FUNDAÇÃO ESCOLA DE COMÉRCIO ÁLVARES PENTEADO - FECAP

Bacharelado em Ciência da Computação

POO e Estrutura de Dados – Professora Katia Milani Lara Bossi

Estudantes:

- André Gregório dos Santos – RA: 24026489
- Guilherme Reis Fogolin de Godoy – RA: 24026241
- Pedro Henrique Nascimento Lemos – RA: 23025380
- Yan Ramos Cezareto – RA: 24026005

Turma: 3NACOMP_S

Entrega 03 para POO e Estrutura de Dados

Objetivo: Descrever as estruturas de dados implementadas no projeto Uber Report.

01. Classe **ProcurarCorridaPassageiroActivity.java** no método **configurarAutoComplete**.

```
private void configurarAutoComplete() {
    EditText edtDestino = findViewById(R.id.input_destino);

    edtDestino.setOnClickListener( View v -> {
        List<Place.Field> campos = Arrays.asList(Place.Field.ID, Place.Field.NAME, Place.Field.LAT_LNG);
        Intent intent = new Autocomplete.IntentBuilder(AutocompleteActivityMode.OVERLAY, campos)
                .build( context: ProcurarCorridaPassageiroActivity.this);

        startActivityForResult(intent,  requestCode: 100);
    });
}
```

Nesse método, a estrutura de dados utilizada é uma **lista (List)** para armazenar os campos que serão retornados pela API de Autocomplete do Google Places. Essa lista é

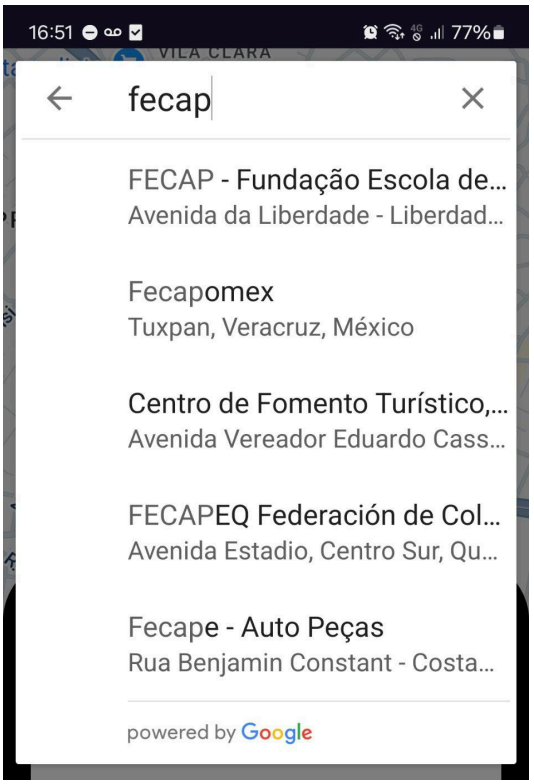
criada com o **método Arrays.asList** e contém os campos Place.Field.ID, Place.Field.NAME e Place.Field.LAT_LNG.

- **List<Place.Field>:** É uma lista imutável que armazena os campos necessários para o Autocomplete.

Campos incluídos:

- **Place.Field.ID:** Identificador único do local.
- **Place.Field.NAME:** Nome do local.
- **Place.Field.LAT_LNG:** Coordenadas geográficas do local.

Dessa forma, os usuários do aplicativo quando foram buscar locais de destino em uma corrida da Uber receberão uma lista de endereços que se assemelham aos termos que foram digitados no campo de pesquisa. No exemplo abaixo foi digitado "fecap" e, assim, são retornados algumas sugestões como a faculdade FECAP ou mesmo uma auto peças. Posteriormente ao selecionar o endereço que precisa, as coordenadas geográficas do local são retornadas também.



02. Classe **ObterAlertaImpl.java** no método **obterAlertasProximos.**

```
public void obterAlertasProximos(double latitude, double longitude, double raio, Callback<List<Alerta>> callback) {
    if (GeoUtils.calcularDistancia(ultimaLatitude, ultimaLongitude, latitude, longitude) < DISTANCIA_MINIMA) {
        Log.d( tag: "ObterAlertaImpl",  msg: "Distância mínima não atingida. Usando cache.");
        callback.onResponse( call: null, Response.success(cacheAlertas));
        return;
    }
```

A estrutura de dados **ArrayList** (que implementa a **interface List**) está sendo usada nesse método como uma forma de armazenar e manipular dinamicamente uma lista de objetos do tipo Alerta, de acordo com a resposta da API (com base no Retrofit).

A variável **cacheAlertas** é que armazena uma **lista de Alertas** previamente obtida. Ela evita chamadas repetidas à API se a nova localização estiver muito próxima da anterior.

Complementando, o objeto **Alerta** é composto dos seguintes atributos:

```
public class Alerta {
    7 usages
    private String nomeAlerta;
    7 usages
    private String tipoAlerta;
    7 usages
    private String dataHoraAlerta;
    7 usages
    private Double latitude;
    7 usages
    private Double longitude;
    7 usages
    private Integer fk_idUser;
```