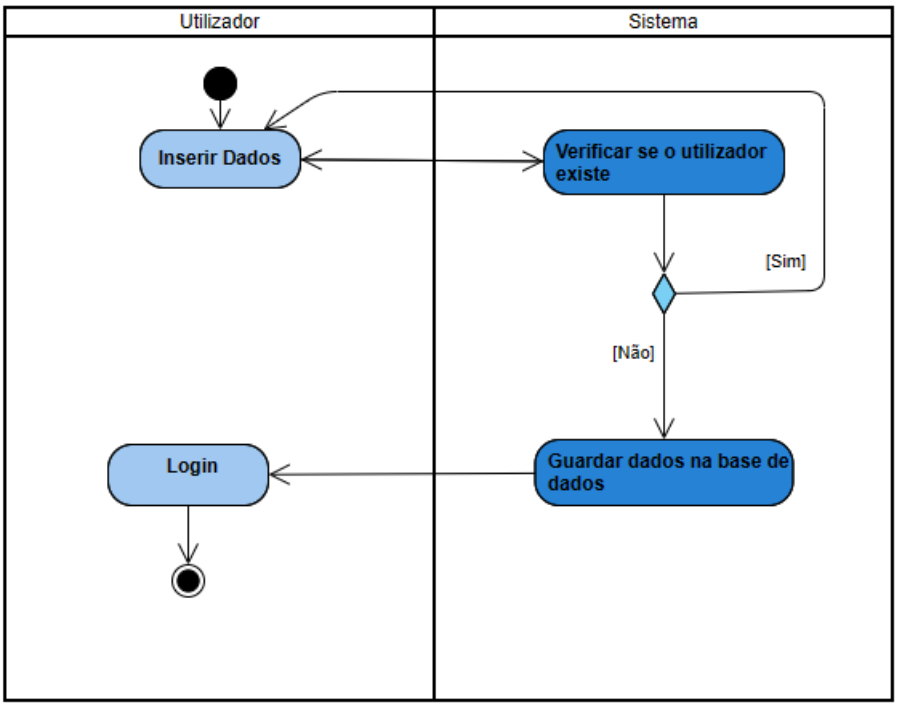

Figura 3.6: Diagrama de atividade do Registar.

Na figura 3.7 é apresentado o diagrama de atividade relativamente ao Adicionar Evento, na qual representa o ciclo que existe na ação de criação de eventos.

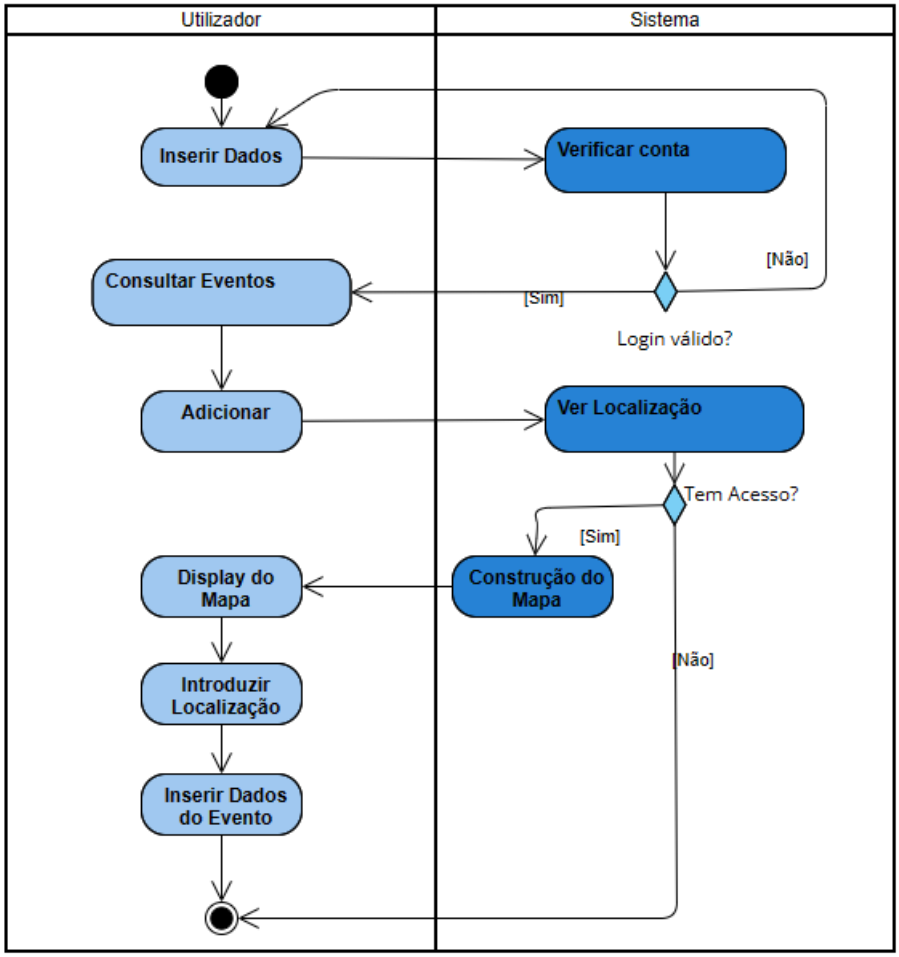

Figura 3.7: Diagrama de atividade do Adicionar Evento.

Na figura 3.8 é apresentado o diagrama de atividade relativamente ao Adicionar Lugar, na qual representa o ciclo que existe na ação de criação de lugares.

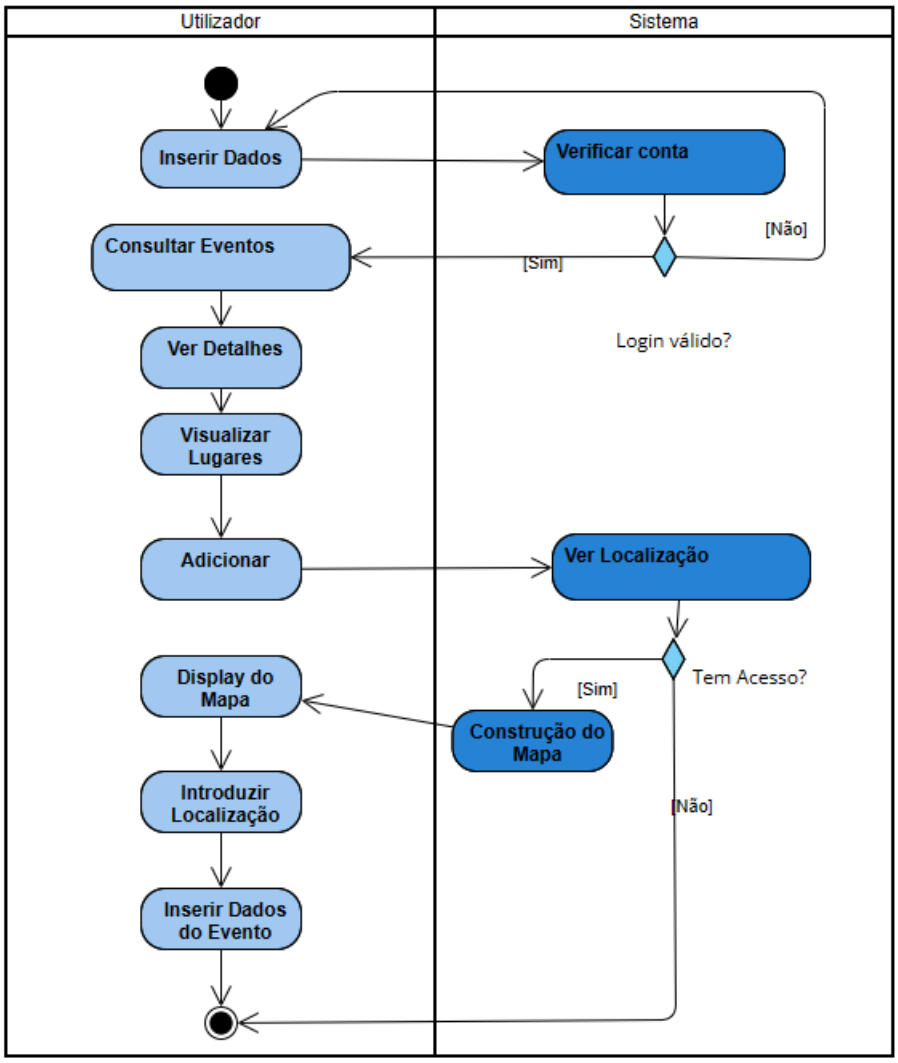

Figura 3.8: Diagrama de atividade do Adicionar Lugar.

3.6 Diagramas de Estado

Um diagrama de estados é um tipo de diagrama UML que mostra transições entre vários objetos.

O diagrama de estados para o login está representado na figura 3.9 enquanto o diagrama de estados para o registo de um utilizador na aplicação apresenta-se na figura 3.10.

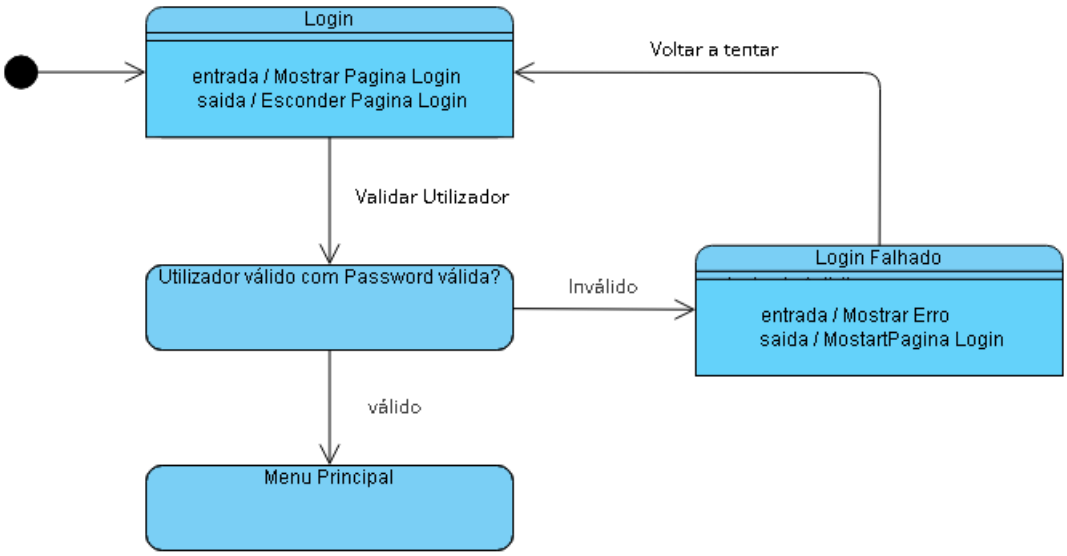

Figura 3.9: Diagrama de estados do Login.

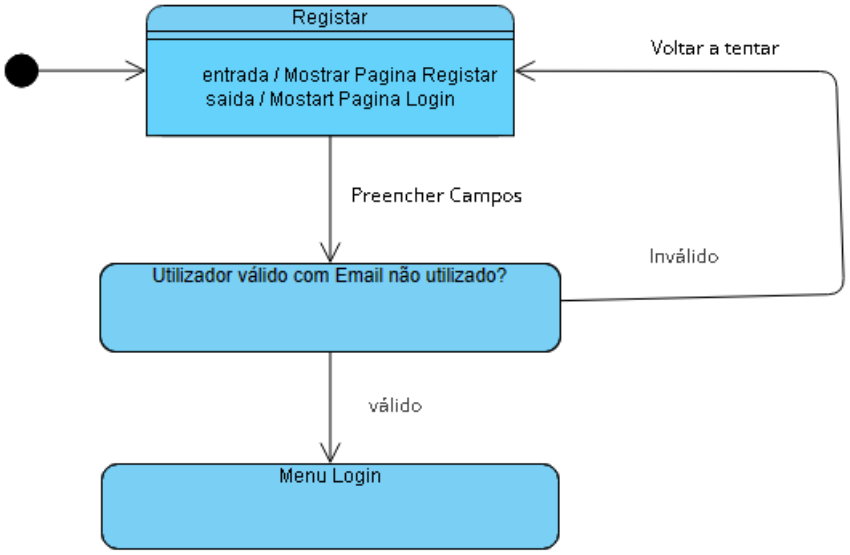

Figura 3.10: Diagrama de estados do Registar.

Capítulo

4

Tecnologias e Ferramentas Utilizadas

4.1 Introdução

No presente capítulo são apresentadas as tecnologias e as ferramentas que permitiram o desenvolvimento da aplicação. Elas são de caráter obrigatório e fundamental para o funcionamento da aplicação.

4.2 Bibliotecas

4.2.1 *Firebase Realtime Database*

O *Firebase Realtime Database* é uma base de dados na nuvem. Os dados são armazenados como JSON e sincronizados em tempo real para cada cliente conectado. Quando se criam aplicações multiplataforma com os *Software Development Kit* (SDK) para Apple, Android e JavaScript, todos os clientes compartilham uma instância do *Realtime Database* e recebem automaticamente atualizações com os dados mais recentes.

Os principais recursos desta ferramenta são:

1. **Em tempo real** - Em vez de solicitações HTTP, o *Firebase Realtime Database* usa a sincronização de dados. Sempre que os dados são alterados, todos os dispositivos conectados recebem essa atualização em milissegundos. Deste modo, criam-se experiências colaborativas e imersivas sem se preocupar com códigos de rede;

2. ***Off-line*** - As Aplicações que utilizam Firebase permanecem responsivas mesmo em modo *off-line*, pois o SDK do *Firebase Realtime Database* mantém os seus dados em disco. Quando a conexão é restabelecida, o dispositivo cliente recebe as alterações perdidas e faz a sincronização com o estado atual do servidor;

3. **Acessível em dispositivos clientes** - A *Firebase Realtime Database* pode ser acedida diretamente de um dispositivo móvel ou navegador da Web, sem um servidor de aplicativos. A segurança e a validação de dados estão disponíveis via regras de segurança baseadas em expressão do *Firebase Realtime Database*, executadas quando os dados são lidos ou gravados;

4. **Escalonamento entre as várias bases de dados** - É possível dividir os dados entre várias instâncias da base de dados no mesmo projeto do Firebase com também controlar o acesso às informações em cada base de dados com regras personalizadas do *Firebase Realtime Database* para cada instância da base de dados.

O *Realtime Database* é uma base de dados NoSQL e consequentemente, tem otimizações e funcionalidades diferentes de uma base de dados relacional. A API do *Realtime Database* foi desenvolvida para autorizar apenas operações que possam ser executadas com rapidez. Isso possibilita uma ótima experiência em tempo real que atende a milhões de utilizadores sem comprometer a capacidade de resposta. Por isso, é importante analisar como os utilizadores precisam de aceder aos dados e estruturá-los adequadamente (4).

Nesta base de dados estão implementadas duas tabelas inerentes a este projeto. Uma delas, denominada *Users*, contém as informações pessoais de cada utilizador. A outra tabela existente, denominada Eventos, guarda todas as informações de cada evento.

4.2.2 *GeoCoder*

O *GeoCoder* é uma classe para lidar com a geocodificação e geocodificação reversa. A geocodificação é o processo de transformação de uma morada ou um descrição de um local em coordenadas (latitude e longitude). A geocodificação reversa é o processo de transformação de coordenadas (latitude e longitude) numa morada. A quantidade de informação da localização da geocodificação reversa pode variar, por exemplo, uma pode conter o endereço completo do prédio mais próximo, enquanto outra pode conter apenas o nome da cidade e o código postal (5).

```
private List<Address> address = geocoder.getFromLocation(latitude,
    longitude, 1);
Address add = address.get(0);
private String morada = add.getAddressLine(0);
```

Excerto de Código 4.1: Exemplo da implementação do *GeoCoder* numa geocodificação reversa.

4.3 *SDK Maps*

Neste projeto foi utilizado o SDK do Google Maps, por permitir criar mapas dinâmicos, interativos e personalizados (6).
Este SDK tem várias bibliotecas que permitem fazer uma customização do mapa:

- GeoJSON;
- KML;
- Mapas de Calor;
- Personalização de marcadores com ícones de balão;
- Adicionar várias camadas ao mesmo mapa;
- Gerar *clusters*;
- Codificar e descodificar polilinhas;
- Calcular distâncias, áreas e rumos usando geometria esférica.

Capítulo

5

Implementação e Testes

5.1 Introdução

Neste capítulo são descritas as classes mais importantes e os serviços implementados.

5.2 Classes

5.2.1 Localizacao

A Localizacao é uma classe que retorna as coordenadas de uma morada, ou retorna a morada de um par de coordenadas. Para este efeito foram implementadas duas funções:

- getAddress;
- getCoordenadas.

A função getAddress recebe como parâmetros um contexto, o valor da latitude e da longitude. Com a ajuda do *Geocoder* convertemos o par de coordenadas numa morada. Esta função é utilizada na criação de eventos e lugares do evento, quando escolhemos a localização desejada a partir do mapa.

```
public static String getAddress(Context context, double latitude, double
    longitude) {

        Geocoder geocoder = new Geocoder(context);
        String address = null;
```

```
        try {
            List<Address> addresses = geocoder.getFromLocation(latitude,
                 longitude, 1);
            if (addresses != null && !addresses.isEmpty()) {
                address = addresses.get(0).getAddressLine(0);
            }
        } catch (IOException e) {
        }
        return address;
    }
```

Excerto de Código 5.1: Função de conversão de coordenada numa morada.

A função getCoordenadas recebe como parâmetros um contexto e uma morada. Com a ajuda do *Geocoder* convertemos a morada num par ordenado de coordenadas. Esta função é utilizada quando apresentamos os eventos e lugares dos eventos num mapa.

```
public static LatLng getCoodenadas(Context context,String endereco) {
        geocoder = new Geocoder(context);
        List<Address> addressList;
        LatLng latLng = null;
        try {
            addressList = geocoder.getFromLocationName(endereco, 1);
            if (addressList != null) {
                latLng = new LatLng(addressList.get(0).getLatitude(),
                    addressList.get(0).getLongitude());
            }

        } catch (Exception e) {
            e.printStackTrace();
        }
        return latLng;
    }
```

Excerto de Código 5.2: Função de conversão de coordenada numa morada.

5.3 Serviços

Os Serviços são as componentes que uma aplicação pode executar operações de longa duração em segundo plano. Uma vez iniciado, um serviço pode continuar em execução por algum tempo, mesmo que o utilizador se encaminhe para outra aplicação. Além disso, uma componente pode-se vincular a um

serviço para interagir com este e até mesmo realizar operações de comunicação entre processos (IPC).

Uma das características que melhor distingue um serviço de uma atividade é o facto de não ter uma interface de utilizador.

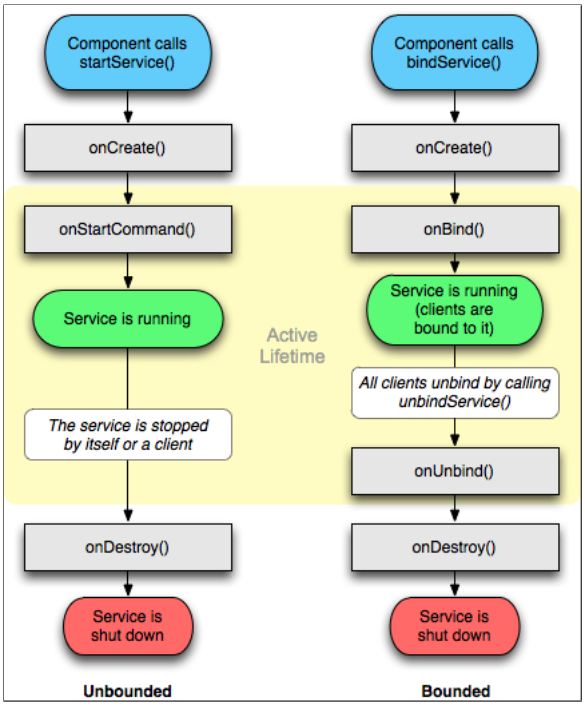

Figura 5.1: Ciclo de vida dos Serviços (10).

Neste projeto, são utilizados somente os serviços do tipo *Unbounded service*, que serão aprofundados na próxima secção (7).

5.3.1 *Unbounded service*

Este tipo de serviço é normalmente utilizado para realizar operações de longa duração e são totalmente independentes da componente onde começam o seu ciclo de vida. Um dos métodos mais importantes é o onStartCommand(), onde basicamente devem ser tomadas as providências necessárias para que as tarefas que definem o serviço sejam desempenhadas.

Existem dois tipos de *Unbounded Services* :

- ***Foreground***;

- ***Background***.

Um serviço em *Foreground* é uma aplicação que executa operações visíveis. A notificação aparece na barra de *status* e não pode ser descartada a menos que o serviço já tenha sido concluído ou tenha sido removido. Para dar início a esta atividade, é necessário chamar o método startForeground().

Os serviços em *Background* são componentes da aplicação móvel que executam tarefas de longa duração sem qualquer interação com o utilizador. Este tipo de serviços é executado somente quando a aplicação também se encontra em execução, pelo que será terminado juntamente com a aplicação(8) (9).

5.3.2 *LocationService*

Este serviço tem como propósito recolher a localização do utilizador a cada 30 segundos, sendo esta guardada na base de dados para depois ser possível visualizar no mapa a localização do utilizador durante o evento.

Para se poder usar este serviço, é necessário que o *Android Manifest* tenha as seguintes permissões:

- `android:name="android.permission.ACCESS_FINE_LOCATION";`
- `android:name="android.permission.ACCESS_COARSE_LOCATION";`
- `android:name="android.permission.ACCESS_NETWORK_STATE";`
- `android:name="android.permission.INTERNET".`

Estas permissões são fulcrais para a localização poder ser obtida.

Figura 5.2: Visualização da localização do utilizador no mapa.

Capítulo

6

Conclusões e Trabalho Futuro

6.1 Conclusões Principais

Com a elaboração deste trabalho foi possível aprofundar-se conhecimentos no ramo da Programação para Dispositivos Móveis. Foi ainda possível a aquisição de competências de execução acerca dos diversos tipos de serviços, como também da tecnologia *FireBase* aliada ao ambiente de desenvolvimento *Android Studio*. Ao longo do projeto, procuraram-se soluções que dessem resposta aos problemas e desafios.

O principal desafio deste projeto foi a aquisição da localização do utilizador em *background* ao longo do evento. Este desafio foi ultrapassado com sucesso utilizando a classe referida na secção 5.3.2. A conquista deste desafio tornou-se a maior vantagem deste projeto, destacando-se assim das outras plataformas já existentes.

Ao longo do desenvolvimento deste projeto fui encontrando alguns problemas, estando estes relacionados com a utilização dos mapas e a utilização do *Autocomplete* para a escolha da localização dos eventos e dos lugares do evento. Após um estudo mais aprofundado da utilização dos mapas e da biblioteca *Autocomplete* disponível no guia de desenvolvedor foi possível ultrapassar todos os problemas relacionados.

6.2 Trabalho Futuro

Ambiciona-se, após a conclusão deste trabalho, expandir, complementar e implementar novos conhecimentos de modo a alcançar uma versão defini-

tiva da aplicação de modo a torná-la o mais eficiente e prática possível.

Como trabalho futuro a aplicação pode ser melhorada na parte da criação de eventos, esta poderá ter como modelo outros tipos de eventos para além do que foi implementado. Pode também ser implementada a atribuição de equipas aos monitores dos eventos, para um controlo mais focado e personalizado.

Bibliografia

[1] RSVPify.
https://rsvpify.com/.
Ultimo Acesso: 03/07/2023.

[2] Bitrix24.
https://www.bitrix24.eu/.
Ultimo Acesso: 03/07/2023.

[3] Accelevents.
https://www.accelevents.com/.
Ultimo Acesso: 03/07/2023.

[4] FireBase.
https://firebase.google.com/docs/database.
Ultimo Acesso: 02/06/2023.

[5] GeoCoder.
https://developer.android.com/reference/android/location/Geocoder.
Ultimo Acesso: 11/06/2023.

[6] Mapa do Google.
https://developers.google.com/maps/documentation/android-sdk/start.
Ultimo Acesso: 25/06/2023.

[7] Serviços.
https://developer.android.com/guide/components/services?hl=pt-br.
Ultimo Acesso: 13/07/2023.

[8] Serviços Foreground.
https://developer.android.com/guide/components/foreground-services.
Ultimo Acesso: 13/07/2023.

[9] Serviço Background vs Foreground.
https://medium.com/@Codeible/understanding-and-using-services-in-android-background-foreground-services-8130f6bbf2a5.
Ultimo Acesso: 13/07/2023.

[10] Serviço em segundo plano no Android.
https://klebermota.eti.br/2011/11/08/executando-um-servico-em-segundo-plano-no-android/.
Ultimo Acesso: 17/07/2023.