POLITÉCNICO DE LEIRIA

Curso de TeSP em
Programação de Sistemas de Informação
(AMSI)

Ano Letivo 2025/2026 **2º Ano, 1º Semestre**

AMSI

Boleias

Relatório de AMSI

Nº 2231439	Ivan Santos
Nº 2231448	Tiago Bernardino
Nº 2231445	André Inácio

Introdução do documento

Contextualização

A aplicação trata-se de uma plataforma de boleias, permitindo que utilizadores possam tanto oferecer como reservar viagens partilhadas. O objetivo principal é facilitar a ligação entre condutores e passageiros, proporcionando uma experiência prática de reserva de boleias, gestão de wishlist e avaliação de serviços(comentários sobre o condutor). A aplicação distingue-se pelo seu sistema de reembolso proporcional, que ajusta o valor pago pelo passageiro consoante o número de pessoas que reservam a mesma boleia, incentivando a partilha de viagens e promovendo a eficiência do transporte. As operações CRUD que irão ser necessárias para android são a autenticação, as viaturas, as boleias, as reservas, a wishlist e a avaliação do condutor.

Requisitos finais implementados

- Consulta do feed de boleias ativas.
- Adição de boleias a favoritos (watchlist).
- Reserva de boleias e visualização de lista de reservas com cálculo de reembolso.
- Avaliação de condutores após a viagem.
- Consulta de avaliações de condutores.

Motivação

A motivação surge da necessidade de encontrar alternativas de transporte que sejam, simultaneamente, mais económicas e sustentáveis. Num contexto onde os custos do combustível e manutenção de veiculos sao elevados, especialmente para a comunidade estudantil, esta aplicação apresenta-se como uma “solução”. Fora isso a motivação também vem pela a aprendizagem ao longo do desenvolvimento deste projeto.

Descrição do desenvolvimento

Metodologia, procedimento, regras, ect…

A metodologia adotada pelo grupo baseou-se num modelo de trabalho colaborativo, usando apenas o GitHub, onde não existiu uma divisão estrita e isolada de tarefas, pelo menos do meio do percurso até ao fim. Em vez disso, o desenvolvimento foi feito de forma conjunta, em comunicação para a resolução de problemas e tomada de decisões.

As tarefas foram distribuídas de forma dinâmica à medida que as necessidades surgiam, garantindo que todos os membros participassem nas várias vertentes do projeto.

A implementação do código e API foram feitas em conjunto através de sessões de trabalho e partilhas de código.

Tecnologias usadas/pesquisadas

O desenvolvimento desta aplicação exigiu a utilização e a pesquisa de diversas ferramentas.

Nomeadamente, Android Studio, Java, Postman, Gemini, PHP(Projeto WEB), JSON, entre outras.

SOLUÇÃO DESENVOLVIDA

Descrição dos problemas/dificuldades e soluções

Gestão de Interface conforme o utilizador

Numa questão de estética e usabilidade, um dos principais desafios foi a adaptação dinâmica da UI consoante o perfil do user logado. Para garantir uma experiência intuitiva e segura, foi necessário implementar uma lógica de visibilidade seletiva, onde os botões e funcionalidades são ocultados ou bloqueados, apesar de as respostas da API já validarem isso.

Justificação das soluções implementadas

```
public static final String NOME = "nome";  4 usages
public static final String TOKEN = "token";  4 usages
public static final String CONDUTOR = "condutor";  4 usages
public static final String PERFIL_ID = "perfil_id";  4 usages
```

```
navigationView.getMenu().clear();
if ("1".equals(condutor)) {
    navigationView.inflateMenu(R.menu.menu_main_condutor);
} else {
    navigationView.inflateMenu(R.menu.menu_main_passageiro);
}

View headerView = navigationView.getHeaderView( index: 0);
if (headerView != null) {
    TextView nav_tvNome = headerView.findViewById(R.id.tvNome);
    nav_tvNome.setText(nome);
}
```

```
@Override  2 usages
public void onRefreshListaViaturas(ArrayList<Viatura> listaViaturas) {

    this.viaturasDisp = listaViaturas;
    detalhesBoleiaFragment();
    invalidateOptionsMenu();

    if ("1".equals(condutor) && (viaturasDisp == null || viaturasDisp.isEmpty())) {
        Toast.makeText( context: this,  text: "Crie uma viatura primeiro!", Toast.LENGTH_LONG).show();
        finish();
        return;
    }
```

```
if (idBoleia != -1) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.menu_remover, menu);

    MenuItem itemRemover = menu.findItem(R.id.itemRemover);
    if (itemRemover != null && !"1".equals(condutor)) {
        itemRemover.setVisible(false);
    }
    return true;
}
return super.onCreateOptionsMenu(menu);
```

```
if (condutorIsDono()) {
    setInputsEnabled(true);
    fabVerReservas.show();
    fabGuardar.show();
    fabReservar.hide();
    fabAddWishlist.hide();
    fabGuardar.setImageResource(R.drawable.ic_action_guardar);
    fabVerReservas.setImageResource(R.drawable.ic_action_ver_reservas);
    setFabGuardar();
    setFabVerReservas();

} else {
    setInputsEnabled(false);
    fabReservar.show();
    fabAddWishlist.show();
    fabGuardar.hide();
    fabVerReservas.hide();


    fabReservar.setImageResource(R.drawable.ic_action_reservar);
    fabAddWishlist.setImageResource(R.drawable.ic_action_add);
    setFabReservar();
    setFabAddWishlist();
}
```

Bloqueio da UI em pedidos POST

Observou-se que a app bloqueava temporariamente ao realizar pedidos POST.

Para resolver este problema, a solução passou pela implementação de 'new Thread' no momento em que se guarda dados em SQLite, por exemplo no 'adicionarViaturaBD'

Justificação das soluções implementadas

```
public void adicionarReservaBD(Reserva reserva){  1 usage

    if(reserva == null) return;

    new Thread(new Runnable() {
        @Override
        public void run() {
            Reserva auxReserva = boleiasBD.adicionarReservaBD(reserva);
            if (auxReserva != null){
                reservas.add(auxReserva);
            }
        }
    }).start();

}
```

```
public void adicionarBoleiaBD(Boleia boleia){  1 usage

    if(boleia == null) return;

    new Thread(new Runnable() {
        @Override
        public void run() {
            Boleia auxBoleia = boleiasBD.adicionarBoleiaBD(boleia);
            if (auxBoleia != null){
                boleias.add(auxBoleia);
            }
        }
    }).start();

}
```

Validação de estados de negócio

As boleias que estivessem fechadas (cujo as reservas são validadas pelo condutor) não podiam ter interação de clique no feed. Mais uma vez, uma questão de estética, pois a API mesmo que se o utilizador tentasse reservar outra vez a boleia que estivesse fechada, a própria API dava uma resposta de erro.

Justificação das soluções implementadas

```
public void onResponse(JSONObject response) {
    try {

        JSONArray jsonArray = response.getJSONArray( name: "data");
        JSONArray jsonArrayFechadas = response.getJSONArray( name: "boleias_fechadas");


        boleias = JSONParser.parserJsonBoleias(jsonArray, fechada: 0);
        boleiasFechadas = JSONParser.parserJsonBoleias(jsonArrayFechadas, fechada: 1);

        java.util.HashMap<Integer, Boleia> mapaBoleias = new java.util.HashMap<>();

        for (Boleia b : boleias) mapaBoleias.put(b.getId(), b);

        for (Boleia f : boleiasFechadas) mapaBoleias.put(f.getId(), f);

        boleias.clear();
        boleias.addAll(mapaBoleias.values());


        adicionarBoleiasBD(boleias);

        if (boleiasListener != null) {
            boleiasListener.onRefreshListaBoleias(boleias);
        }
```

```
public int getIsFechada() {  3 usages
    return isFechada;
}

public void setIsFechada(int isFechada) {  1 usage
    this.isFechada = isFechada;
}
```

```
lvBoleias.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {

        Boleia boleiaSelecionada = (Boleia) parent.getItemAtPosition(position);

        if (boleiaSelecionada != null && boleiaSelecionada.getIsFechada() == 1) {
            Toast.makeText(getContext(), text: "Esta boleia está fechada e não pode ser acedida.", Toast.LENGTH_SHORT).show();
        } else {

            Intent intent = new Intent(getContext(), DetalhesBoleiaActivity.class);
            intent.putExtra(DetalhesBoleiaActivity.BOLEIA_ID, (int) id);
            startActivityForResult(intent, MenuMainActivity.EDIT);
        }
    }
});
```

Identificação das funcionalidades propostas, mas não implementadas;

Sistema de notificações via MQTT/Mosquitto

Estava prevista notificações para alertas de boleias novas criadas e atualizadas, avaliações ect.. dos quais foram implementadas na UC de SIS mas que não foram ainda implementadas em Android.

Conclusões

Descrição dos resultados obtidos

Os resultados obtidos são positivos na minha opinião, os objetivos foram cumpridos, assim como os requisitos da aplicação, resultando num sistema funcional que permite o ciclo completo da aplicação.

Tem estabilidade nas operações CRUD e a integração entre Android e a API foi bem sucedida, garantindo a persistência de dados e uma interface responsiva.