Escola Superior de Tecnologia e Gestão

TeSP em Programação de Sistemas de Informação

Unidade Curricular: Acesso Móvel a Sistemas de Informação

VeiGest

Aplicação Android para Gestão de Frotas

Edo Pedroso dos Santos (Nº 2231440)
Leonardo Daniel Neves Pereira (Nº 2241550)
Pedro Kaleb De Jesus (Nº 2240103)

Leiria, 17 de janeiro de 2026

Conteúdo

Lista de Figuras

1 Introdução

No panorama empresarial contemporâneo, a mobilidade e o acesso imediato à informação são fatores críticos de sucesso. A gestão de frotas, em particular, exige um controlo rigoroso sobre múltiplos ativos em circulação. O projeto **VeiGest**, desenvolvido no âmbito da unidade curricular de Acesso Móvel a Sistemas de Informação (AMSI), visa dar resposta a esta necessidade através de uma solução móvel nativa para o sistema operativo Android.

A aplicação foi concebida para servir dois perfis principais de utilizadores: os gestores de frota, que necessitam de supervisionar a operação global, e os condutores, que carecem de uma ferramenta prática para consultar atribuições e reportar atividade. O sistema integra funcionalidades de gestão de veículos, planeamento de rotas, consulta de documentação legal e registo de manutenções, tudo centralizado numa única plataforma acessível via smartphone.

Este relatório detalha o processo de desenvolvimento da aplicação, desde a conceção da arquitetura até à implementação final, descrevendo as opções técnicas tomadas, as dificuldades encontradas e as soluções adotadas para cumprir os requisitos académicos e funcionais propostos.

1.1 Requisitos Finais Implementados

A aplicação final respeita integralmente o enunciado do projeto, apresentando uma estrutura robusta baseada numa `MainActivity` que orquestra a navegação entre nove fragmentos distintos. Esta arquitetura "Single Activity"promove uma experiência de utilização fluida e consistente.

Foram implementadas operações CRUD (Create, Read, Update, Delete) completas para as entidades nucleares do sistema: **Veículos** e **Rotas**. A aplicação comunica com um servidor remoto através de uma API RESTful, permitindo a sincronização bidirecional de dados. Funcionalidades adicionais incluem autenticação segura (com persistência de sessão), visualização de documentos com alertas de validade, e um sistema de relatórios capaz de gerar ficheiros PDF e enviá-los por email.

1.2 Motivação

A escolha deste tema prende-se com a vontade de aplicar os conhecimentos teóricos num cenário realista. A gestão de frotas apresenta desafios técnicos interessantes, como a necessidade de lidar com dados complexos, garantir a integridade da informação entre o dispositivo móvel e o servidor, e fornecer uma interface que seja utilizável em contexto de trabalho, muitas vezes em movimento. Um foco especial foi dado à criação do **VeiGest SDK**, uma biblioteca modular que encapsula a lógica de negócio, permitindo a sua reutilização e testes isolados.

2 Descrição do Desenvolvimento

2.1 Metodologia e Arquitetura

A abordagem ao desenvolvimento seguiu uma metodologia modular e iterativa. Reconhecendo a importância da organização do código e da separação de responsabilidades, o projeto foi dividido em dois módulos distintos e interdependentes:

1. **Módulo de Aplicação (app):** Responsável exclusivamente pela interface gráfica (UI) e interação com o utilizador. Contém as Activities, Fragments e Adapters.
2. **Módulo SDK (veigest-sdk):** Encapsula toda a lógica de negócio, modelos de dados, persistência local e comunicação com a API.

Esta separação arquitetural foi fundamental para permitir o desenvolvimento paralelo entre os membros da equipa. Enquanto parte do grupo se focava no design e usabilidade dos ecrãs, outra parte dedicava-se à robustez das comunicações de rede e gestão de base de dados.

O padrão de desenho **Singleton** foi utilizado na classe `SingletonVeiGest` para centralizar a gestão do estado da aplicação e a fila de requisições de rede. Para gerir a assincronia inerente às comunicações móveis, implementou-se o padrão **Observer** através de interfaces de *Listener*, garantindo que a UI reagisse apenas quando os dados estivessem disponíveis, evitando bloqueios na *Main Thread*.

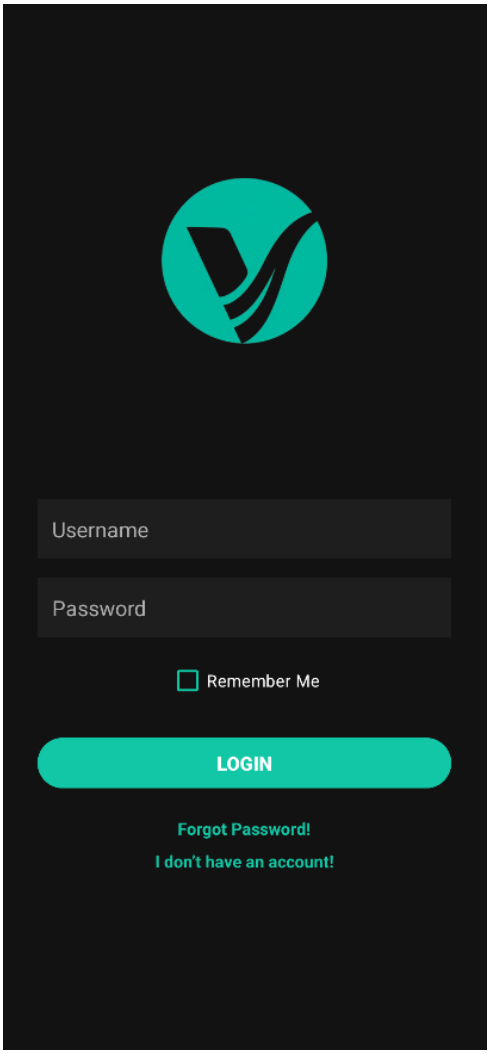

Figura 1: Ecrã de Login com autenticação centralizada no SDK.

2.2 Tecnologias Usadas

O núcleo da aplicação foi desenvolvido em linguagem **Java**, utilizando o Android Studio. A escolha tecnológica recaiu sobre bibliotecas, robustas e amplamente testadas pela comunidade:

- **Volley:** Para a camada de rede. A sua gestão automática de filas de pedidos e cache transparente torna-o ideal para aplicações intensivas em dados.
- **Glide:** Para o carregamento e cache de imagens assíncrono, essencial para a performance das listas de veículos.
- **SQLite:** Para a persistência de dados estruturados offline.
- **SharedPreferences:** Para armazenamento de configurações e tokens de sessão.

2.3 Solução Desenvolvida

A interface do utilizador segue as diretrizes do **Material Design 3**, garantindo uma experiência moderna e acessível. A navegação baseia-se num menu lateral (*Navigation Drawer*) que permite o acesso rápido a todos os módulos.

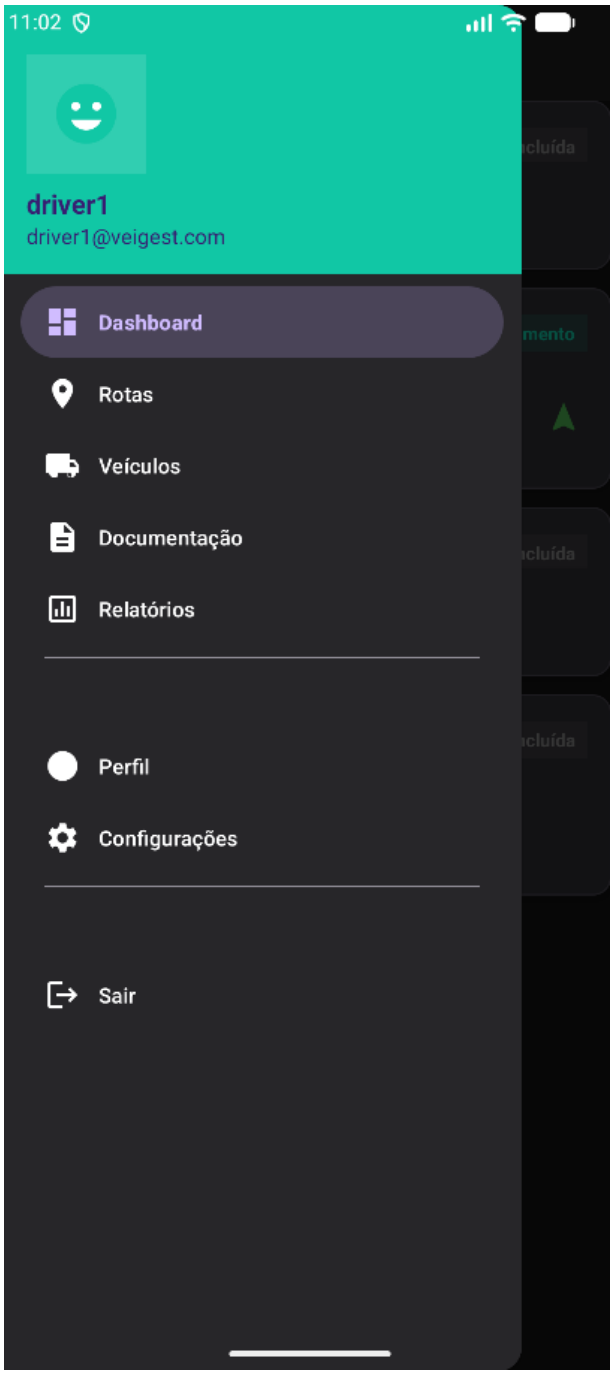

Figura 2: Menu lateral de navegação (Navigation Drawer).

O **Dashboard** atua como o centro de operações, agregando widgets com informação crítica: rota ativa, veículo em uso e estado da documentação.

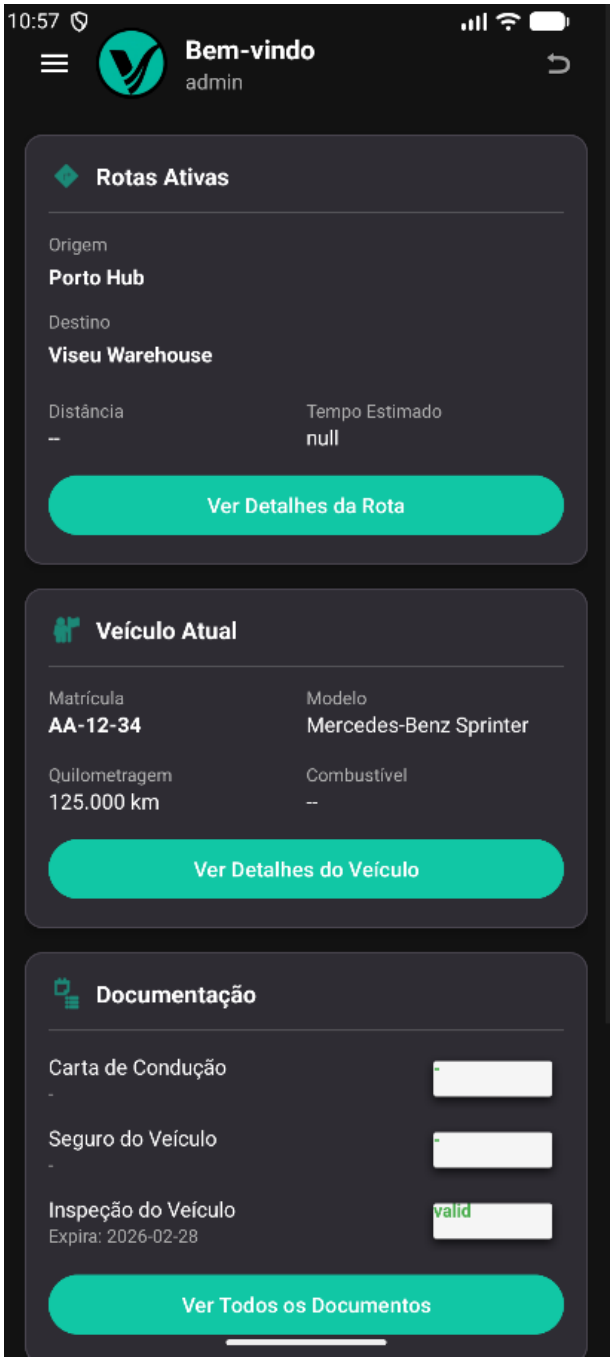

Figura 3: Dashboard principal com widgets informativos.

A gestão de **Veículos** (Figura 4) e **Rotas** (Figura 5) constitui o núcleo funcional da aplicação. Os formulários de criação e edição foram desenhados para serem preenchidos rapidamente, com validações de input em tempo real. O upload de imagens de veículos inclui um passo de compressão no cliente para otimizar o uso de dados móveis.

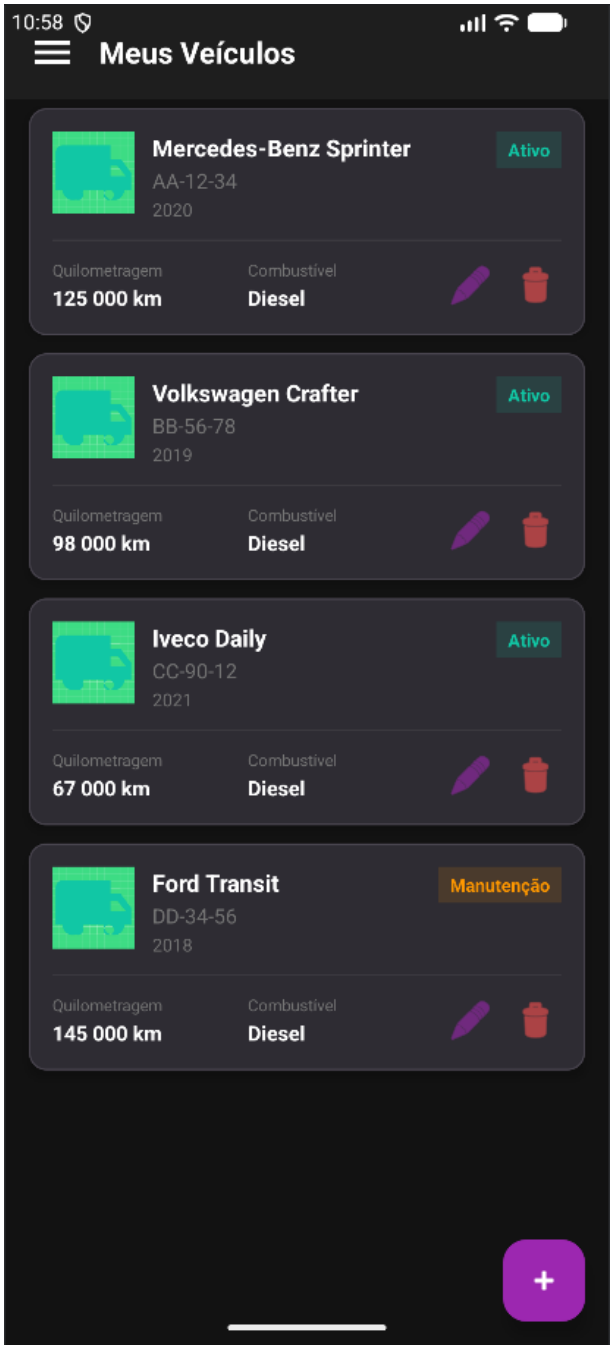

Figura 4: Módulo de gestão de veículos com suporte a imagens.

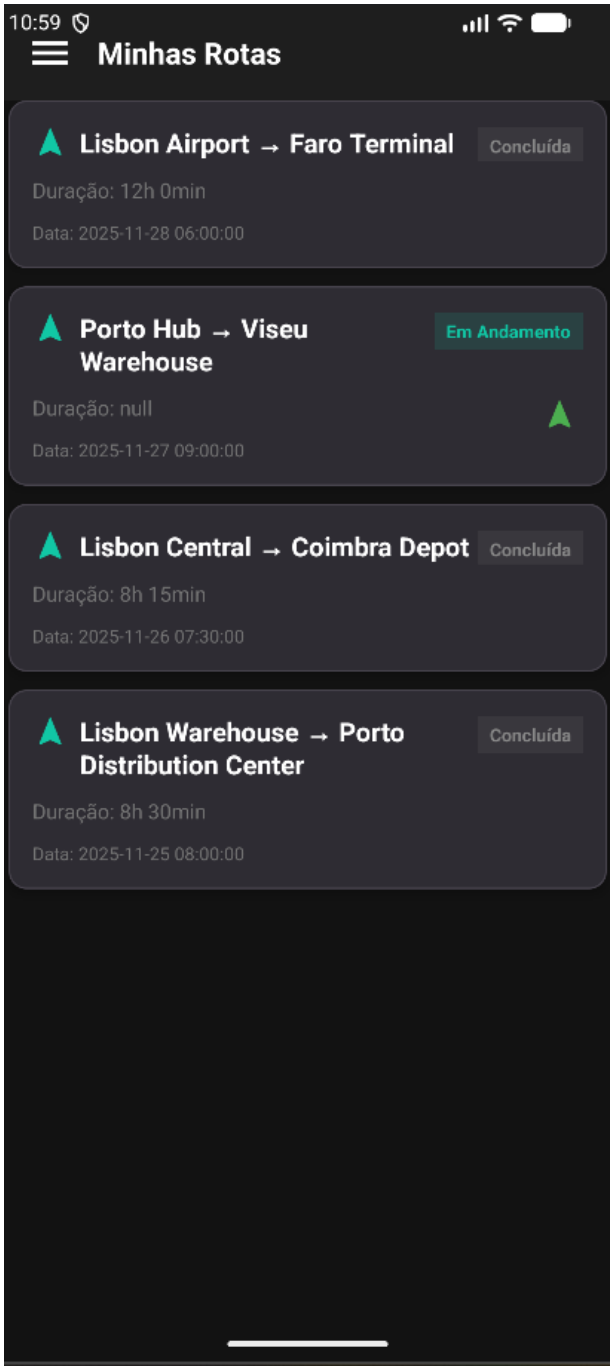

Figura 5: Listagem e gestão de estados de rotas.

A funcionalidade de **Relatórios** demonstra a capacidade da aplicação de gerar artefactos complexos (PDFs) utilizando as APIs de impressão nativas do Android, permitindo a desmaterialização de processos burocráticos.

Figura 6: Geração de relatórios PDF e envio por email.

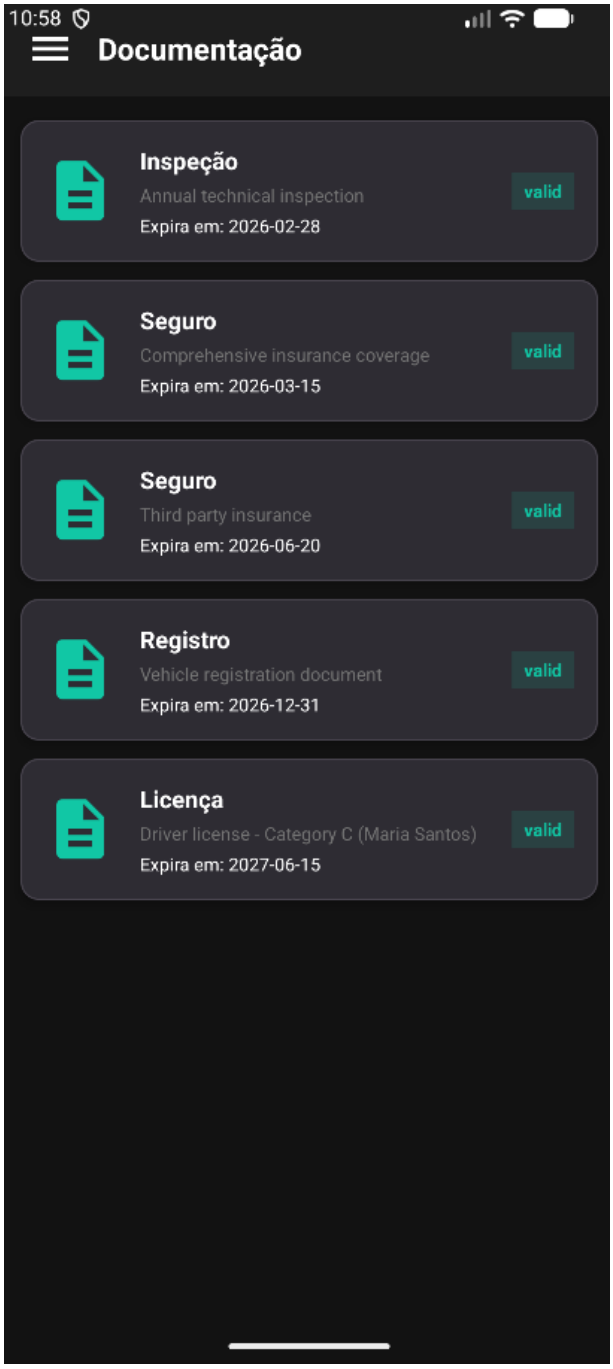

Figura 7: Monitorização de validade de documentos.

3 Conclusões

O desenvolvimento da VeiGest cumpriu os objetivos académicos e técnicos propostos. A aplicação resultante é uma ferramenta de gestão de frotas funcional, que tira partido das capacidades nativas do Android para resolver problemas reais de mobilidade empresarial.

A separação entre SDK e aplicação provou ser uma estratégia vencedora, simplificando os testes e permitindo uma evolução futura mais ágil. As principais dificuldades, relacionadas com a assincronia e gestão de memória (imagens), foram ultrapassadas através da aplicação correta de padrões de desenho de software.

Para evoluções futuras, prevê-se a implementação de sincronização de dados em segundo plano (via WorkManager) e um sistema de notificações push para aumentar a proatividade da aplicação.

Referências

[1] Google Developers, *Guide to App Architecture*, Android Developers Documentation, https://developer.android.com/jetpack/guide

[2] Google Developers, *Transmit network data using Volley*, Android Developers Documentation, https://developer.android.com/training/volley

[3] Google, *Material Design 3 Guidelines*, https://m3.material.io/

[4] Bumptech, *Glide: An image loading and caching library for Android*, https://github.com/bumptech/glide