VotoInformado
Grupo 3

Alexandre Santos, Bruno Correia, Daniel Carlos, Henrique Laia, Vasco Colaço

Projeto Coletivo de Programação em Dispositivos Móveis
Engenharia Informática

Orientador: Prof. Doutor Paulo Fazendeiro

Novembro de 2025

Resumo

A aplicação VotoInformado foi desenvolvida com o objetivo de fornecer aos eleitores informação fiável e acessível sobre candidatos, partidos, programas eleitorais e eventos políticos. Os objetivos centrais do projeto são centralizar a informação política, facilitar a comparação entre candidatos e aumentar o nível de literacia eleitoral da população. A solução consiste numa aplicação Android que inclui funcionalidades como listas de candidatos filtráveis, páginas de partidos , secção de notícias, sondagens, um questionário de posicionamento político e um sistema de criação de petições pelos utilizadores, suportado por Firebase.

Palavras-chave

VotoInformado, Aplicação Móvel, Android, Literacia Política, Eleições, Firebase

Contents

List of Figures

List of Tables

List of Algorithms

UBI	Universidade da Beira Interior
MPSOCD	Multi-objective Particle Swarm Optimization Crowding Distance

Chapter 1

Introdução

1.1 Enquadramento e Motivação

O projeto **VotoInformado** surge no âmbito da unidade curricular de Programação de Dispositivos Móveis, com o intuito de combater a desinformação e o alheamento político. Numa era em que a informação é abundante mas nem sempre fidedigna, torna-se crucial fornecer aos cidadãos ferramentas que centralizem dados oficiais e notícias relevantes sobre o panorama político nacional.

A aplicação visa simplificar o acesso a informações sobre candidatos, partidos, sondagens e eventos eleitorais, promovendo uma participação cívica mais consciente e informada.

1.2 Objetivos

Os principais objetivos deste projeto são:

- Desenvolver uma aplicação Android nativa, robusta e intuitiva.
- Centralizar informações sobre candidatos e partidos políticos.
- Disponibilizar sondagens eleitorais com visualização gráfica de dados.
- Fornecer um feed de notícias políticas atualizado em tempo real.
- Implementar um calendário de datas importantes para o processo eleitoral.

1.3 Estrutura do Relatório

Este relatório está organizado da seguinte forma:

- O **Capítulo 2** descreve a arquitetura do sistema e as tecnologias utilizadas.
- O **Capítulo 3** detalha as funcionalidades implementadas na aplicação.
- O **Capítulo 4** aborda aspetos técnicos da implementação e desafios encontrados.
- O **Capítulo 5** apresenta os resultados obtidos e a validação da solução.

- O **Capítulo 6** apresenta as conclusões e sugestões para trabalho futuro.

Chapter 2

Arquitetura e Tecnologias

Este capítulo descreve a arquitetura de software adotada para o desenvolvimento da aplicação VotoInformado, bem como as tecnologias e bibliotecas utilizadas.

2.1 Arquitetura do Sistema

A aplicação foi desenvolvida para o sistema operativo Android, utilizando a linguagem Java. A estrutura do projeto segue os padrões recomendados pela Google, organizando o código em componentes lógicos para facilitar a manutenção e escalabilidade.

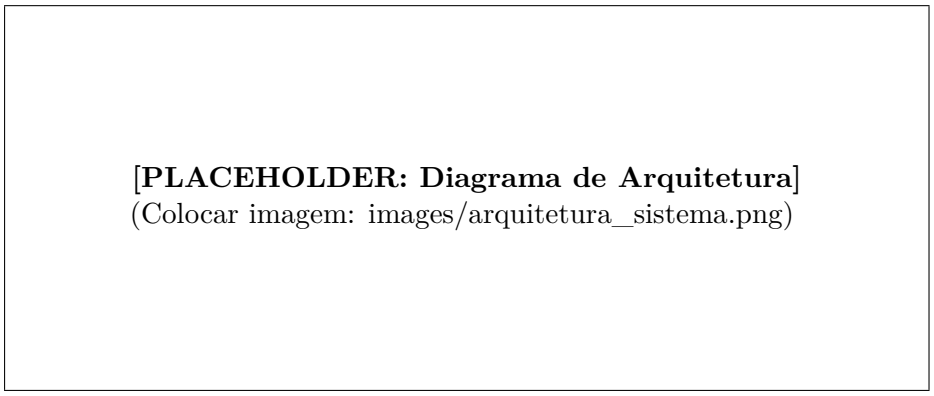

Figure 2.1: Arquitetura de alto nível do sistema VotoInformado.

A arquitetura baseia-se na separação de responsabilidades:

- **Activities e Fragments**: Responsáveis pela interface de utilizador (UI) e interação com o utilizador. Exemplos incluem `HomeActivity`, `NoticiasFragment` e `CandidatosFragment`.
- **Adapters**: Fazem a ponte entre os dados e os componentes de visualização de listas (`RecyclerView`).
- **Modelos (Classes)**: Representam as entidades de dados, como `Candidato`, `Noticia` e `Sondagem`.
- **Utils/Helpers**: Classes utilitárias para acesso a dados e operações comuns.

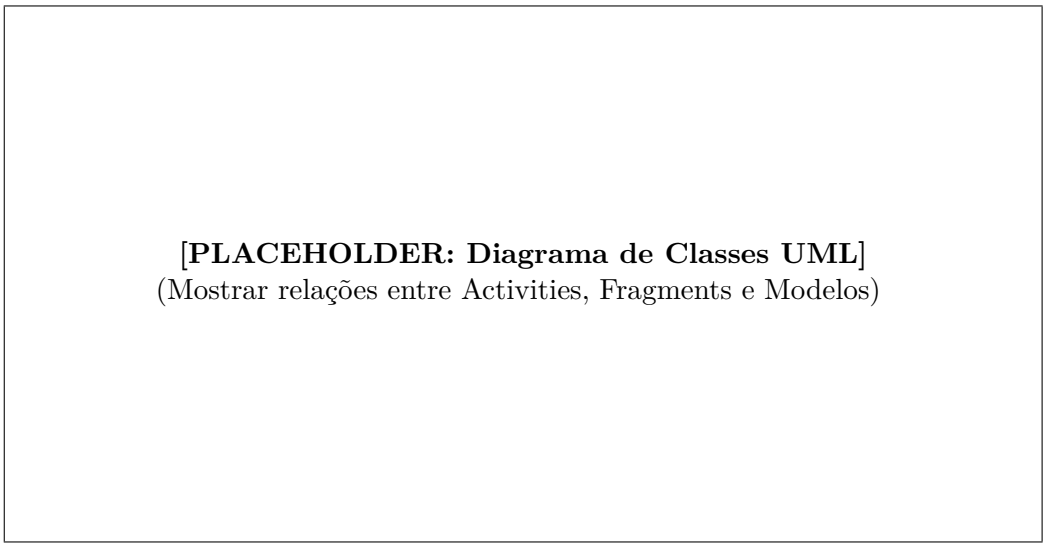

Figure 2.2: Diagrama de Classes simplificado da aplicação.

2.2 Tecnologias Utilizadas

2.2.1 Firebase

O backend da aplicação é suportado inteiramente pela plataforma Firebase, tirando partido dos seus serviços *serverless*:

- **Firebase Authentication**: Gere a autenticação de utilizadores, suportando login por email/password e integração com a conta Google.
- **Cloud Firestore**: Base de dados NoSQL flexível e escalável, utilizada para armazenar toda a informação da aplicação (candidatos, sondagens, datas).
- **Firebase Storage**: Utilizado para o armazenamento e distribuição de ficheiros multimédia, como as fotografias dos candidatos.

2.2.2 Bibliotecas Externas

Para enriquecer a funcionalidade da aplicação, foram integradas diversas bibliotecas *open-source*:

Table 2.1: Bibliotecas externas utilizadas no projeto.

Biblioteca	**Propósito**
Picasso	Carregamento e cache de imagens assíncrono.
MPAndroidChart	Criação de gráficos para visualização de sondagens.
CircleImageView	Exibição de imagens de perfil circulares.
Retrofit/Gson	(Legado) Parsing de dados JSON.

- **Picasso**: Biblioteca poderosa para o carregamento e cache de imagens em Android. Resolve problemas complexos de gestão de memória e reciclagem de vistas em listas.
- **MPAndroidChart**: Utilizada para a criação de gráficos interativos e visualmente apelativos na secção de sondagens.
- **CircleImageView**: Permite a exibição de imagens de perfil com formato circular de forma simples e eficiente.
- **Retrofit/Gson** (Legado): Inicialmente utilizadas para parsing JSON, foram substituídas pela integração direta com o Firestore, mas fizeram parte do processo de desenvolvimento.

Chapter 3

Funcionalidades

A aplicação VotoInformado oferece um conjunto de funcionalidades desenhadas para informar o eleitor.

3.1 Autenticação e Perfil

A segurança e personalização são garantidas através de um sistema de autenticação robusto.

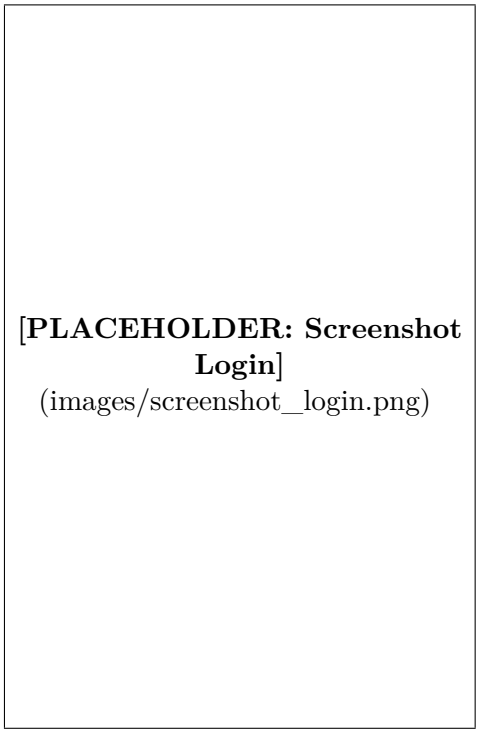

Figure 3.1: Ecrã de Login da aplicação.

- **Login e Registo**: Os utilizadores podem criar conta com email e password ou utilizar a sua conta Google para um acesso mais rápido.

- **Gestão de Sessão**: A aplicação mantém a sessão do utilizador ativa, permitindo acesso direto sem necessidade de login constante.

3.2 Requisitos Funcionais

A tabela abaixo resume as principais funcionalidades implementadas.

Table 3.1: Requisitos Funcionais do VotoInformado.

ID	Funcionalidade	Descrição
RF01	Autenticação	Permitir login e registo via Email/Password e Google.
RF02	Listar Candidatos	Visualizar lista de candidatos com foto e partido.
RF03	Detalhe Candidato	Ver biografia, propostas e cargos de um candidato.
RF04	Feed Notícias	Ler notícias políticas atualizadas via RSS.
RF05	Sondagens	Visualizar gráficos de intenção de voto.

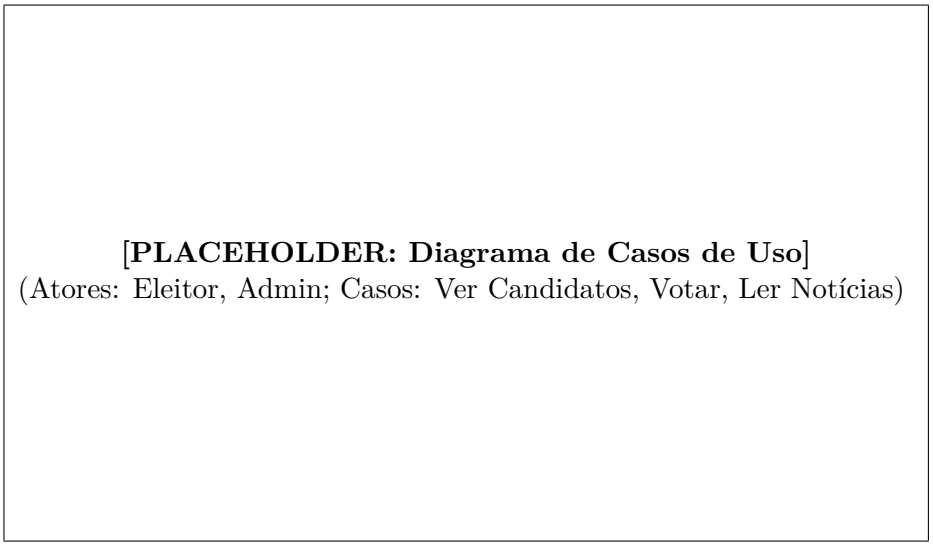

Figure 3.2: Diagrama de Casos de Uso do sistema.

3.3 Consulta de Candidatos

Esta é uma das funcionalidades centrais da aplicação.

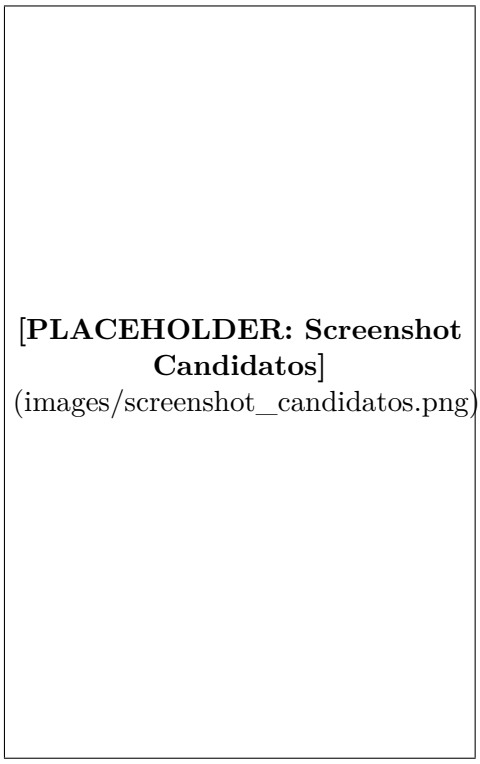

Figure 3.3: Lista de candidatos apresentada ao utilizador.

- **Listagem**: Apresenta uma lista de todos os candidatos registados na base de dados.

- **Detalhes**: Ao selecionar um candidato, o utilizador tem acesso a uma ficha detalhada que inclui:

  - Biografia e dados pessoais (idade, profissão).

  - Partido político e cargos desempenhados.

  - Lista de propostas eleitorais principais.

3.4 Notícias em Tempo Real

Para manter o utilizador atualizado sobre a atualidade política:

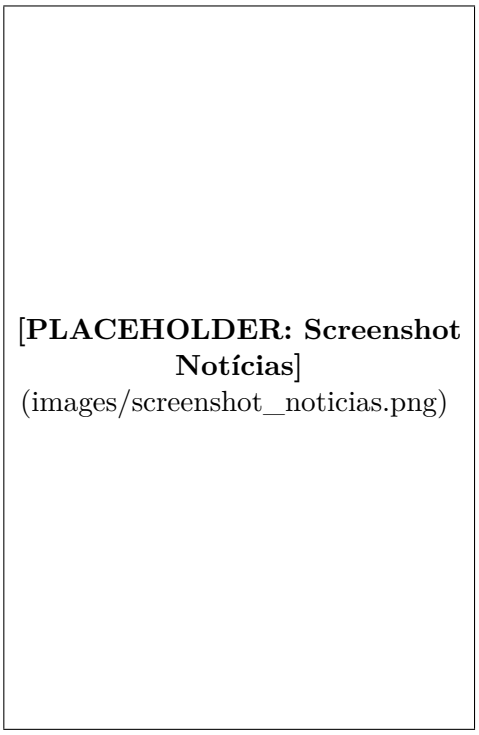

Figure 3.4: Feed de notícias políticas.

- **Feed RSS**: Integração com o feed de política da RTP Notícias.
- **Pesquisa**: Barra de pesquisa que permite filtrar notícias por palavras-chave em tempo real.
- **Visualização**: Abertura da notícia completa num navegador interno ou externo.

3.5 Sondagens e Estatísticas

A secção de sondagens permite visualizar as tendências de voto.

- **Gráficos**: Visualização clara dos resultados através de gráficos de barras.
- **Ficha Técnica**: Informação sobre a amostra, margem de erro e empresa responsável pela sondagem.

3.6 Datas Importantes

Um calendário eleitoral que lista eventos cruciais como debates, dias de reflexão e o dia das eleições, garantindo que o eleitor não perde prazos importantes.

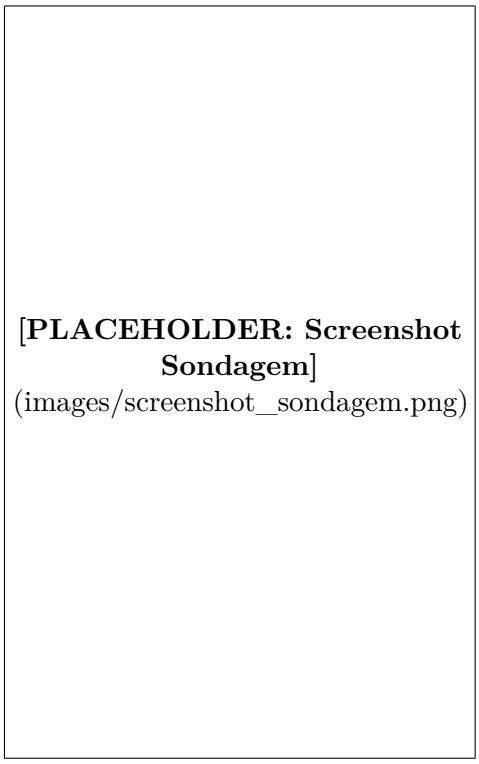

Figure 3.5: Visualização gráfica dos resultados de uma sondagem.

3.7 Bússola Política

A Bússola Política é uma ferramenta interativa que permite ao utilizador descobrir o seu posicionamento político através de um questionário.

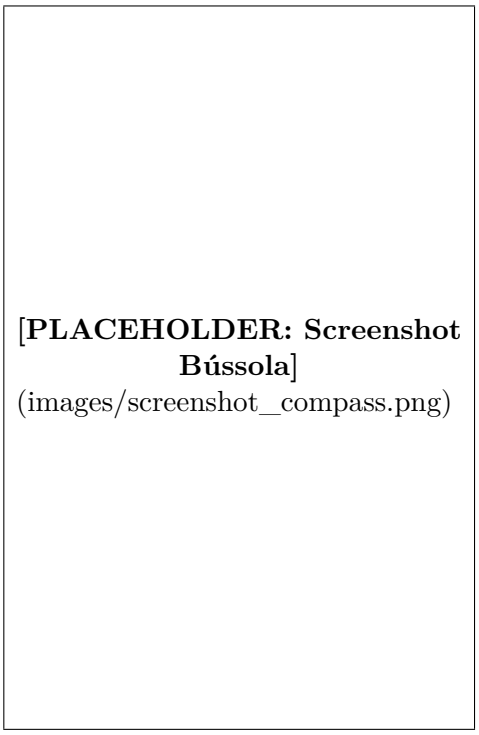

Figure 3.6: Resultado do teste da Bússola Política.

- **Questionário**: Um conjunto de perguntas sobre temas económicos e sociais, onde o utilizador expressa o seu grau de concordância.

- **Visualização**: O resultado é apresentado num gráfico bidimensional (Eixo Económico vs. Eixo Social), comparando a posição do utilizador com a de vários candidatos e partidos.

3.8 Petições Públicas

Para promover a participação cívica ativa, a aplicação inclui um sistema de petições públicas.

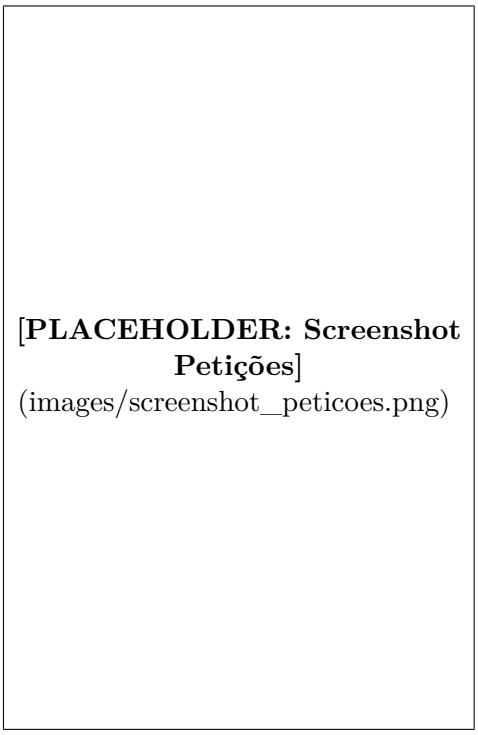

Figure 3.7: Lista de petições públicas criadas pelos utilizadores.

- **Criação**: Qualquer utilizador autenticado pode criar uma nova petição, definindo um título, uma descrição e adicionando uma **imagem ilustrativa**.
- **Assinatura**: Os utilizadores podem apoiar causas assinando petições existentes.
- **Ordenação**: A lista de petições pode ser ordenada por popularidade (mais votadas) ou por data (mais recentes), facilitando a descoberta de causas relevantes.

Chapter 4

Implementação

Este capítulo aborda os desafios técnicos e as soluções de implementação adotadas durante o desenvolvimento.

4.1 Migração para API Customizada

Inicialmente, o projeto utilizava o Firebase Firestore para persistência de dados diretamente na aplicação móvel. Para aumentar a complexidade técnica e centralizar a lógica de negócio, migrou-se para uma arquitetura com uma **API REST intermédia**.

Esta nova arquitetura consiste numa API desenvolvida em **Node.js**, que atua como *back-end* para a aplicação Android. A API utiliza uma base de dados **MongoDB** para persistência de dados, comunicando através da biblioteca **Mongoose**. A autenticação é gerida através de **JWT (JSON Web Tokens)**, garantindo uma sessão segura e independente do estado do servidor.

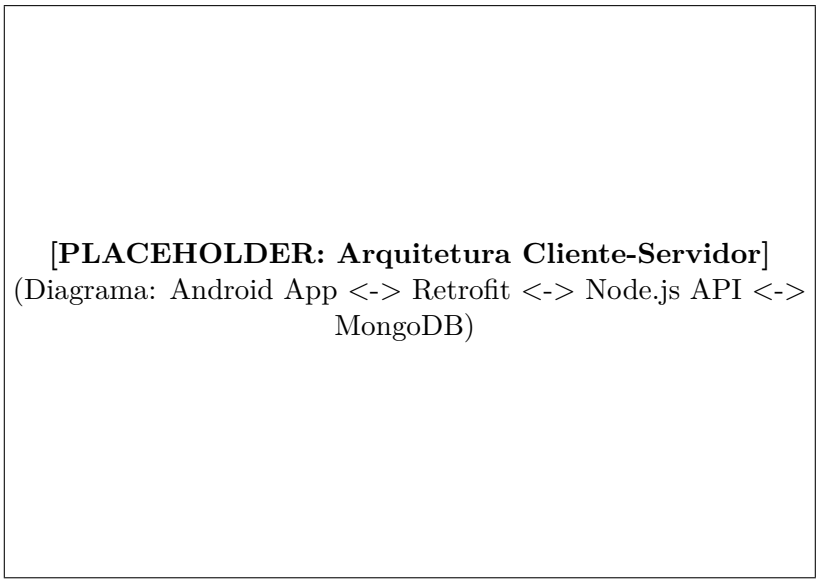

Figure 4.1: Arquitetura da comunicação com a API customizada.

```
1 public interface ApiService {
2     @GET("api/candidates")
3     Call<List<Candidato>> getCandidates();
4
```

```
5       @POST("api/petitions")
6       Call<Peticao> createPetition(@Body Peticao peticao);
7
8       @Multipart
9       @POST("api/petitions/upload")
10      Call<Map<String, String>> uploadPetitionImage(@Part MultipartBody.Part
            image);
11  }
```

Listing 4.1: Definição da Interface Retrofit

4.2 Parsing de Notícias (RSS)

Para obter as notícias, foi implementada a classe `NoticiasFetcher`. Esta classe realiza uma requisição HTTP ao feed RSS da RTP e faz o parsing do XML resultante utilizando `DocumentBuilder`. Um desafio interessante foi a extração de imagens, que não vinham num campo explícito, mas sim embutidas na descrição HTML. Foi utilizada uma expressão regular (Regex) para extrair o atributo `src` das tags `<img>`.

[PLACEHOLDER: Diagrama de Sequência]
(Fluxo: Fragment -> Fetcher -> HTTP Request -> XML Parsing -> Return List)

Figure 4.2: Diagrama de sequência do processo de obtenção de notícias.

4.3 Gestão de Imagens

As imagens das petições e dos candidatos são armazenadas localmente no servidor (*back-end*) e servidas como ficheiros estáticos. A base de dados armazena apenas o URL relativo da imagem. Na aplicação Android, a biblioteca **Picasso** é utilizada para carregar e exibir estas imagens de forma assíncrona, gerindo automaticamente o download, cache e

redimensionamento das imagens.

4.4 Adapters e RecyclerViews

A exibição de listas eficientes foi conseguida através da implementação de `RecyclerView.Adapter` personalizados. O padrão *ViewHolder* é utilizado para reciclar as vistas, garantindo uma rolagem suave mesmo com listas longas de candidatos ou notícias.

4.5 Motor de Quiz e Visualização Gráfica

A funcionalidade da Bússola Política envolveu o desenvolvimento de um motor de cálculo de pontuação e uma vista personalizada para o gráfico.

4.5.1 Modelo de Dados

A classe `Question` encapsula o texto da pergunta, o peso e o eixo a que pertence (Económico ou Social). A classe `CompassCandidate` estende a informação base de um candidato adicionando as coordenadas (x, y) da sua posição política estimada.

4.5.2 CompassView

Para desenhar o gráfico, foi criada a classe `CompassView` que estende `View`.

- **onDraw()**: Este método é sobrescrito para desenhar os eixos, as quadrículas de fundo, e os pontos que representam o utilizador e os candidatos.
- **Transformação de Coordenadas**: As coordenadas lógicas (de -10 a +10) são convertidas para coordenadas de ecrã (pixels) tendo em conta a largura e altura da vista, garantindo que o gráfico se adapta a diferentes tamanhos de ecrã.

4.6 Sistema de Petições

O sistema de petições foi adaptado para consumir a nova API, mantendo a autenticação de utilizadores via Firebase Auth (gerida pelo backend).

4.6.1 Estrutura de Dados

A classe `Peticao` foi atualizada para mapear os documentos MongoDB.

```java
public class Peticao {
    @SerializedName("_id")
    private String id;
    private String titulo;
    private String descricao;
    private List<String> assinaturas; // Lista de UIDs
    private String imageUrl; // URL da imagem
    // ... getters e setters
}
```

Listing 4.2: Estrutura simplificada da classe Peticao

4.6.2 Upload de Imagens

Para permitir o upload de imagens nas petições sem custos adicionais de serviços cloud, implementou-se um sistema de **armazenamento local** no servidor.

- **Backend**: Utiliza o middleware `multer` para receber ficheiros *multipart/form-data* e guardá-los numa pasta local `uploads/`. O servidor serve estes ficheiros estaticamente.
- **Android**: A atividade `CreatePeticaoActivity` seleciona a imagem da galeria, converte-a num ficheiro temporário e envia-a para o endpoint `/api/petitions/upload` antes de submeter os dados da petição.

4.6.3 Eliminação de Petições

Foi implementada a funcionalidade de eliminação de petições através do método HTTP `DELETE`. O endpoint `/api/petitions/:id` permite remover documentos da coleção `petitions` no Firestore, garantindo a integridade dos dados. A aplicação Android expõe esta funcionalidade através da classe utilitária `DatabaseHelper`.

4.7 Resolução de Problemas e Otimizações

Durante a fase final de testes e integração, foram identificados e resolvidos três problemas críticos que afetavam a experiência do utilizador e a estabilidade da aplicação.

4.7.1 Ligação de Candidatos em Sondagens

Foi detetada uma inconsistência na ligação entre os resultados das sondagens e os perfis dos candidatos. O problema devia-se a uma discrepância nos identificadores: a base de dados

utilizava IDs do MongoDB (`_id`), enquanto alguns resultados de sondagens referenciavam candidatos por um ID textual (e.g., "andre_pestana").

Solução: Implementou-se uma lógica de correspondência robusta nos adaptadores (`ResultadoSondagemA` `SondagemAdapter`) e no `HomeFragment`. O algoritmo tenta agora corresponder o candidato sequencialmente por:

1. ID do MongoDB (`_id`).
2. ID textual personalizado (`id`).
3. Nome do candidato (como recurso final, insensível a maiúsculas/minúsculas).

4.7.2 Conectividade com API (Render Cold Start)

A API, alojada no plano gratuito do serviço Render, entra em modo de suspensão após períodos de inatividade. O tempo de arranque inicial ("cold start") excedia frequentemente o timeout padrão de 1.5 segundos definido no cliente Android, levando a aplicação a falhar silenciosamente para o URL local (`localhost`).

Solução: O timeout de conexão e leitura no `ApiClient` foi aumentado para **15 segundos**. Isto garante que a aplicação aguarda o tempo suficiente para que o servidor acorde, assegurando uma conexão bem-sucedida mesmo após inatividade.

4.7.3 Carregamento de Imagens de Perfil

Identificou-se que as imagens de perfil dos utilizadores não eram carregadas corretamente em dispositivos físicos. A causa raiz era dupla: o backend guardava URLs absolutos (e.g., `http://localhost:3000/...`) durante o registo, que são inacessíveis externamente, e os comentários não persistiam a foto do autor.

Solução:

- **Backend**: O controlador de autenticação foi atualizado para guardar apenas o **caminho relativo** da imagem (e.g., `/uploads/profiles/...`). Adicionalmente, a criação de comentários foi corrigida para persistir o URL da foto do autor.
- **Frontend**: Implementou-se uma lógica de "sanitização" no cliente (`HomeFragment`, `ComentarioAdapter`). A aplicação deteta agora URLs legados contendo "localhost" ou "127.0.0.1", converte-os em caminhos relativos e prepõe dinamicamente o URL base correto da API (`ApiClient.getBaseUrl()`).

Chapter 5

Resultados e Testes

5.1 Estado Atual do Projeto

A aplicação encontra-se num estado funcional estável, com todas as funcionalidades principais implementadas e operacionais. A navegação entre ecrãs é fluida e a integração com o Firebase responde com latência reduzida.

5.2 Testes Realizados

Para garantir a qualidade do software, foram realizadas várias etapas de verificação:

5.2.1 Testes Manuais

Foram realizados testes exaustivos em emuladores e dispositivos físicos para validar:

- O fluxo de registo e login.
- A correta visualização dos dados dos candidatos.
- A resiliência da aplicação a falhas de rede (tratamento de erros no carregamento de notícias).
- A adaptação da interface ao Modo Escuro.

Table 5.1: Exemplo de casos de teste executados.

ID	Ação	Resultado Esperado	Estado
CT01	Login com credenciais inválidas	Exibir mensagem de erro	Passou
CT02	Carregar lista de notícias	Exibir lista com títulos e imagens	Passou
CT03	Clicar em candidato	Abrir detalhe do candidato	Passou

5.2.2 Testes Unitários e Instrumentados

O projeto inclui uma estrutura para testes unitários (JUnit) e testes de interface (Espresso), permitindo a verificação automática de componentes críticos e fluxos de utilizador.

5.3 Análise de Desempenho

A utilização do *Android Profiler* permitiu identificar e corrigir fugas de memória, especialmente no carregamento de imagens, confirmando a eficácia da introdução da biblioteca Picasso.

Chapter 6

Conclusão e Trabalho Futuro

6.1 Conclusão

O projeto VotoInformado atingiu os seus objetivos principais, resultando numa aplicação móvel capaz de informar os eleitores de forma clara e acessível. A evolução da arquitetura para uma **API intermédia em Node.js** com base de dados **MongoDB**, provou ser uma decisão acertada, garantindo maior segurança, centralização da lógica e escalabilidade da solução.

A aprendizagem adquirida durante o desenvolvimento, nomeadamente na gestão de dependências, chamadas assíncronas e design de interfaces Android, foi valiosa para a equipa.

6.2 Trabalho Futuro

Apesar de funcional, a aplicação tem margem para evolução. Algumas sugestões para versões futuras incluem:

- **Notificações Push**: Alertar os utilizadores para novas sondagens ou notícias urgentes.
- **Comparador de Candidatos**: Uma ferramenta para colocar lado a lado as propostas de dois candidatos.
- **Gamificação**: Introduzir quizzes sobre política para incentivar a aprendizagem de forma lúdica.
- **Suporte Offline**: Melhorar a cache de dados para permitir a consulta básica mesmo sem ligação à internet.

Bibliography

Appendix A

Appendix

Nam placerat ullamcorper ante non venenatis. Phasellus et ipsum at lorem rhoncus euismod. Phasellus in risus elit, sed mollis dolor. Aenean non ligula ut metus porta laoreet. Duis mi quam, sollicitudin non posuere eu, facilisis vestibulum purus. Cras eget odio et diam imperdiet consectetur eu vel libero. Cras in dapibus felis. Praesent sed nunc neque. Donec lobortis venenatis pretium. Praesent quis lorem ipsum, id mattis ante.

A.1 Datasheets dos componentes utilizados

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent at magna viverra neque bibendum pellentesque. Morbi ullamcorper auctor turpis vitae mollis. Fusce elementum mauris eu magna tristique vel aliquet erat iaculis. Donec sed augue mi. Aenean commodo lorem ac nulla iaculis rhoncus. Mauris facilisis, ante in molestie bibendum, lorem augue vehicula metus, ac auctor turpis quam nec purus. Nam malesuada accumsan neque, quis vulputate nibh dapibus vitae. Vestibulum eu arcu ut est posuere malesuada. Donec aliquet, mauris vel viverra bibendum, risus sem fringilla orci, placerat laoreet felis velit ac justo. Mauris sit amet sollicitudin magna. Sed commodo enim sed nibh consectetur cursus. Duis turpis lacus, semper non facilisis eu, semper eu lacus. Donec vel urna urna, eget gravida magna.

Glossary

LaTeX	Conjunto de macros para o processador de textos TeX, utilizado amplamente para a produção de textos matemáticos e científicos devido à sua alta qualidade tipográfica.