APLICATIVO PARA MAPEAMENTO E SELEÇÃO DE VAGAS DE ESTACIONAMENTO E RAMPAS DE ACESSO PARA IDOSOS, DEFICIENTES FÍSICO E PESSOAS COM MOBILIDADE REDUZIDA

APPLICATION FOR MAPPING AND SELECTION OF PARKING VACANCIES AND ACCESS RAMPS FOR ELDERLY, PHYSICALLY DISABLED AND PEOPLE WITH REDUCED MOBILITY

Allef Sousa[1]
Rafael Gomes Caldas[2]
Cacildo José Devós[3]

RESUMO

O desenvolvimento deste projeto tem a finalidade de criar uma aplicação para dispositivos móveis capaz de auxiliar idosos, deficientes físicos ou pessoas com mobilidade reduzida temporária, visando informar as rotas e informações das vagas de estacionamento e/ou rampas de acesso destinados a esse público. Para o desenvolvimento do aplicativo foi utilizado a API do Google Maps, responsável por renderizar os mapas, rotas e marcações, utilizando a plataforma de infraestrutura para dispositivos móveis Firebase, responsável por oferecer suporte à autenticação dos usuários por meio de telefone e provedores de identidade federados do Google e Facebook, provendo suporte para armazenamento e sincronização em tempo real para o banco de dados. O aplicativo permite que os usuários, pesquise, selecione rotas e receba informações como distância, tempo estimado de chegada e a indicação adicionada pelos os usuários se as vagas estão sendo utilizadas naquele instante, podendo ainda por meio de colaboração entre os usuários, adicionar novas vagas de estacionamento e rampas de acesso que não estejam inseridas na aplicação, atualizando os dados em tempo real.

Palavras-chave: Mobilidade; Vagas de Estacionamento; Idosos; Deficiente Físico

ABSTRACT

The development of this project has the purpose of creating an application for mobile devices for the elderly, disabled or persons with reduced temporary mobility, for information such as routes and information on parking spaces and / or access ramps for an audience. For the development of the application was used a Google Maps API, responsible for rendering maps, routes and brands, use a platform infrastructure for mobile devices Firebase, responsible for supporting authentication of users through telephone and federated identity providers Google and Facebook, providing support for real-time storage and synchronization for the database. The application allows users to search, select routes and receive information such as distance, estimated time of arrival and indication added by users if as vacancies are being at that moment, and can also through collaboration among users, add new parking spaces and access ramps that are not inserted in the application, updating the data in real time.

Key words: Mobility; Parking lots; Elderly; Handicappe

[1]Graduando do Curso de Ciência da Computação da Universidade de Franca, allefsouza_1@hotmail.com;
[2]Graduando do Curso de Ciência da Computação da Universidade de Franca, rafael.gcaldas01@gmail.com;
[3]Professor, Orientador e Coordenador de Ciência da Computação da Universidade de Franca, email@email.com;
Franca - SP, outubro de 2017.

INTRODUÇÃO

Atualmente a mobilidade urbana é considerado um dos grandes desafios das grandes e pequenas cidades brasileiras, fruto do crescimento desenfreado e da falta de planejamento, somando – se ao crescente número da frota de veículos e da baixa qualidade da infraestrutura do transporte público, limitando o direito de ir e vir de uma grande parcela de população, os idosos, deficientes físicos e pessoas com mobilidade reduzida.

A falta de conhecimento e informação acabam por si só agravando ainda mais essa situação, pois os direitos e recursos reservados a essa parcela da população não são utilizados de forma correta. Um exemplo são as vagas de estacionamento e rampas de acesso, ondes os usuários que necessitam desse serviço desconhece suas localizações devido as más condições de sinalização e manutenção, dificultando a locomoção daqueles que fazem o uso de cadeiras de roda e outros mecanismos para se locomoverem pela cidade.

> Lei Federal nº 10.098, de 19 de dezembro de 2000 que dispõe sobre normas gerais e critérios básicos para a promoção da acessibilidade das pessoas portadoras de deficiência e com dificuldade de locomoção, que, em seu art. 7°, estabelece a obrigatoriedade de reservar 2% (dois por cento) das vagas em estacionamento regulamentado de uso público e privado para serem utilizadas exclusivamente por veículos que transportem pessoas portadoras de deficiência ou com dificuldade de locomoção. (Planalto, 2017).

Devido à essas dificuldades surgiu a motivação de desenvolver uma aplicação móvel afim de facilitar a localização dessas vagas e rampas, utilizando o Global Positioning system (GPS), onde o aplicativo permite aos usuários pesquisar e selecionar as vagas especiais e rampas de acessos mais próximas, assim como as rotas e tempo estimado de chegada ao seu destino final.

Sendo o principal objetivo tornar mais fácil a locomoção dessa parcela da população através colaboração entre os usuários, onde os mesmos poderão contribuir

enviando novas vagas e rampas de acesso não cadastrados, assim tornando a base de dados mais completa e atualizada.

DESENVOLVIMENTO

Utilizando o ambiente de desenvolvimento integrado (IDE) oficial para o desenvolvimento de aplicativos Android Studio, o aplicativo foi desenvolvido sobre a plataforma Android que é composta por um sistema operacional, middlewares, um conjunto de aplicações e o Android SDK que é um conjunto de ferramentas e APIS responsáveis pelo o desenvolvimento de aplicações para a plataforma e implementada com a linguagem de programação orientada à objetos Java.

O android é totalmente baseado no sistema operacional Linux, possui código aberto e tem sua distribuição sob a licença Apache 2.0.

> O fato de o Android ser de código aberto contribui para o seu aperfeiçoamento, uma vez que desenvolvedores de todos os lugares do mundo podem contribuir para o seu código – fonte, adicionando novas funcionalidades ou simplesmente corrigindo falhas. Já os desenvolvedores de aplicações podem desfrutar de uma plataforma de desenvolvimento moderna com diversos recursos incríveis, com tudo o que há de mais moderno. (LECHETA, 2010 p.22).

GOOGLE MAPS ANDROID API

Disponibilizado pelo Google, a api do Google Maps possibilita aos desenvolvedores criar aplicações baseadas em geolocalização, possibilitando a criação de mapas, rotas, cálculos de distâncias e tempo para vários destinos, conversão de endereços em coordenadas geográficas e várias outras funcionalidades.

Para utilizar a Google Maps Android Api é necessário instalar o SDK do Google Play Services, que contém as funcionalidades da api, sendo necessário também registrar o projeto do aplicativo no Google API Console e obter uma chave de API do Google com restrição a aplicativos Android e utilizar o certificado de assinatura na aplicação.

Após obter a chave de api, a mesma é inserida no arquivo AndroidManifest.xml como elemento filho da tag <application>.

```
<meta-data
    android:name="com.google.android.geo.API_KEY"
    android:value="YOUR_API_KEY"/>
```

Figura 1- Meta Data - Chave da Google Maps Android Api. Fonte: (Google Developers, 2017)

Após esse procedimento permissões de localização são inseridas para ser possível indicar a posição atual do usuário no mapa.

> Para proteger a integridade do sistema e a privacidade do usuário, o Android executa cada aplicativo em um sandbox de acesso limitado. Se o aplicativo quiser usar recursos ou informações fora do sandbox, ele precisará solicitar a permissão explicitamente. Dependendo do tipo de permissão solicitada pelo aplicativo, o sistema a concederá automaticamente ou solicitará que o usuário conceda a permissão. (Google Developers, 2017).

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.myapp" >
  ...
  <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
  ...
</manifest>
```

Figura 2 Permissão de localização exata. Fonte: (Google Developers, 2017)

Algumas permissões são inseridas automaticamente ao arquivo AndroidManifest.xml, não sendo necessários inseri – las explicitamente, que é o caso da permissão android.permission.INTERNET, responsável pelo download dos blocos de mapas e a permissão android.permission.ACCESS_NETWORK_STATE, que verifica o status da conexão possibilitando o download dos dados.

O mesmo caso se aplica a biblioteca OpenGL ES versão 2 onde a configuração é vinculada ao Google play Services é inserida automaticamente ao manifesto da aplicação.

Para adicionar o mapa na aplicação um elemento <fragment> é adicionado ao layout, onde é definido o atributo android: name, com o valor “com.google.android.gms.maps.MapFragment”.

```
<?xml version="1.0" encoding="utf-8"?>
<fragment xmlns:android="http://schemas.android.com/apk/res/android"
    android:name="com.google.android.gms.maps.MapFragment"
    android:id="@+id/map"
    android:layout_width="match_parent"
    android:layout_height="match_parent"/>
```

Figura 3 - Elemento Fragment. Fonte: (Google Developers, 2017)

Uma instância de MapFragment é criada onde o método FragmentTransaction.add() é invocado adicionando o Fragment à Activity atual.

```
mMapFragment = MapFragment.newInstance();
FragmentTransaction fragmentTransaction =
        getFragmentManager().beginTransaction();
fragmentTransaction.add(R.id.my_container, mMapFragment);
fragmentTransaction.commit();
```

Figura 4- Bloco de código com a instância MapFragment. Fonte: Google Developers

Para ser possível manipular o mapa a interface OnMapReadyCallback é implementada definindo a instância que retorna o objeto MapFragment, onde a interface de retorno de chamada é implementada, e o método onCreate do arquivo de layout é definido para a visualização do conteúdo. Após obter o a referência para o fragmento chamado, o método getMapAsync é invocado obtendo o retorno de chamado no elemento Fragment.

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    ...
}
```

Figura 5 - Método onCreate. Fonte: (Google Developers, 2017)

É importante utilizar o método de retorno de chamada onMapReady(GoogleMap), onde é obtido um manipulador para o objeto GoogleMap, responsável por definir opções de visualização, tipos de mapas, marcadores e outras funcionalidades, é ele quem modela o objeto de mapa na aplicação. E será invocado no momento que o mapa estiver pronto para ser utilizado.

```
@Override
public void onMapReady(GoogleMap map) {
    map.addMarker(new MarkerOptions()
        .position(new LatLng(0, 0))
        .title("Marker"));
}
```

Figura 6 - Método onMapReady. Fonte: (Google Developers, 2017).

GOOGLE PLACES API

> A Google Places API Web Service é um serviço que retorna informações sobre locais — definidos nessa API como estabelecimentos, localizações geográficas ou pontos de interesse proeminentes — usando solicitações HTTP. (Google Developers, 2017)

A api Google Places é disponibilizada para o desenvolvimento de aplicações baseadas em locais complementando os serviços baseados em geolocalização disponibilizados pelos serviços de localização da plataforma Android.

A interface Place representa um local na api, incluindo informações do local, como nome, endereço, localização, id, tipo do local entre outros.

Para manipular Google Places Api para Android assim como no Google Maps Api é necessário ter o SDK do Google Play Services instalado e a chave de api inserida no arquivo de manifesto da aplicação.

Ressaltando que apesar da Google Places Api ser disponibilizada de forma ilimitada e gratuita, devido à políticas do Google, é necessário que, após exceder o limite padrão que é de 1000 solicitações no período de 24 horas, assim como no caso de exceder o limite de 150.000 solicitações também durante o período de 24 horas a

aplicação apresentará falhas em ambos os casos, caso não tenha sido solicitado o aumento dos limites de uso da api.

Para a conexão é implementado uma instância da api dos serviços do Google play no método onCreate da activity, utilizando a GoogleApiClient.Builder inserindo as Apis no construtor.

```
import com.google.android.gms.common.api.GoogleApiClient;
import com.google.android.gms.common.api.GoogleApiClient.OnConnectionFailedListener;
import android.support.v4.app.FragmentActivity;

public class MyActivity extends FragmentActivity
        implements OnConnectionFailedListener {
    private GoogleApiClient mGoogleApiClient;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        mGoogleApiClient = new GoogleApiClient
                .Builder(this)
                .addApi(Places.GEO_DATA_API)
                .addApi(Places.PLACE_DETECTION_API)
                .enableAutoManage(this, this)
                .build();
    }

    // TODO: Please implement GoogleApiClient.OnConnectionFailedListener to
    // handle connection failures.
}
```

Figura 7 - Bloco de implementação da Google Places Api. Fonte: (Google Developers, 2017)

Ao utilizar dados do Google Places Api como resultados de buscas de endereço ou preenchimento automático, requisitos de atribuição indicando a Google como autor devem ser aplicados.

Caso a aplicação exiba dados da Places API em um mapa, o mesmo deverá ser do Google e ter o logotipo do Google visível, não precisando fornecer atribuições adicionais ao Google. Ou se a aplicação exibir dados da Places API em uma página ou exibição que não possua um mapa do Google, uma imagem "Powered by Google", já inclusa na biblioteca de serviços do Google Play deverá ser exibida com esses dados, não sendo permitida a alteração da mesma.

SELETOR DE LOCAL

O widget de IU seletor de local é utilizado para permitir que o usuário pesquise um local e obtenha os detalhes desse local como retorno.

Para poder manipular o seletor de local é necessário obter a permissão ACCES_FINE_LOCATION, para permitir que a aplicação acesse um local com precisão.

O construtor PlacePicker.IntentBuilder() é implementado para construir uma Intent que será passada para o método startActivityResult(), que permite identificar a solicitação quando o resultado for retornado.

Atribuições ao Google também devem ser consideradas ao utilizar o seletor de local.

```
int PLACE_PICKER_REQUEST = 1;
PlacePicker.IntentBuilder builder = new PlacePicker.IntentBuilder();

startActivityForResult(builder.build(this), PLACE_PICKER_REQUEST);
```

Figura 8 - Implementação do Seletor de Local. Fonte: (Google Developers, 2017)

Places Autocomplete

O widget Places Autocomplete é uma caixa de diálogo de pesquisa que fornece a funcionalidade de preenchimento automático na Google Places API for Android tornando possível a realização de pesquisas de locais. Durante a pesquisa, o Places Autocomplete retorna sugestões de locais, como estabelecimentos, endereços e pontos de interesse, que é nada mais que uma instância de Places utilizada pela aplicação para obter dados do local pesquisado.

Para manipular o widget é necessário adicionar a classe PlaceAutocompleteFragment, assim como um fragmento ao layout XML e um detector à atividade.

```
<fragment
  android:id="@+id/place_autocomplete_fragment"
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:name="com.google.android.gms.location.places.ui.PlaceAutocompleteFragment"
  />
```

Figura 9 – Adicionando o PlaceAutocompleteFragment. Fonte: (Google Developers, 2017)

Para manipular o retorno que possui a resposta da pesquisa realizada pelo usuário, uma referência ao fragmento é criada e adicionado um detector à classe PlaceAutocompleteFragment.

```
PlaceAutocompleteFragment autocompleteFragment = (PlaceAutocompleteFragment)
getFragmentManager().findFragmentById(R.id.place_autocomplete_fragment);

autocompleteFragment.setOnPlaceSelectedListener(new PlaceSelectionListener() {
    @Override
    public void onPlaceSelected(Place place) {
        // TODO: obter informações sobre o local selecionado.
        Log.i(TAG, "Place: " + place.getName());
    }

    @Override
    public void onError(Status status) {
        // TODO: Solucionar o erro.
        Log.i(TAG, "Ocorreu um erro: " + status);
    }
  });
```

Figura 10 - Criando uma referência para o PlaceAutocomplete. Fonte: (Google Developers, 2017)

FIREBASE

O Firebase é uma plataforma de desenvolvimento de aplicativos baseados em armazenamento e sincronização de dados em tempo real, oferecendo serviços baseados na nuvem, sendo integrado com várias ferramentas de desenvolvimento existentes da Google, como o Google Cloud Messaging.

O Firebase se destaca por ser uma solução completa de back-end tanto para o desenvolvimento mobile quanto web, sendo disponibilizado um SDK e um console para criar e gerenciar aplicações, oferecendo diversas ferramentas como:

- Analytics - um painel para monitorar o comportamento dos usuários da aplicação, segmentação demográfica e desempenho de campanha;
- Autenticação - suporte para autenticação via e-mail, por telefone e provedores de identidade federados como o Facebook, GitHub, Google, e Twitter;
- Relatório de Erros - monitoramento dos erros da aplicação em todos os dispositivos e é integrado com o Analytics para analisar o comportamento dos usuários após falhas;
- Database Real Time - base de dados NoSQL é utilizada e sincronizada em tempo real armazenando os dados no formato JSON;
- Notifications - que permite gerenciar notificações enviadas para os usuários da aplicação;
- Offline – permite o armazenamento dos dados na memória cache local, permitindo utilizar a aplicação em modo off-line;
- Storage – serviço de armazenamento de mídias, como áudio, imagens e vídeos;

O SDK fornece suporte para o desenvolvimento de projetos em diversas plataformas como IOS, Web, Unity e Android. Sendo hospedado e mantido pelo Google, onde é oferecido um plano gratuito para iniciantes, e diversos planos baseado em consumo.

Para o desenvolvimento na plataforma Android é necessário um dispositivo com Android 4.0 ou superior, ter o SDK do Google Play Services do Repositório do Google, e a IDE Android Studio versão 1.5 ou superior.

A configuração da aplicação com o app é realizada através do arquivo de configuração e pelo arquivo google-services.json, que é referenciado pela pasta módulo do projeto, ambos arquivos são fornecidos pelo console do Firebase na página de Configurações do Projeto.

A seguinte regra deve ser inserida no arquivo build.gradle para incluir o plugin google-services.

E o comando apply plugin: 'com.google.gms.google-services' deve ser inserido no final do arquivo para ativação do plugin do Gradle, assim como as dependências dos SDKs do Firebase.

As seguintes dependências foram utilizadas nesse projeto:

- com.google.firebase:firebase-core:11.0.4 Analytics
- com.google.firebase:firebase-auth:11.0.4 Authentication
- com.google.firebase:firebase-database:11.0.4 Realtime Database
- com.google.firebase:firebase-storage:11.0.4 Storage

FIREBASE AUTHENTICATION

O Firebase Authentication disponibiliza serviços de autenticação por meio de senhas, números de telefone e provedores de identidade federados sendo integrado a padrões como OAuth 2.0 e OpenID Connect, permitindo o acesso ao Firebase Realtime Database, informações básicas do perfil do usuário e controlar os dados armazenados em outros produtos do Firebase.

O Firebase Authentication oferece recursos como:

- Autenticação baseada em e-mail e senha;
- Autenticação através dos provedores de identidade federados;
- Autenticação por número de Telefone;

No Firebase um usuário é representado por um objeto User, que possui um conjunto de propriedades básicas como, um código exclusivo, um endereço de e-mail e senha, um nome e um URL de foto, que é armazenado na base de dados, onde não será possível adicionar novas propriedades ao objeto User.

Graças ao objeto User é possível associar mais de um método de login a um usuário possibilitando que o mesmo faça login na mesma conta, mas utilizando formas diferentes como, e-mail e senha, com o número do telefone ou utilizando o login do Facebook ou outros provedores de identidade federados. Pois a instância do objeto user monitora todos os provedores de acesso associados a um usuário.

Para configurar a autenticação na aplicação é necessário ter o SDK Firebase instalado e ter associado à aplicação ao projeto Firebase, assim como ter inserido a dependência ao arquivo build.gradle, e no caso dos provedores de autenticação a ativação no console do Firebase na seção autenticação.

O Firebase utiliza listeners responsáveis por monitorar o estado atual da instância do objeto user, que será notificado sempre que houver alguma alteração, por exemplo quando um usuário realizar login ou sair da aplicação ou quando o token de acesso for atualizado.

A autenticação disponibilizada pelo Firebase disponibiliza três tipos de tokens.

Tokens de código Firebase que é criado pelo Firebase quando um usuário realiza login em uma aplicação Firebase, permitindo identificar um usuário com segurança.

Tokens de provedores de identidade, onde tokens específicos de formatos diferentes são criados pelos provedores de identidade federados, como Google e Facebook, normalmente nos padrões de acesso OAuth 2.0, que são utilizados para verificar se a autenticação dos usuários com os provedores de identidade obteve sucesso e posteriormente os converte em credenciais utilizadas pelos serviços do Firebase.

Tokens personalizados do Firebase, que são Tokens JWTs assinados usando a chave privada de uma conta de serviço permitindo a autenticação, sendo utilizado de maneira similar aos tokens fornecidos pelos provedores de identidade federados.

Para acionar os listeners que serão responsáveis por monitorar o estado da instância do objeto user, é necessário instanciar os objetos FirebaseAuth e AuthStateListener, e no método onCreate inicializar a instância de FireBaseAuth e o método AuthStateListener, ficando a cargo dos métodos onStart e onStop responsáveis por adicionar e remover o monitoramento realizado pelo listeners respectivamente.

O SDK do Firebase permite gerenciar usuários do Firebase Authentication permitindo realizar programaticamente diversas tarefas como, criar e atualizar perfis de usuários, identificar usuários conectados, receber o perfil de um usuário, receber informações específicas dos provedores de perfil dos usuários, definir e enviar e-mails de verificação e redefinição de senhas, importar, reautenticar e excluir perfis de usuários.

```
FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
if (user != null) {
    // User is signed in
} else {
    // No user is signed in
}
```

Figura 11 - Identificando usuários conectados. Fonte: (Google Developers, 2017)

Autenticando com o login do Google

É possível permitir a autenticação dos usuários com contas do Google integrando o login do Google à aplicação. Para isso é necessário adicionar as dependências, Firebase Authentication – ‘compile 'com.google.firebase:firebase-auth:11.2.0'’ e o Login do Google – ‘compile 'com.google.android.gms:play-services-auth:11.2.0'’ no arquivo build.gradle da aplicação, e assim como em outros recursos do Firebase associar à aplicação ao projeto Firebase, e especificar a impressão digital SHA-1 da aplicação, na seção de configurações no console do Firebase.

SHA-1 é um certificado de assinatura que permite criar uma chave de cliente para a API OAuth2 para a aplicação e pode ser obtido executando o comando keytool fornecido com o Java para obter a impressão digital do certificado.

Ainda no console do Firebase na seção de autenticação o método de login do Google tem que estar ativado.

Para integrar o Login do Google com a aplicação basta configurar o objeto GoogleSignInOptions, passando o id do cliente do servidor para o método requestIdToken.

```
// Configure Google Sign In
GoogleSignInOptions gso = new GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
        .requestIdToken(getString(R.string.default_web_client_id))
        .requestEmail()
        .build();
```

Figura 12 – Objeto de Configuração para o login do Google. Fonte: (Google Developers, 2017)

Por fim no método onCreate da atividade de login, utilizando a instância compartilhada do objeto FirebaseAuth, é verificado se o usuário está conectado. Após que um usuário se conecta, o token de código recebido pelo objeto GoogleSignInAccount deve ser substituído por uma credencial do Firebase e a autenticação é realizada fazendo uma chamada para signInWithCredential, que no caso de sucesso, o método getCurrentUser poderá ser invocado recebendo os dados da conta do usuário.

Autenticando com o login do Facebook

Os procedimentos para realizar login com o Facebook em partes se assemelha com o login do Google, onde também é possível permitir a autenticação dos usuários com contas do Facebook a partir da integração do login do Facebook com a aplicação.

Onde também é necessário adicionar a dependência para o Firebase Authentication – “compile 'com.google.firebase:firebase-auth:11.2.0” no arquivo build.gradle da aplicação. Porém com o login do Facebook é necessário fornecer o Id e chave secreta fornecidos pelo Facebook no site Facebook Developers, que serão inseridos na seção de autenticação no console do Firebase, onde o método de login com o Facebook será ativado.

Na página de configuração da aplicação do Facebook, na seção de configurações do produto, a URI de redirecionamento OAuth deve estar listada como um dos URIs de redirecionamento.

Para configurar o login com o Facebook além de realizar os procedimentos especificados pelo Facebook, é necessário configurar o objeto LoginButton, solicitando as permissões public_profile e - mail.

E assim como no processo de login com o Google, no método onCreate da atividade de login, utilizando a instância compartilhada do objeto FirebaseAuth, é verificado se o usuário está conectado, após que um usuário se conecta, o método de retorno de chamada onSuccess do LoginButton é utilizado para receber um token de acesso que será substituído por uma credencial do Firebase e a autenticação é realizada fazendo chamada para signInWithCredential, que no caso de sucesso, o método getCurrentUser poderá ser invocado recebendo os dados da conta do usuário.

Autenticando com o número do telefone

O Firebase Authentication permite realizar login utilizando o número de telefone de um usuário. Após o usuário inserir o número de telefone uma mensagem SMS com um código de verificação de uso único é enviada ao telefone dele, que é inserido ao realizar o login.

Vale a pena ressaltar que apesar de ser mais simples e conveniente autenticação usando apenas um número de telefone, essa forma de realizar login é considerada menos segura que os outros métodos disponibilizados pelo o Firebase,

pois um número de telefone pode ser facilmente transferido entre usuários, por isso a Google recomenda que outros métodos mais seguros de login também devem ser oferecidos.

Para ativar o login com um número de telefone no console do Firebase, na seção autenticação, o método de login com um número de telefone deve estar ativado, da mesma forma como é feito nos outros métodos.

E para enviar o código de verificação para o usuário uma interface com um formulário deve será exibida, assim como um aviso indicando que irá receber um sms para verificação do número.

O método PhoneAuthProvider.verifyPhoneNumber receberá o número de telefone como parâmetro e repassará para verificação no Firebase. E por mais que o método verifyPhoneNumber seja invocado mais de uma vez ele não enviará uma segunda mensagem até que o tempo limite da solicitação tenha expirado.

Quando o método PhoneAuthProvider.verifyPhoneNumber é invocado, é necessário fornecer uma instância de OnVerificationStateChangedCallbacks que contenha implementações das seguintes funções de retorno de chamada, onVerificationCompleted e onVerificationFailed que manipulam os resultados da solicitação.

O método onVerificationCompleted(PhoneAuthCredential) é chamado em duas situações, na verificação instantânea quando o número de telefone pode ser verificado instantaneamente sem que você precise enviar ou inserir um código de verificação. E na recuperação automática, que ocorre quando o Google Play Services pode detectar automaticamente o SMS de verificação recebido e executar a verificação sem a ação do usuário, recurso que pode não estar disponível para algumas operadoras.

Mas nas duas situações o número de telefone do usuário foi verificado, possibilitando utilizar o objeto PhoneAuthCredential que é retornado para fazer o login do usuário.

O método onVerificationFailed(FirebaseException) é chamado em situações de resposta a uma solicitação de verificação que falhou devido a um código de verificação inválido.

Logo após o envio do código de verificação o objeto PhoneAuthCredential possuirá o código e a identificação da verificação que foram passados para o retorno de chamada. Quando onVerificationCompleted é chamado, o objeto PhoneAuthCredential é recebido diretamente, para criar o objeto

PhoneAuthCredential basta apenas chamar o método PhoneAuthProvider.getCredential, necessitando apenas passar o objeto PhoneAuthCredential para FirebaseAuth.signInWithCredential concluindo o fluxo de login.

```
PhoneAuthProvider.getInstance().verifyPhoneNumber(
        phoneNumber,        // Phone number to verify
        60,                 // Timeout duration
        TimeUnit.SECONDS,   // Unit of timeout
        this,               // Activity (for callback binding)
        mCallbacks);        // OnVerificationStateChangedCallbacks
```

Figura 13 - Método verifyNumberPhone. Fonte: (Google Developers, 2017)

Para desconectar um usuário nos métodos de login com o Google, Facebook e como um número de telefone, basta apenas invocar o método signOut, FirebaseAuth.getInstance(). signOut().

Firebase Realtime Database

Firebase Realtime Database é um banco de dados NoSQL hospedado na nuvem, que permite o armazenamento dos dados JSON sincronizados, onde todos os clientes compartilham uma instância em tempo real, recebendo atualizações dos dados automaticamente, permanecendo disponível mesmo quando a aplicação está sem conexão com a internet, pois os dados são mantidos localmente, e mesmo off-line os eventos em tempo real continuam sendo acionados. Quando a conexão é recuperada os dados locais são sincronizados com os dados atualizados remotamente, onde o gerenciamento dos conflitos é realizado automaticamente. Por meio da integração com o Firebase Authentication, é possível definir quem tem acesso aos dados e como esses dados podem ser acessados.

O padrão do Firebase Realtime Database para ler e escrever no banco de dados é restrito para que somente usuários autenticados possam executar essas ações.

Para escrever no banco de dados é necessário recuperar a instância do banco de dados usando o método getInstance e referenciar o local que deseja gravar.

```
// Write a message to the database
FirebaseDatabase database = FirebaseDatabase.getInstance();
DatabaseReference myRef = database.getReference("message");

myRef.setValue("Hello, World!");
```

Figura 14 - Escrevendo no Firebase Realtime Database. Fonte: (Google Developers, 2017)

Para ler no banco de dados, além de permissão é necessário adicionar um ValueEventListener para a referência recém criada. O método onDataChange() é acionado quando o ouvinte está ativado e a cada alteração dos dados realizada pelos os usuários.

O Firebase permite que a aplicação continue funcionando mesmo que a conexão com a internet é interrompida, quando a persistência em disco está ativada, os dados são armazenados localmente no dispositivo para que possa manter o estado enquanto estiver off-line, mesmo que o sistema operacional seja reiniciado. Sendo possível ativar a persistência local com a seguinte linha:

FirebaseDatabase.getInstance().setPersistenceEnabled(true);

O padrão de capacidade de armazenamento local é de 10 MB de dados previamente sincronizados, quando o cache ultrapassar esse tamanho, o Firebase Realtime Database limpa os dados que foram menos usados recentemente, exceto os dados sincronizados.

Upload de arquivos – Google Cloud Storage

Com o objetivo de oferecer maior credibilidade das vagas e rampas de acesso adicionadas, o aplicativo possui uma seção onde os usuários podem registrar imagens das vagas e rampas, podendo denunciar irregularidades, como a falta de manutenção e sinalização, falta de segurança, ou simplesmente informar aos outros usuários que a vaga ou rampa possui boas condições de uso.

Portanto foi utilizado o recurso de cloud storage do Firebase, que oferece recursos para o armazenamento de conteúdo gerado pelos usuários permitindo o

acesso por meio do Firebase e do Google Cloud, realizando upload e o download deles a partir de clientes móveis utilizando apenas os SDKs do Firebase.

Para utilizar o Cloud Storage é necessário adicionar a dependência “compile 'com.google.firebase:firebase-storage:11.4.0'” ao arquivo build.gradle, e criar uma instância do Firebase Storage.

Os arquivos são armazenados em um intervalo do Google Cloud Storage, apresentados em estrutura hierárquica, assim como no sistema de arquivos de um disco rígido local ou no Firebase Realtime Database.

Para a aplicação poder acessar um arquivo, uma referência é criada, que é utilizada para realizar upload ou download de dados, receber ou atualizar metadados ou excluir o arquivo.

Para criar uma referência basta invocar o método getReference através da instância Firebase Storage.

Os métodos putBytes(), putFile() ou putStream() são os responsáveis por realizar upload do arquivo no Cloud Storage.

É possível utilizar listeners para monitorar as atividades de êxitos, falhas, progresso ou pausas durante o processo de upload através dos seguintes métodos:

OnProgressListener, invocado periodicamente durante à transferência de dados.

OnPausedListener, invocado sempre que a tarefa está pausada.

OnSuccessListener, invocado na conclusão da tarefa.

OnFailureListener, invocado sempre que o upload falhar.

O processo de upload continua sendo executado em segundo plano após mudanças no ciclo de vida de atividades, comportamento que pode causar erros. Sendo necessário implementar os listeners no escopo da atividade cancelando o processo automaticamente quando a atividade for interrompida. Bastando chamar o método getActiveUploadTasks assim que a atividade é reiniciada para receber os processos de upload que ainda estiverem sendo executadas.

Considerações Finais

Dado que o principal objetivo deste projeto é auxiliar e facilitar a locomoção de idosos, deficientes físicos e pessoas com mobilidade reduzida temporária por meio da

colaboração entre os usuários, onde os mesmos poderão contribuir enviando novas vagas e rampas de acesso não cadastradas, contribuindo para tornar a base de dados mais completa e atualizada, por meio de uma aplicação para dispositivos móveis, pode ser concluir que o objetivo foi alcançado.

Pois o protótipo desenvolvido utilizando recursos para o desenvolvimento de aplicativos para dispositivos móveis disponibilizados pelo Google, como o Google Maps Api, e o Firebase, aliada a uma interface amigável, comprovou ser uma ferramenta útil no cotidiano das pessoas, permitindo inserir e localizar os pontos desejados de forma clara e intuitiva.

Os recursos utilizados no desenvolvimento demonstraram grande eficiência e eficácia durante o desenvolvimento, pois oferecem interfaces simples e intuitivas, reduzindo o tempo e a complexidade no desenvolvimento de recursos que demandariam grandes custos no seu desenvolvimento, como segurança da aplicação e dos usuários e manter a integridade e disponibilidade dos dados em tempo real.

Como desenvolvimento futuro pode – se sugerir a inserção de comando de voz, melhorando a acessibilidade, e a integração com sistemas públicos, onde as autoridades responsáveis por disponibilizar esses recursos poderiam obter dados mais precisos de como eles estão sendo distribuídos pelas cidades, e quais regiões necessitam de manutenção e/ou implantação de novas vagas e rampas de acesso.

REFERÊNCIAS

FACEBOOK. **Login do Facebook para Android**. Disponível em: <https://developers.facebook.com/docs/facebook-login/android>. Acesso em: 08 set. 2017.

FIREBASE. **Autenticar com o Firebase no Android usando um número de telefone**. Disponível em: <https://firebase.google.com/docs/auth/android/phone-auth?hl=pt-br>. Acesso em: 10 set. 2017.

FIREBASE. **Autenticar usando o Login do Google no Android**. Disponível em: <https://firebase.google.com/docs/auth/android/google-signin?hl=pt-br>. Acesso em: 11 set. 2017.

FIREBASE. **Set up Firebase Realtime Database for Android**. Disponível em: <https://firebase.google.com/docs/database/android/start/?hl=pt-br>. Acesso em: 16 set. 2017.

FIREBASE. **Como ativar recursos off-line no Android**. Disponível em: <https://firebase.google.com/docs/database/android/offline-capabilities?hl=pt-br>. Acesso em: 08 out. 2017.

FIREBASE. **Add Firebase to your android project**. Disponível em: <https://firebase.google.com/docs/android/setup>. Acesso em: 05 ago. 2017.

FIREBASE. **Autenticar usando o login do Facebook no Android**. Disponível em: < https://firebase.google.com/docs/auth/android/facebook-login?hl=pt-br>. Acesso em: 13 ago. 2017.

GOOGLE MAPS APIS. **O melhor do Google Maps para cada aplicativo para android.** Disponível em: <https://developers.google.com/maps/documentation/android-api/?hl=pt-br>. Acesso em: 02 out. 2017.

GOOGLE MAPS APIS. **Configuração do Projeto.** Disponível em: <https://developers.google.com/maps/documentation/android-api/config?hl=pt-br>. Acesso em 02 set. 2017.

GLAUBER, Nelson. **Dominando o Android: Do básico ao avançado.** 2. ed. São Paulo: Novatec, 2015. 948 p.

LECHETA, R. Ricardo. **Google Android: Aprenda a criar aplicações para dispositivos móveis com o android SDK.** 5 ed. São Paulo: Novatec, 2015. 1068 p.

MONTEIRO, B. João. **Google android: Crie aplicações para celulares e tabletes.** 1 ed. São Paulo: Casa do Código, 2012. 312 p.

PLANALTO. **LEI Nº 10.098, DE 19 DE DEZEMBRO DE 2000.**. Disponível em: < http://www.planalto.gov.br/ccivil_03/leis/l10098.htm>. Acesso em: 03 ago. 2017.