UNIVERSIDADE
LUSÓFONA

Longe, Com Tudo

Trabalho Final de curso

Relatório Intercalar 2º Semestre

Bernardo Pinheiro, a22301338, LEI

Filipe Lau, a22301329, LEI

Orientador: Rui Ribeiro

Co-orientador: Fernando Angelino

Entidade Externa: Incubadora de ideias DEISI

Departamento de Engenharia Informática da Universidade Lusófona

Centro Universitário de Lisboa

10/04/2026

www.ulusofona.pt

Direitos de cópia

Agradecimentos

Resumo

O projeto “Longe, com Tudo” consiste no desenvolvimento de uma aplicação móvel destinada a pessoas que vivem em zonas rurais ou isoladas e que apresentam baixa literacia digital. A solução tem como objetivo facilitar o acesso a bens essenciais, permitindo que o utilizador realize compras, consulte produtos, faça pedidos de medicamentos e obtenha apoio de forma simples e intuitiva.

A aplicação estrutura-se em três módulos principais, correspondentes aos três perfis de utilizador: **Cliente**, **Entregador** e **Fornecedor**.

O módulo de cliente inclui navegação simplificada, exploração de lojas e farmácias, gestão de carrinho, checkout, rastreamento de encomenda, histórico de compras e centro de ajuda.

O módulo de entregador cobre a gestão de pedidos disponíveis, navegação GPS para recolha e entrega, confirmação de entregas e histórico de atividade.

O módulo de fornecedor abrange a gestão da loja, catálogo de produtos, stock, pedidos recebidos e análise de vendas.

Todo o desenho funcional foi orientado para reduzir a complexidade e tornar o uso acessível a públicos vulneráveis, focando-se em ações rápidas, visuais e de baixo esforço cognitivo.

Atualmente, o projeto encontra-se numa fase inicial de especificação, onde foram definidos requisitos funcionais e não funcionais, casos de uso, estrutura de dados e componentes principais. Embora ainda não exista implementação, o planeamento prevê o desenvolvimento futuro de um protótipo funcional (MVP) que permita validar a utilidade e a usabilidade da solução junto do público-alvo.

Palavras chave:

Dart
Flutter
Firebase
Aplicação Móvel
Dispositivos Móveis

Abstract

The "Longe, com Tudo" project consists of developing a mobile application aimed at people living in rural or isolated areas with low digital literacy. The solution seeks to facilitate access to essential goods, allowing users to make purchases, browse products, request medications, and obtain support in a simple and intuitive way.

The application is structured around three main modules, corresponding to the three user profiles: Client, Delivery Driver and Supplier.

The client module includes simplified navigation, exploration of shops and pharmacies, cart management, checkout, order tracking, purchase history and a help centre.

The delivery driver module covers the management of available orders, GPS navigation for collection and delivery, delivery confirmation and activity history.

The supplier module encompasses shop management, product catalogue, stock control, received orders and sales analysis.

The entire functional design was oriented towards reducing complexity and making the application accessible to vulnerable users, focusing on fast, visual and low cognitive effort interactions.

Currently, the project is in an early specification phase, where functional and non-functional requirements, use cases, data structure and key components have been defined. Although implementation has not yet begun, the planning foresees the future development of a functional prototype (MVP) that will allow validating the usefulness and usability of the solution with the target audience.

Key-words:

Índice

Lista de Figuras

Lista de Tabelas

Lista de Siglas

API	Interface de Programação de Aplicações
HTML	Linguagem de Marcação de Hipertexto
GCP	Google Cloud Platform
AVD	Android Virtual Device

1 Introdução

Pessoas que vivem em zonas rurais e remotas de Portugal, onde o acesso a bens essenciais continua a ser um desafio diário. Em muitas destas localidades, a combinação entre isolamento geográfico, envelhecimento populacional e reduzida literacia digital cria um conjunto de dificuldades que não é adequadamente enfrentado pelos serviços atuais. A distância aos centros urbanos, a escassez de transportes e a falta de ferramentas digitais adaptadas traduzem-se, para muitos residentes, numa dependência crescente de terceiros e numa perda de autonomia que afeta a qualidade de vida.

O projeto **"Longe, com tudo"** surge precisamente desta necessidade: criar uma aplicação móvel intuitiva, capaz de apoiar pessoas com pouca experiência digital no processo de encomenda de bens essenciais, unindo tecnologia a uma rede de apoio humano.

O objetivo deste TFC é explorar este problema de forma estruturada, fundamentar a relevância da solução e desenvolver um protótipo funcional.

1.1 Enquadramento

Portugal destaca-se pelo elevado grau de envelhecimento demográfico, situando-se em 2024 como o 11.º país com a idade mediana mais elevada a nível mundial, atingindo cerca de 47 anos. Este valor evidencia uma tendência crescente e posiciona o país significativamente acima da média global.

Esta tendência é especialmente notória nas regiões do interior, onde muitas aldeias enfrentam despovoamento, isolamento territorial e acesso reduzido a serviços essenciais, refletindo um cenário onde muitos cidadãos enfrentam desafios acrescidos de acesso, autonomia e inclusão digital.

1.2 Motivação e Identificação do Problema

A motivação deste projeto deriva das dificuldades vividas por pessoas que habitam aldeias isoladas, onde o acesso a certos bens essenciais representa um desafio significativo. Em muitos destes contextos, serviços como transportes públicos, supermercados e farmácias encontram-se a vários quilómetros de distância.

Sendo a população destas mesmas aldeias maioritariamente idosa, não possui meios digitais ou físicos para garantir autonomia no seu quotidiano.

Além da motivação social, existe também um enquadramento tecnológico relevante: embora existam inúmeras aplicações relacionadas com compras, saúde ou comunicação, estas são frequentemente complexas e/ou pouco adaptadas a utilizadores com reduzida literacia digital. Isto cria um hiato entre a oferta de soluções tecnológicas e a realidade das populações mais vulneráveis.

1.3 Objetivos

O objetivo geral deste trabalho é desenvolver um protótipo funcional de uma aplicação móvel que facilite o acesso a bens essenciais para pessoas que vivem em zonas rurais e remotas, garantindo uma utilização simples, intuitiva e adequada a utilizadores com baixa literacia digital.

Para suportar este objetivo geral, foram definidos os seguintes objetivos específicos:

- Analisar o contexto e necessidades reais dos utilizadores-alvo, identificando limitações de mobilidade, isolamento geográfico e dificuldades de utilização de tecnologia.
- Recolher e estruturar requisitos funcionais e não funcionais de forma coerente, assegurando que a solução final responde às necessidades identificadas.
- Modelar a solução proposta, incluindo a definição de casos de uso, fluxos de interação, entidades de dados e arquitetura conceptual.
- Desenhar protótipos de interface simples, acessíveis e adequados ao público-alvo, assegurando boa usabilidade mesmo em dispositivos de baixo desempenho.
- Planear a futura implementação do MVP, definindo tecnologias prováveis, ferramentas de desenvolvimento e etapas necessárias para validar o conceito.
- Avaliar riscos, limitações e dependências que possam influenciar o desenvolvimento nas fases seguintes.
- Preparar uma base sólida para desenvolvimento posterior, permitindo que, numa fase futura, o protótipo seja evoluído para um sistema funcional integrado com parceiros e serviços externos.

1.4 Estrutura do Documento

O presente relatório encontra-se organizado da seguinte forma:

• Na Secção 1 é apresentada a pertinência e viabilidade do projeto, incluindo a contextualização do problema e a justificação da sua relevância social e tecnológica.

• Na Secção 2 são definidos o enquadramento, os objetivos e a metodologia adotada para orientar o desenvolvimento do trabalho.

• Na Secção 3 é realizada a especificação e modelação da solução, apresentando os requisitos funcionais e não funcionais, bem como os modelos estruturais e comportamentais da aplicação.

• Na Secção 4 é descrita a solução proposta, incluindo arquitetura conceptual, tecnologias previstas e componentes principais do sistema.

• Na Secção 5 é apresentado DEFINIR/COMPLETAR

• Na Secção 6 é apresentado o planeamento inicial do projeto, incluindo o cronograma e análise crítica do progresso.

2 Pertinência e Viabilidade

2.1 Pertinência

A pertinência do projeto "Longe, com Tudo" assenta na resposta direta a um problema real e documentado: a dificuldade de acesso a bens essenciais por parte de populações idosas ou residentes em zonas rurais.

Segundo o INE, mais de 22% da população portuguesa vive em áreas de baixa densidade, onde o acesso a serviços essenciais é limitado e onde existe uma proporção significativamente maior de pessoas idosas.

Adicionalmente, de acordo com a OCDE, Portugal é um dos países europeus com maior índice de isolamento social entre pessoas idosas, verificando-se que dificuldades de mobilidade, distância a serviços e dependência de terceiros reforçam este isolamento.

A pertinência da solução aumenta quando se observam fatores como:

- **Baixa literacia digital**: Cerca de 40% dos adultos portugueses têm competências digitais reduzidas (Eurostat, 2024), o que justifica a necessidade de uma aplicação extremamente simples e acessível.
- **Desigualdade no acesso ao comércio e saúde**: Em muitas regiões, supermercados, farmácias e transportes situam-se a vários quilómetros da residência dos utilizadores.
- **Aumento de serviços de compras online**, mas com interfaces complexas e pouco adequadas a pessoas idosas, excluindo precisamente as populações que mais beneficiariam do serviço.

A aplicação proposta atua como uma ponte, oferecendo autonomia a utilizadores que, de outra forma, dependeriam de terceiros.

2.2 Viabilidade

Apesar de o presente trabalho se focar no desenvolvimento de um protótipo funcional, a solução demonstra forte potencial de viabilidade futura, quer técnica, quer social.

A viabilidade futura da solução assenta em três fatores principais:

- **Escalabilidade social:**
  O modelo pode ser facilmente expandido com o apoio de voluntários locais, instituições sociais, juntas de freguesia ou organizações comunitárias que já prestem apoio a populações envelhecidas. A integração com estas entidades permitiria operacionalizar entregas, apoio ao utilizador e acompanhamento contínuo.

- **Escalabilidade através de parcerias:**

A aplicação pode integrar-se com supermercados, farmácias e serviços de logística através de APIs simples, possibilitando automatizar pedidos e disponibilizar funcionalidades adicionais à medida que forem estabelecidas novas parcerias.

- **Sustentabilidade técnica:**
  A escolha de tecnologias leves e multiplataforma (como Flutter) facilita a manutenção, possibilita a execução em dispositivos baratos e recondicionados, e reduz custos operacionais, tornando o projeto viável para implementação em larga escala.

Assim, embora atualmente se encontre numa fase preliminar de conceptualização, a solução apresenta características que permitem a sua evolução para um serviço funcional, sustentável e com impacto social real.

2.3 Análise Comparativa com Soluções Existentes

2.3.1 Soluções existentes

1. Uber Eats / Glovo / Bolt Food

As principais plataformas de entrega de refeições e bens essenciais — como Uber Eats, Glovo ou Bolt Food — disponibilizam serviços de encomenda através de aplicações móveis com uma ampla rede de parceiros comerciais. Apesar de permitirem a aquisição de produtos alimentares e farmacêuticos, estas aplicações não estão adaptadas a populações com baixa literacia digital e requerem que o utilizador viva numa zona urbana abrangida pela rede de entregadores. Nas regiões rurais, a cobertura é muito reduzida ou inexistente, tornando estas soluções pouco adequadas ao contexto das aldeias isoladas.
Referência: Uber Eats (2024). Disponível em: https://www.ubereats.com

---

➤ 2. Continente Online / Pingo Doce Online / Auchan Online

Os grandes retalhistas portugueses oferecem serviços de compras online e entregas ao domicílio, permitindo que os utilizadores adquiram bens essenciais através de plataformas digitais. No entanto, estas lojas online exigem um nível de literacia digital moderado a elevado, incluindo navegação por categorias complexas, criação de contas e gestão de métodos de pagamento digitais. A cobertura geográfica também é limitada, com entregas sobretudo em áreas urbanas ou suburbanas.
Referência: Continente Online (2024). Disponível em: https://www.continente.pt

---

➤ 3. Farmácias Online (Farmácia.pt, Wells, outras)

Diversas plataformas permitem a compra online de medicamentos não sujeitos a receita médica e produtos de bem-estar. Embora constituam um recurso útil para algumas populações, a maioria destas soluções exige competências digitais e não inclui um sistema de apoio humano que auxilie utilizadores com dificuldades. Além disso, a entrega em zonas rurais

pode demorar vários dias, reduzindo a eficácia para populações isoladas.
Referência: Farmácia.pt (2024). Disponível em: https://www.farmacia.pt

---

➤ 4. SocialCare Apps (auxílio a idosos)

Existem aplicações internacionais focadas no apoio a idosos, como "Papa" (EUA), que liga séniores a assistentes humanos para pequenas tarefas, companhia e compras. Estas soluções introduzem o conceito de apoio humano combinado com tecnologia, mas não estão disponíveis em Portugal e requerem subscrições elevadas.
Referência: Papa — Family on Demand (2024). Disponível em: https://www.joinpapa.com

---

➤ 5. Projetos Locais de Apoio Municipal

Algumas autarquias portuguesas oferecem serviços de entrega de medicamentos ou compras para idosos através das juntas de freguesia. Contudo, estes serviços são geralmente informais, não digitalizados e com recursos muito limitados, dependendo fortemente da disponibilidade de voluntários.
Referência: Câmara Municipal de Viseu — Programas de Apoio Sénior (2023).

2.3.2 Análise de benchmarking

A tabela seguinte compara a solução "Longe, com tudo" com as alternativas de mercado. Os critérios foram definidos tendo em conta as necessidades das populações-alvo.

Tabela 1 Benchmarking

Característica / Solução	Longe, com Tudo	Apps de Entrega (UberEats/Glovo)	Supermercados Online	Farmácias Online	Apps de Apoio a Idosos	Serviços Municipais
Acesso em zonas rurais remotas	✓	–	–	–	–	–
Interface simples e adaptada a baixa literacia digital	✓	–	–	–	–	–
Apoio humano integrado (telefone/chat)	✓	–	–	–	✓	✓
Entrega de bens essenciais variados	✓	✓	✓	✓	–	✓
Totalmente orientado a população idosa	✓	–	–	–	✓	✓
Cobertura nacional escalável	✓	–	–	–	–	–
Baixo custo para utilizadores	✓	–	–	–	–	✓
Aplicação móvel focada em simplicidade	✓	–	–	–	–	–

Conclusão do benchmarking

A análise mostra que, embora existam soluções parcialmente semelhantes, **nenhuma integra simultaneamente acessibilidade digital, apoio humano, cobertura rural e foco no público idoso**. A solução proposta ocupa um nicho ainda não explorado por plataformas comerciais.

2.4 Proposta de inovação e mais-valias

A inovação do projeto "Longe, com tudo" assenta na combinação de três elementos fundamentais: **simplicidade digital, apoio humano e alcance territorial**. Embora existam aplicações de compras ou apoio sénior, nenhuma integra estes aspetos de forma orientada para populações envelhecidas em territórios isolados.

Elementos diferenciadores:

- **Interface ultra simples**, concebida para utilizadores com pouca experiência tecnológica:
    - ícones grandes (problema do dedo gordo)
    - navegação linear
    - ausência de menus complexos
- **Rede de apoio humano integrada**, permitindo que o utilizador solicite ajuda por telefone ou mensagem para completar a encomenda. Este elemento aproxima a solução de um modelo híbrido, mais inclusivo.
- **Foco explícito em zonas rurais e remotas**, tradicionalmente ignoradas pelos grandes serviços de entrega.
- **Acompanhamento do processo de encomenda em tempo real** com feedback simplificado.
- **Possibilidade de integração com parceiros locais**, como mercearias, farmácias e juntas de freguesia, promovendo economia local.

Mais-valias (benefícios):

- **Melhoria da autonomia dos utilizadores**, permitindo que cidadãos idosos mantenham independência.
- **Aumento da acessibilidade digital**, ao oferecer uma ferramenta adaptada a necessidades específicas.
- **Impacto social significativo**, reduzindo isolamento e dependência de terceiros.
- **Sustentabilidade territorial**, ao apoiar o comércio local e reduzir deslocações.
- **Eficiência operacional**, através de um sistema unificado que liga fornecedores, utilizadores e rede de apoio.

No caso de existir parceria empresarial, esta solução oferece **diferenciação de mercado**, expansão para regiões pouco exploradas e oportunidade de responsabilidade social corporativa.

2.5 Identificação de oportunidade de negócio

O projeto apresenta forte potencial de expansão comercial, especialmente no mercado português e posteriormente em países com desafios semelhantes (Espanha, Itália, Grécia). Alguns modelos de negócio possíveis incluem:

➤1. Parcerias com autarquias e juntas de freguesia

As autarquias podem financiar o uso da aplicação para apoiar populações idosas, contribuindo para políticas públicas de inclusão digital.

➤2. Comissão sobre vendas

A plataforma pode receber uma pequena percentagem por cada encomenda realizada através de parceiros locais.

➤3. Portal para fornecedores

Comércio local pode subscrever um plano de adesão, obtendo:

- maior visibilidade
- integração em rede logística rural
- acesso a dados relevantes sobre procura

➤4. Expansão para serviços complementares

A médio prazo, a aplicação pode integrar:

- agendamento de transportes
- consultas médicas por teleassistência
- pedidos de apoio social
- acompanhamento de medicação

3 Especificação e Modelação

3.1 Análise de Requisitos

A definição de requisitos constitui uma etapa fundamental do projeto, permitindo identificar de forma estruturada as funcionalidades necessárias que a solução deve garantir para responder às necessidades dos utilizadores-alvo. No contexto do projeto “Longe, com Tudo”, o levantamento de requisitos foi orientado para a criação de uma aplicação móvel simples, acessível e ajustada a utilizadores com baixa literacia digital, residentes em zonas geograficamente isoladas.

Na primeira entrega, os requisitos identificados focavam-se exclusivamente no perfil de cliente, não tendo sido ainda reconhecida a necessidade de contemplar outros tipos de utilizador. À medida que o projeto evoluiu, tornou-se evidente que a solução exigiria três perfis distintos — cliente, fornecedor e entregador — cada um com responsabilidades, fluxos e necessidades funcionais próprias. Esta identificação tardia implicou uma revisão e expansão significativa de toda a especificação de requisitos.

Nesta entrega, os requisitos encontram-se completamente redefinidos e estruturados para os três perfis de utilizador, cobrindo tanto requisitos funcionais como não funcionais, ao nível macro e micro.

3.1.1 Requisitos do Utilizador (Cliente)

Requisitos Funcionais

RF-001 - Botão para regressar ao ecrã anterior: O sistema deve disponibilizar, em todas as páginas, um botão para regressar ao ecrã anterior.

- RF-001.1 — O sistema deve apresentar um botão "Voltar" visível e consistente em todos os ecrãs navegáveis.
- RF-001.2 — Ao clicar em "Voltar", o sistema deve regressar ao ecrã anterior dentro do fluxo atual.
- RF-001.3 — Ao regressar, o sistema deve preservar o estado do ecrã anterior (ex.: pesquisa, filtros, scroll, seleção).
- RF-001.4 — Se não existir ecrã anterior no fluxo, o sistema deve manter o utilizador no mesmo ecrã sem erro.

RF-002 - Aviso ao sair de ecrã com dados não guardados: O sistema deve alertar o utilizador quando tentar sair de um ecrã com dados não guardados.

- RF-002.1 — O sistema deve detetar alterações não guardadas em formulários/campos.
- RF-002.2 — Ao tentar sair do ecrã com alterações não guardadas, o sistema deve mostrar um aviso claro.
- RF-002.3 — O aviso deve oferecer opções "Sair sem guardar" e "Continuar/Cancelar".

- RF-002.4 — Se o utilizador escolher continuar, o sistema deve manter o utilizador no ecrã e preservar os dados inseridos.

RF-003 - Indicação visual do ecrã atual: O sistema deve mostrar visualmente em que ecrã se encontra.

RF-003.1 — O sistema deve apresentar um título no topo que identifique o ecrã atual.

RF-003.2 — O título deve corresponder à funcionalidade atual (ex.: "Lojas", "Carrinho", "Farmácia").

RF-003.3 — O título deve manter-se visível durante o scroll do conteúdo ou reaparecer automaticamente.

RF-003.4 — O sistema não deve apresentar títulos ambíguos; nomes iguais devem ser diferenciados.

RF-004 - Padrão visual consistente: O sistema deve manter um padrão visual consistente (cores, tamanho e localização dos botões).

- RF-004.1 — O sistema deve usar o mesmo estilo visual de botões em todos os ecrãs.
- RF-004.2 — Os botões principais devem manter posição consistente.
- RF-004.3 — O sistema deve manter consistência nos ícones (mesmo significado = mesmo ícone).
- RF-004.4 — O sistema deve manter consistência de terminologia em toda a app.

RF-005 - Ecrã inicial com atalhos: O sistema deve ter um ecrã inicial com os principais atalhos.

- RF-005.1 — O sistema deve apresentar um ecrã inicial (Home) como ponto de entrada após login/arranque.
- RF-005.2 — O ecrã inicial deve conter atalhos para as funcionalidades principais (Compras, Farmácia, Histórico, Ajuda).
- RF-005.3 — Os atalhos devem ser acionáveis com um clique e levar diretamente ao módulo correspondente.
- RF-005.4 — O ecrã inicial deve permitir retornar rapidamente ao fluxo anterior quando aplicável.

RF-006 - Acesso ao Centro de Ajuda: O sistema deve permitir aceder ao Centro de Ajuda a partir de qualquer ecrã.

- RF-006.1 — O sistema deve permitir abrir o Centro de Ajuda a partir de qualquer ecrã.
- RF-006.2 — Ao abrir/fechar o Centro de Ajuda, o sistema deve preservar o estado do ecrã atual.
- RF-006.3 — O acesso ao Centro de Ajuda não deve bloquear ações críticas sem confirmação. RF-006.4 — O Centro de Ajuda deve fechar com um clique, sem perder dados do ecrã atual.

RF-007 - Botão para regressar ao ecrã inicial: O sistema deve disponibilizar, em todas as páginas, um botão para regressar ao ecrã inicial.

- RF-007.1 — O sistema deve disponibilizar um botão "Home/Início" acessível em todas as páginas.
- RF-007.2 — Ao clicar em "Home/Início", o sistema deve navegar para o ecrã inicial imediatamente.
- RF-007.3 — Ao regressar ao início, o carrinho deve manter o seu estado.
- RF-007.4 — O sistema deve evitar múltiplos caminhos inconsistentes para o "Home".

RF-008 - Listagem de estabelecimentos: O sistema deve mostrar todos os estabelecimentos disponíveis.

- RF-008.1 — O sistema deve listar estabelecimentos disponíveis para compra numa lista.
- RF-008.2 — Cada estabelecimento listado deve apresentar pelo menos nome e tipo.
- RF-008.3 — A lista deve permitir seleção do estabelecimento com um clique.
- RF-008.4 — Se não existirem estabelecimentos, o sistema deve apresentar mensagem clara e opção de voltar.

RF-009 - Estimativa de entrega: O sistema deve mostrar a estimativa de entrega por estabelecimento ou produto.

- RF-009.1 — O sistema deve apresentar a estimativa de entrega associada a cada estabelecimento.
- RF-009.2 — A estimativa deve ser apresentada em formato simples (ex.: "~45 min", "1–2 dias").
- RF-009.3 — Se a estimativa não estiver disponível, o sistema deve mostrar "Estimativa indisponível" sem bloquear a compra.
- RF-009.4 — A estimativa deve atualizar quando o utilizador muda de estabelecimento.

RF-010 - Pesquisa de estabelecimentos: O sistema deve permitir pesquisar estabelecimentos por nome.

- RF-010.1 — O sistema deve disponibilizar um campo de pesquisa de estabelecimentos.
- RF-010.2 — A pesquisa deve filtrar a lista à medida que o utilizador escreve.
- RF-010.3 — A pesquisa deve ignorar maiúsculas/minúsculas e acentos.
- RF-010.4 — Se não existirem resultados, o sistema deve apresentar "Sem resultados" e permitir limpar pesquisa.

RF-011 - Filtros de produtos: O sistema deve permitir aplicar filtros (preço, categoria, etc.).

- RF-011.1 — O sistema deve permitir aplicar filtros de produtos dentro de um estabelecimento. RF-011.2 — O sistema deve permitir limpar todos os filtros com uma ação.

- RF-011.3 — O sistema deve manter os filtros aplicados ao voltar do detalhe do produto para a listagem.
- RF-011.4 — O sistema deve indicar visualmente que existem filtros ativos.

RF-012 - Apenas produtos em stock: Cada estabelecimento deve listar apenas produtos em stock.

- RF-012.1 — O sistema deve esconder produtos sem stock da listagem principal por defeito.
- RF-012.2 — Se apresentado algum produto sem stock, deve estar claramente marcado como "Indisponível".
- RF-012.3 — O sistema deve impedir a ação de adicionar ao carrinho produtos sem stock.
- RF-012.4 — O sistema deve revalidar stock antes do checkout e informar alterações ao utilizador.

RF-013 - Informação completa por produto: Cada produto deve ter imagem, preço e descrição.

- RF-013.1 — Cada produto deve apresentar imagem, preço e nome na listagem.
- RF-013.2 — O detalhe do produto deve apresentar descrição do produto.
- RF-013.3 — O sistema deve apresentar descrição alternativa da imagem para acessibilidade.
- RF-013.4 — Se não existir imagem, o sistema deve mostrar placeholder e manter preço/descrição visíveis.

RF-014 - Múltiplos métodos de pagamento: O sistema deve permitir múltiplos métodos de pagamento.

RF-014.1 — O sistema deve apresentar uma lista de métodos de pagamento no checkout.

RF-014.2 — O utilizador deve conseguir selecionar exatamente um método de pagamento por pedido.

RF-014.3 — O sistema deve guardar o método selecionado no pedido.

RF-014.4 — Se um método estiver indisponível, o sistema deve desativá-lo e indicar o motivo.

RF-015 - Pagamento em dinheiro: O sistema deve permitir pagamento em dinheiro na entrega.

- RF-015.1 — O sistema deve permitir escolher "Dinheiro na entrega" como método de pagamento.
- RF-015.2 — Ao confirmar o pedido com "Dinheiro", o sistema deve marcar o pedido com esse método.
- RF-015.3 — O sistema deve apresentar confirmação explícita de que o pagamento será no ato da entrega.
- RF-015.4 — O sistema não deve exigir dados de pagamento digital quando "Dinheiro na entrega" está selecionado.

RF-016 - Notas adicionais ao pedido: O sistema deve permitir adicionar notas adicionais ao pedido.

- RF-016.1 — O sistema deve disponibilizar um campo "Notas para a entrega" no checkout.
- RF-016.2 — O utilizador deve conseguir escrever e editar a nota antes de confirmar o pedido. RF-016.3 — O sistema deve associar a nota ao pedido guardado.
- RF-016.4 — O campo deve ser opcional e não bloquear o checkout quando vazio.

RF-017 - Histórico de compras: O sistema deve registar todas as compras e permitir consulta a qualquer momento.

- RF-017.1 — O sistema deve guardar todos os pedidos concluídos.
- RF-017.2 — O sistema deve disponibilizar um ecrã de histórico acessível a qualquer momento. RF-017.3 — Cada item do histórico deve mostrar data e total (mínimo).
- RF-017.4 — O histórico deve persistir entre sessões.

RF-018 - Repetir pedidos: O sistema deve permitir repetir pedidos do histórico.

- RF-018.1 — O sistema deve permitir selecionar um pedido do histórico.
- RF-018.2 — O sistema deve permitir "Repetir pedido", criando um carrinho baseado no pedido selecionado.
- RF-018.3 — O sistema deve pedir confirmação antes de substituir o carrinho atual.
- RF-018.4 — O sistema deve recalcular total do novo pedido com preços atuais, informando diferenças relevantes.

RF-019 - Validação de stock ao repetir: O sistema deve impedir a seleção de quantidades superiores ao stock ao repetir pedidos.

- RF-019.1 — Ao repetir um pedido, o sistema deve validar stock atual para cada produto.
- RF-019.2 — Se a quantidade anterior exceder stock atual, o sistema deve ajustar para o máximo disponível e avisar.
- RF-019.3 — Se um produto não estiver disponível, o sistema deve removê-lo do pedido repetido e avisar.
- RF-019.4 — O sistema deve apresentar um resumo das alterações antes da confirmação do pedido repetido.

RF-020 - Detalhes completos do pedido: O sistema deve mostrar os detalhes completos de cada pedido.

- RF-020.1 — O sistema deve mostrar detalhe do pedido com lista de produtos e quantidades. RF-020.2 — O sistema deve mostrar preço unitário e total por item.
- RF-020.3 — O sistema deve mostrar total final e método de pagamento do pedido.
- RF-020.4 — O sistema deve mostrar data/hora do pedido e identificador do pedido.

RF-021 - Ordenação do histórico: Os pedidos devem estar ordenados por data (mais recente → mais antigo).

- RF-021.1 — O sistema deve ordenar pedidos por defeito do mais recente para o mais antigo. RF-021.2 — O sistema deve manter a ordenação consistente após abrir e fechar o ecrã.
- RF-021.3 — A ordenação deve considerar data/hora (não apenas data).
- RF-021.4 — O sistema deve tratar datas inválidas colocando o pedido no fim e sinalizando.

RF-022 - Filtros e ordenação do histórico: O sistema deve permitir aplicar filtros e alterar a ordem de ordenação.

- RF-022.1 — O sistema deve permitir filtrar histórico por estabelecimento.
- RF-022.2 — O sistema deve permitir alternar ordenação (mais antigo → mais recente).
- RF-022.3 — O sistema deve permitir limpar filtros com uma ação.
- RF-022.4 — O sistema deve indicar visualmente quando há filtros/ordenação alternativa ativa.

RF-023 - Um único carrinho ativo: Não deve ser permitido ter mais do que um carrinho ativo.

- RF-023.1 — O sistema deve garantir que existe apenas um carrinho ativo por utilizador.
- RF-023.2 — Ao iniciar compras noutro estabelecimento, o sistema deve pedir confirmação para substituir o carrinho.
- RF-023.3 — O sistema deve impedir a mistura de produtos de estabelecimentos diferentes no mesmo carrinho.
- RF-023.4 — O sistema deve manter o carrinho ativo entre ecrãs e sessões.

RF-024 - Visualização do carrinho: O sistema deve apresentar um carrinho com os produtos selecionados.

- RF-024.1 — O sistema deve apresentar uma lista dos itens adicionados ao carrinho.
- RF-024.2 — Cada item no carrinho deve mostrar nome, quantidade e preço.
- RF-024.3 — O sistema deve permitir remover um item do carrinho.
- RF-024.4 — O sistema deve atualizar o total automaticamente após qualquer alteração no carrinho.

RF-025 - Carrinho em modo preview: O carrinho deve estar sempre visível em modo preview.

- RF-025.1 — O sistema deve apresentar um ícone/preview do carrinho sempre visível durante o fluxo de compras.
- RF-025.2 — O preview deve permanecer visível ao navegar entre listagens, detalhes e filtros. RF-025.3 — O preview não deve ocultar ações críticas.
- RF-025.4 — O preview deve ser clicável e abrir o carrinho expandido.

RF-026 - Contagem e total no preview: O carrinho em preview deve exibir a contagem de produtos e o valor total.

- RF-026.1 — O preview do carrinho deve mostrar a contagem total de itens.

- RF-026.2 — O preview deve mostrar o total acumulado quando aplicável.
- RF-026.3 — A contagem e total devem atualizar em tempo real ao adicionar/remover itens.
- RF-026.4 — Se o carrinho estiver vazio, a contagem deve ser 0 e o total 0,00.

RF-027 - Expansão do carrinho: Ao clicar no carrinho, deve expandir para mostrar todos os produtos.

- RF-027.1 — Ao clicar no preview, o sistema deve abrir o carrinho em modo expandido.
- RF-027.2 — O carrinho expandido deve mostrar todos os itens e permitir scroll.
- RF-027.3 — O carrinho expandido deve ter botão para fechar e voltar ao ecrã anterior.
- RF-027.4 — Ao fechar, o sistema deve regressar exatamente ao ponto onde o utilizador estava.

RF-028 - Acesso ao carrinho em qualquer momento: O sistema deve permitir aceder ao carrinho a qualquer momento.

- RF-028.1 — O sistema deve permitir abrir o carrinho a qualquer momento durante a compra.
- RF-028.2 — O acesso ao carrinho deve estar disponível em todas as páginas do fluxo de compras.
- RF-028.3 — A abertura do carrinho não deve apagar filtros/pesquisa do ecrã subjacente.
- RF-028.4 — O sistema deve permitir retomar o fluxo após fechar o carrinho sem reiniciar o processo.

RF-029 - Esvaziar carrinho: O sistema deve disponibilizar um botão para esvaziar o carrinho.

- RF-029.1 — O sistema deve disponibilizar um botão "Esvaziar carrinho".
- RF-029.2 — A ação deve remover todos os itens do carrinho e repor contagem/total.
- RF-029.3 — Após esvaziar, o sistema deve manter o utilizador no ecrã do carrinho.
- RF-029.4 — O sistema deve desativar "Finalizar compra" quando o carrinho estiver vazio.

RF-030 - Confirmação ao esvaziar: O sistema deve pedir confirmação ao esvaziar o carrinho.

- RF-030.1 — Ao clicar em "Esvaziar carrinho", o sistema deve apresentar um diálogo de confirmação.
- RF-030.2 — O diálogo deve ter "Confirmar" e "Cancelar".
- RF-030.3 — Se o utilizador cancelar, o carrinho deve permanecer inalterado.
- RF-030.4 — Se o utilizador confirmar, o carrinho deve ser esvaziado e o sistema deve mostrar feedback.

RF-031 - Aviso de carrinho vazio: O sistema deve mostrar um aviso caso tente finalizar compra com o carrinho vazio.

- RF-031.1 — Se o utilizador clicar em "Finalizar compra" com carrinho vazio, o sistema deve bloquear a ação.
- RF-031.2 — O sistema deve mostrar um pop-up com instrução clara.
- RF-031.3 — O pop-up deve ter um botão para fechar e regressar ao ecrã atual.
- RF-031.4 — O sistema não deve criar pedido nem avançar no checkout com carrinho vazio.

RF-032 - Listagem de medicamentos sem receita: O sistema deve listar todos os medicamentos que não requeiram receita.

- RF-032.1 — O sistema deve listar medicamentos/produtos de farmácia que não exigem receita.
- RF-032.2 — A listagem deve permitir adicionar ao carrinho produtos sem receita.
- RF-032.3 — O sistema deve permitir pesquisar e/ou filtrar produtos de farmácia.
- RF-032.4 — O sistema deve identificar claramente produtos "Sem Receita".

RF-033 - Informação dos medicamentos: Cada medicamento deve ter imagem, preço e descrição.

- RF-033.1 — Cada produto de farmácia deve apresentar imagem, nome e preço na listagem.
- RF-033.2 — O detalhe deve apresentar descrição do produto/uso.
- RF-033.3 — O sistema deve apresentar descrição alternativa da imagem para acessibilidade.
- RF-033.4 — Se não existir imagem, o sistema deve manter o produto comprável com placeholder.

RF-034 - Requisitar produtos com receita: O sistema deve disponibilizar um botão 'Requisitar produtos'.

- RF-034.1 — O sistema deve disponibilizar um botão "Requisitar produtos" para fluxo com receita.
- RF-034.2 — Ao clicar, o sistema deve abrir o ecrã de submissão de documentos (CC e receita).
- RF-034.3 — O sistema deve explicar de forma simples o que é necessário submeter.
- RF-034.4 — O sistema deve permitir cancelar o fluxo e regressar à farmácia sem perder o carrinho.

RF-035 - Fotografia de CC e receita: O sistema deve pedir fotografia do CC e receita médica ao requisitar.

- RF-035.1 — O sistema deve solicitar captura/seleção de fotografia do CC.
- RF-035.2 — O sistema deve solicitar captura/seleção de fotografia da receita médica.
- RF-035.3 — O sistema deve impedir submissão se faltar algum documento obrigatório.
- RF-035.4 — O sistema deve permitir refazer fotografia antes de submeter.

RF-036 - Confirmação de submissão: O sistema deve confirmar submissão dos documentos.

- RF-036.1 — Após submissão, o sistema deve apresentar confirmação de envio com mensagem clara.
- RF-036.2 — A confirmação deve indicar que a receita será analisada (estado "Em análise").
- RF-036.3 — O sistema deve registar o estado da submissão associado ao utilizador.
- RF-036.4 — Se ocorrer erro no envio, o sistema deve mostrar mensagem e opção "Tentar novamente".

RF-037 - Adição automática de medicamentos prescritos: Os produtos prescritos devem ser automaticamente adicionados ao carrinho após validação.

- RF-037.1 — Após validação da receita, o sistema deve adicionar automaticamente os medicamentos prescritos ao carrinho.
- RF-037.2 — O sistema deve notificar o utilizador que os itens foram adicionados.
- RF-037.3 — Se algum medicamento não estiver disponível, o sistema deve informar e não adicionar esse item.
- RF-037.4 — O sistema deve permitir ao utilizador rever os itens adicionados antes de finalizar a compra.

RF-038 - Botão de ajuda em todos os ecrãs: O sistema deve apresentar o botão de ajuda no canto inferior direito em todos os ecrãs.

- RF-038.1 — O sistema deve apresentar um botão flutuante de Ajuda no canto inferior direito.
- RF-038.2 — O botão deve estar presente em todos os ecrãs principais e fluxos críticos.
- RF-038.3 — O botão não deve ocultar elementos essenciais.
- RF-038.4 — O botão deve ser acessível com um clique.

RF-039 - Expansão do menu de ajuda: O botão deve expandir e mostrar as opções do centro de ajuda.

- RF-039.1 — Ao clicar no botão de Ajuda, o sistema deve expandir um menu com opções.
- RF-039.2 — O menu deve apresentar opções de forma simples (ex.: "Mensagem", "Chamada").
- RF-039.3 — O menu deve poder ser fechado com um clique sem alterar o estado do ecrã atual.
- RF-039.4 — O menu deve ser consistente em todos os ecrãs.

RF-040 - Contacto com suporte: O sistema deve permitir contactar suporte por mensagem e chamada.

- RF-040.1 — O sistema deve permitir iniciar contacto por mensagem a partir do Centro de Ajuda.

- RF-040.2 — O sistema deve permitir iniciar contacto por chamada a partir do Centro de Ajuda.
- RF-040.3 — Após iniciar contacto, o sistema deve manter o contexto do utilizador para retomar depois.
- RF-040.4 — Se o contacto falhar (sem rede), o sistema deve informar e sugerir alternativa.

Requisitos Não Funcionais

RNF-001 - Simplicidade de utilização: O sistema deve ser simples de utilizar por pessoas com baixa literacia digital.

- RNF-001.1 — A aplicação deve permitir executar tarefas principais sem exigir conhecimentos técnicos.
- RNF-001.2 — A aplicação deve usar linguagem simples e evitar termos técnicos nas ações principais.
- RNF-001.3 — A aplicação deve minimizar decisões por ecrã, evitando múltiplas escolhas complexas em simultâneo.

RNF-002 - Máximo de passos por fluxo: O sistema deve garantir que ações principais são concluídas com no máximo 3 a 4 passos.

- RNF-002.1 — O fluxo "ver loja → adicionar item → abrir carrinho → finalizar" não deve exceder 4 passos principais.
- RNF-002.2 — O fluxo "abrir histórico → selecionar pedido → repetir" não deve exceder 3 passos principais.
- RNF-002.3 — O sistema deve evitar ecrãs intermédios desnecessários.

RNF-003 - Feedback visual: O sistema deve fornecer feedback visual claro após cada ação.

- RNF-003.1 — Após ações como "Adicionar ao carrinho", deve existir confirmação visual imediata.
- RNF-003.2 — Em carregamentos, o sistema deve apresentar indicador de progresso/estado.
- RNF-003.3 — Em erro, o sistema deve mostrar mensagem clara com instrução de correção.

RNF-004 - Contraste e legibilidade: O sistema deve permitir leitura clara (alto contraste e legibilidade).

- RNF-004.1 — O texto e elementos principais devem apresentar contraste suficiente para leitura em condições normais.
- RNF-004.2 — A aplicação deve evitar texto sobre imagens sem fundo/contraste adequado.

- RNF-004.3 — Ícones críticos devem ter rótulo textual.

RNF-005 - Tamanhos adequados a utilizadores idosos: O sistema deve usar tamanhos de letra e botões adequados a utilizadores idosos.

- RNF-005.1 — Elementos clicáveis devem ter área de toque confortável.
- RNF-005.2 — Texto essencial deve ter tamanho suficiente para leitura confortável.
- RNF-005.3 — O sistema deve evitar ações que exijam precisão motora fina.

RNF-006 - Sem gestos complexos: O sistema deve evitar dependência de gestos complexos.

- RNF-006.1 — Todas as ações essenciais devem ser executáveis com clique simples.
- RNF-006.2 — A aplicação deve evitar gestos obrigatórios sem alternativa por botão.
- RNF-006.3 — A aplicação deve suportar navegação completa sem gestos avançados.

RNF-007 - Tempos de resposta rápidos: O sistema deve apresentar tempos de resposta rápidos em uso normal.

- RNF-007.1 — Os ecrãs principais devem abrir rapidamente em condições normais de rede.
- RNF-007.2 — A pesquisa e filtros devem responder sem bloqueios visíveis.
- RNF-007.3 — A aplicação deve evitar congelamentos durante scroll e navegação.

RNF-008 - Otimização de dados e recursos: O sistema deve otimizar o consumo de dados e recursos.

- RNF-008.1 — Imagens devem ser carregadas de forma otimizada.
- RNF-008.2 — O sistema deve usar cache local para reduzir downloads repetidos.
- RNF-008.3 — A aplicação deve minimizar chamadas redundantes durante a navegação.

RNF-009 - Preservação de dados entre sessões: O sistema deve preservar dados do utilizador entre sessões.

- RNF-009.1 — O carrinho deve ser restaurado após fechar e reabrir a app.
- RNF-009.2 — O histórico deve manter-se disponível entre sessões.
- RNF-009.3 — O estado do fluxo deve ser recuperável após interrupção inesperada.

RNF-010 - Fiabilidade em pedidos e pagamentos: O sistema deve evitar falhas críticas durante pedidos e pagamentos.

- RNF-010.1 — Em erro durante checkout, o sistema deve permitir retomar sem perder o carrinho.
- RNF-010.2 — O sistema deve validar dados essenciais antes de confirmar um pedido.
- RNF-010.3 — Em caso de falha, o sistema deve apresentar mensagem e caminho de recuperação.

RNF-011 - Tratamento de erros: O sistema deve tratar erros de forma clara e segura.

- RNF-011.1 — Erros devem ser apresentados em linguagem simples (sem códigos técnicos).
- RNF-011.2 — O sistema deve prevenir operações inválidas.
- RNF-011.3 — O sistema deve garantir que uma falha num módulo não bloqueia toda a aplicação.

RNF-012 - Proteção de dados pessoais: O sistema deve proteger dados pessoais e sensíveis.

- RNF-012.1 — Dados sensíveis devem ser armazenados de forma segura no dispositivo.
- RNF-012.2 — A aplicação deve impedir acesso a histórico/dados sem autenticação.
- RNF-012.3 — A aplicação deve evitar expor dados sensíveis em notificações ou ecrãs de bloqueio.

RNF-013 - Privacidade e RGPD: O sistema deve respeitar princípios de privacidade e RGPD.

- RNF-013.1 — O utilizador deve poder consultar informação de privacidade e finalidades de dados.
- RNF-013.2 — O sistema deve recolher apenas dados necessários para as funcionalidades previstas.
- RNF-013.3 — Deve existir mecanismo para apagar dados locais do utilizador.

RNF-014 - Proteção de documentos: Documentos (CC/receitas) devem ser tratados com proteção reforçada.

- RNF-014.1 — Documentos devem ser guardados apenas pelo tempo estritamente necessário ao processo.
- RNF-014.2 — Documentos devem estar acessíveis apenas a fluxos autorizados.
- RNF-014.3 — Após submissão/validação, o sistema deve permitir remover/limpar documentos armazenados localmente.

RNF-015 - Compatibilidade com dispositivos de baixo custo: O sistema deve funcionar em dispositivos Android de baixo custo/recondicionados.

- RNF-015.1 — A aplicação deve manter-se funcional em dispositivos com desempenho reduzido.
- RNF-015.2 — A aplicação deve ser utilizável em diferentes tamanhos e resoluções de ecrã.
- RNF-015.3 — A aplicação deve evitar dependências pesadas que prejudiquem dispositivos com pouca memória.

RNF-016 - Disponibilidade: O sistema deve estar disponível para uso regular (exceto manutenção).

- RNF-016.1 — Funcionalidades locais devem manter-se acessíveis mesmo sem rede.
- RNF-016.2 — O sistema deve informar quando um serviço externo estiver indisponível.
- RNF-016.3 — O sistema deve permitir retentar operações após falha de rede.

RNF-017 - Funcionamento com internet fraca: O sistema deve lidar com internet fraca/instável.

- RNF-017.1 — Em falha de rede, o sistema deve mostrar estado "Sem ligação" e instruções.
- RNF-017.2 — Operações que dependem de rede devem ser reexecutáveis sem duplicar ações do utilizador.
- RNF-017.3 — O sistema deve evitar perder dados introduzidos durante oscilações de rede.

RNF-018 - Integridade de dados: O sistema deve manter consistência dos dados do carrinho e pedidos.

- RNF-018.1 — O total do carrinho deve refletir corretamente quantidades e preços atuais.
- RNF-018.2 — A validação de stock deve ocorrer antes de confirmar pedido.
- RNF-018.3 — O sistema deve impedir estados inválidos.

RNF-019 - Manutenibilidade modular: O sistema deve ser modular para facilitar evolução do projeto.

- RNF-019.1 — Módulos devem ser isolados.
- RNF-019.2 — Alterações num módulo não devem obrigar a reescrever os restantes.
- RNF-019.3 — Regras comuns devem ser reutilizáveis e centralizadas.

RNF-020 - Conteúdos atualizáveis: Conteúdos e configurações devem ser atualizáveis com impacto mínimo.

- RNF-020.1 — Textos e rótulos devem ser centralizados.
- RNF-020.2 — Imagens e recursos devem ser organizados e substituíveis sem alterar fluxos.
- RNF-020.3 — Configurações devem ser geríveis sem reestruturação do código.

RNF-021 - Testabilidade: O sistema deve permitir validação por testes funcionais e de aceitação.

- RNF-021.1 — Cada fluxo principal deve ser testável do início ao fim com passos reproduzíveis.
- RNF-021.2 — Estados e mensagens devem ser determinísticos.
- RNF-021.3 — O sistema deve permitir simular falhas de rede e validar comportamentos de recuperação.

RNF-022 - Observabilidade: O sistema deve permitir diagnóstico básico de problemas.

- RNF-022.1 — O sistema deve registar eventos essenciais.
- RNF-022.2 — O registo deve evitar armazenar dados sensíveis em texto simples.
- RNF-022.3 — O sistema deve permitir recolha de informação mínima para suporte.

RNF-023 - Portabilidade: O sistema deve ser preparado para evolução futura (ex.: integração com API).

- RNF-023.1 — As fontes de dados devem ser abstraídas para permitir trocar local/API sem reescrever UI.
- RNF-023.2 — O modelo de dados deve suportar sincronização futura sem perder histórico/carrinho.
- RNF-023.3 — A aplicação deve manter comportamento consistente independentemente da fonte de dados.

RNF-024 - Português europeu consistente: O sistema deve usar português consistente e adequado ao público-alvo.

- RNF-024.1 — O sistema deve manter terminologia consistente em toda a app.
- RNF-024.2 — O sistema deve evitar português do Brasil e manter português europeu.
- RNF-024.3 — Mensagens devem ser curtas, diretas e sem jargão técnico.

3.1.2 Requisitos do Entregador

Requisitos Funcionais

RFE-001 - Listagem de entregas: O sistema deve listar as entregas atribuídas ao entregador.

- RFE-001.1 — A lista deve estar ordenada por prioridade temporal (mais urgente primeiro).
- RFE-001.2 — Deve ser possível filtrar por estado (por aceitar, em percurso, concluída, falhada).

RFE-002 - Detalhes da entrega: O sistema deve permitir ver detalhes completos de uma entrega.

- RFE-002.1 — O detalhe deve mostrar morada completa e instruções/notas do utilizador.
- RFE-002.2 — O detalhe deve mostrar itens/quantidades e método de pagamento.

RFE-003 - Aceitar entrega: O sistema deve permitir aceitar uma entrega atribuída.

- RFE-003.1 — O entregador deve conseguir aceitar com um clique e confirmar a atribuição.
- RFE-003.2 — A entrega aceite deve passar para estado "Aceite" e ficar registada.

RFE-004 - Estado Em percurso: O sistema deve permitir atualizar o estado para "Em percurso".

- RFE-004.1 — O entregador deve conseguir marcar "Em percurso" com um clique.

- RFE-004.2 — A hora de início de percurso deve ficar registada no histórico da entrega.

RFE-005 - Estado Entregue: O sistema deve permitir concluir a entrega com estado "Entregue".

- RFE-005.1 — O entregador deve conseguir marcar "Entregue" com um clique.
- RFE-005.2 — A hora de conclusão deve ser guardada automaticamente.

RFE-006 - Estado Falhada: O sistema deve permitir marcar uma entrega como "Falhada" com motivo.

- RFE-006.1 — Ao marcar "Falhada", o sistema deve exigir seleção de um motivo.
- RFE-006.2 — O sistema deve guardar o motivo e a hora do evento.

RFE-007 - Navegação para a entrega: O sistema deve permitir abrir navegação para a morada de entrega.

- RFE-007.1 — Deve existir botão "Abrir navegação" no detalhe da entrega.
- RFE-007.2 — A aplicação deve abrir o mapa com a morada preenchida.

RFE-008 - Confirmação de pagamento: O sistema deve suportar confirmação de pagamento em dinheiro na entrega.

- RFE-008.1 — Se o pagamento for "dinheiro", o sistema deve mostrar valor a cobrar.
- RFE-008.2 — O entregador deve conseguir confirmar "Pagamento recebido" quando aplicável.

RFE-009 - Prova de entrega: O sistema deve permitir registar prova de entrega.

- RFE-009.1 — O sistema deve permitir registar assinatura ou confirmação equivalente.
- RFE-009.2 — O sistema deve associar a prova ao registo da entrega.

RFE-010 - Contactar utilizador: O sistema deve permitir contactar o utilizador relativamente a uma entrega.

- RFE-010.1 — Deve existir ação "Contactar utilizador" a partir do detalhe da entrega.
- RFE-010.2 — O sistema deve registar que foi efetuada tentativa de contacto.

RFE-011 - Reportar incidentes: O sistema deve permitir reportar incidentes/problemas na entrega.

- RFE-011.1 — Deve existir opção "Reportar incidente" com categorias predefinidas.
- RFE-011.2 — O incidente deve aceitar descrição curta e ficar associado à entrega.

RFE-012 - Histórico de entregas: O sistema deve permitir consultar histórico de entregas realizadas.

- RFE-012.1 — O histórico deve listar entregas concluídas/falhadas por data.
- RFE-012.2 — Deve ser possível abrir detalhe de entregas passadas.

Requisitos Não Funcionais

RNFE-001 - Simplicidade da interface: A interface do entregador deve ser simples e operável com poucos cliques.

- RNFE-001.1 — Atualizar um estado deve exigir no máximo 2 cliques.
- RNFE-001.2 — Botões principais devem ser grandes e fáceis de clicar.

RNFE-002 - Priorização de informação crítica: A aplicação deve priorizar informação crítica (morada, notas, pagamento) sem ruído.

- RNFE-002.1 — O detalhe deve mostrar morada, notas e pagamento acima de informação secundária.
- RNFE-002.2 — O sistema deve evitar ecrãs intermédios desnecessários.

RNFE-003 - Performance: A aplicação deve responder rapidamente ao abrir listas e detalhes.

- RNFE-003.1 — A lista de entregas deve abrir rapidamente em condições normais.
- RNFE-003.2 — O detalhe da entrega deve abrir sem bloqueios visíveis.

RNFE-004 - Consumo de dados: A aplicação deve minimizar consumo de dados ao sincronizar estados.

- RNFE-004.1 — Atualizações de estado devem enviar apenas dados essenciais.
- RNFE-004.2 — O sistema deve evitar pedidos repetidos ao servidor sem necessidade.

RNFE-005 - Funcionamento com rede fraca: O sistema deve manter o funcionamento com internet fraca/instável.

- RNFE-005.1 — Sem rede, o sistema deve permitir registar ações como "pendentes".
- RNFE-005.2 — Ao recuperar rede, o sistema deve sincronizar automaticamente as ações pendentes.

RNFE-006 - Consistência de estados: O sistema deve evitar duplicações/estados inconsistentes na entrega.

- RNFE-006.1 — O sistema deve impedir marcar a mesma entrega como "Entregue" duas vezes.
- RNFE-006.2 — O sistema deve garantir sequência lógica de estados.

RNFE-007 - Segurança: O entregador deve ver apenas entregas atribuídas à sua conta.

- RNFE-007.1 — O entregador deve ver apenas entregas atribuídas à sua conta.
- RNFE-007.2 — O sistema deve bloquear acesso a detalhes por ID não atribuído.

RNFE-008 - Privacidade: O sistema deve minimizar exposição de dados pessoais do utilizador.

- RNFE-008.1 — O sistema deve mostrar apenas dados necessários para a entrega.
- RNFE-008.2 — O sistema deve evitar expor dados sensíveis fora do contexto do detalhe da entrega.

3.1.3 Requisitos do Fornecedor

Requisitos Funcionais

RFF-001 - Autenticação: O sistema deve permitir autenticação do fornecedor para acesso ao painel.

- RFF-001.1 — O sistema deve permitir login com credenciais de fornecedor.
- RFF-001.2 — Sessões devem expirar por inatividade.

RFF-002 - Criar produtos: O sistema deve permitir criar produtos no catálogo.

- RFF-002.1 — Criar produto deve incluir nome, preço, imagem e descrição.
- RFF-002.2 — O sistema deve validar campos obrigatórios antes de guardar.

RFF-003 - Editar produtos: O sistema deve permitir editar informação de produtos.

- RFF-003.1 — Editar produto deve permitir alterar preço, imagem e descrição.
- RFF-003.2 — O sistema deve guardar alterações e mostrar confirmação.

RFF-004 - Desativar produtos: O sistema deve permitir desativar/remover produtos do catálogo.

- RFF-004.1 — Desativar produto deve removê-lo da listagem do utilizador sem apagar histórico.
- RFF-004.2 — O sistema deve permitir reativar produto desativado.

RFF-005 - Atualizar stock: O sistema deve permitir atualizar stock por produto.

- RFF-005.1 — O fornecedor deve atualizar stock por valor absoluto ou ajuste.
- RFF-005.2 — O sistema deve refletir stock atualizado nos produtos visíveis ao utilizador.

RFF-006 - Validação de stock: O sistema deve impedir valores de stock inválidos (ex.: negativos).

- RFF-006.1 — O sistema deve impedir stock negativo.
- RFF-006.2 — O sistema deve validar números e rejeitar valores inválidos.

RFF-007 - Listagem de pedidos: O sistema deve listar pedidos recebidos.

- RFF-007.1 — A lista de pedidos deve estar ordenada por data/hora (mais recente primeiro).
- RFF-007.2 — Cada pedido deve mostrar estado e total resumido.

RFF-008 - Aceitar pedido: O sistema deve permitir aceitar um pedido.

- RFF-008.1 — O fornecedor deve conseguir aceitar pedido com um clique.
- RFF-008.2 — Ao aceitar, o pedido passa a estado "Aceite/Em preparação".

RFF-009 - Recusar pedido: O sistema deve permitir recusar um pedido com motivo.

- RFF-009.1 — Ao recusar, o sistema deve exigir motivo (categoria ou texto curto).
- RFF-009.2 — O pedido recusado deve ficar marcado e não avançar para preparação.

RFF-010 - Atualizar estado do pedido: O sistema deve permitir atualizar estado do pedido (ex.: em preparação, pronto).

- RFF-010.1 — O fornecedor deve marcar estado "Em preparação" e "Pronto".
- RFF-010.2 — O sistema deve registar data/hora de cada mudança de estado.

RFF-011 - Identificador de expedição: O sistema deve permitir gerar/mostrar identificador do pedido para expedição.

- RFF-011.1 — O sistema deve apresentar o identificador do pedido para expedição/entrega.
- RFF-011.2 — O identificador deve ser visível no detalhe e na lista de pedidos.

RFF-012 - Histórico de pedidos: O sistema deve permitir consultar histórico de pedidos processados.

- RFF-012.1 — O histórico deve permitir consultar pedidos concluídos/recusados.
- RFF-012.2 — Deve ser possível filtrar histórico por período e estado.

Requisitos Não Funcionais

RNFF-001 - Eficiência da interface: A interface do fornecedor deve ser eficiente para gestão de produtos e pedidos.

- RNFF-001.1 — Ações principais devem ser possíveis com poucos cliques.
- RNFF-001.2 — A interface deve manter consistência de botões, cores e labels.

RNFF-002 - Feedback de ações: O sistema deve fornecer feedback claro após ações.

- RNFF-002.1 — Após cada ação, o sistema deve mostrar confirmação visível.

- RNFF-002.2 — Em erro, o sistema deve indicar causa e como resolver.

RNFF-003 - Performance de listas: O sistema deve carregar listas de produtos/pedidos rapidamente.

- RNFF-003.1 — Listas de produtos e pedidos devem abrir rapidamente em condições normais.
- RNFF-003.2 — Pesquisa e filtros devem responder sem bloqueios visíveis.

RNFF-004 - Suporte a catálogos grandes: O sistema deve suportar catálogos grandes sem degradação significativa.

- RNFF-004.1 — O sistema deve suportar muitos produtos sem falhas ou lentidão extrema.
- RNFF-004.2 — O sistema deve otimizar imagens para não degradar o desempenho.

RNFF-005 - Segurança de acesso: O fornecedor deve aceder apenas aos seus próprios produtos e pedidos.

- RNFF-005.1 — O fornecedor deve ver apenas os seus produtos e pedidos.
- RNFF-005.2 — O sistema deve impedir acesso direto por ID a recursos de outros fornecedores.

RNFF-006 - Auditoria: O sistema deve registar alterações críticas (stock, preço, estado de pedido).

- RNFF-006.1 — O sistema deve registar alterações de stock com data/hora e utilizador responsável.
- RNFF-006.2 — O sistema deve registar mudanças de estado de pedido.

RNFF-007 - Consistência entre stock e pedidos: O sistema deve evitar inconsistências entre stock e pedidos aceites.

- RNFF-007.1 — O sistema deve impedir aceitar pedido se stock for insuficiente.
- RNFF-007.2 — O sistema deve manter consistência entre pedidos aceites e stock reservado/atualizado.

RNFF-008 - Privacidade: O sistema deve expor apenas os dados necessários para preparar/expedir pedidos.

- RNFF-008.1 — O sistema deve expor apenas dados necessários para preparação/expedição.
- RNFF-008.2 — O sistema deve minimizar dados pessoais visíveis em listagens.

3.1.4 Descrição detalhada dos requisitos principais

Utilizador

RF-012 - Apenas produtos em stock

Objetivo: Garantir que o utilizador nunca adiciona ao carrinho produtos indisponíveis.

Dependências:

- RFF-005 / RFF-006 — stock depende das atualizações do fornecedor
- RF-024 — carrinho só aceita produtos com stock válido
- RF-019 — stock revalidado ao repetir pedidos

Critérios de aceitação:

- Produtos com stock 0 não aparecem na listagem por defeito
- Produtos sem stock apresentados estão marcados como "Indisponível" com botão desativado
- Stock revalidado antes de confirmar o pedido
- Se produto ficar sem stock entre adição e checkout, é removido com aviso

RF-014 / RF-015 - Métodos de pagamento e pagamento em dinheiro

Objetivo: Permitir que utilizadores sem meios digitais consigam sempre concluir uma compra.

Dependências:

- RFE-008 — entregador recebe indicação do método e confirma recebimento em dinheiro
- RF-024 — método de pagamento associado ao pedido na confirmação

Critérios de aceitação:

- Apenas um método selecionável por pedido
- "Dinheiro na entrega" não solicita dados bancários
- Confirmação explícita de pagamento no ato da entrega
- Método registado no pedido e visível pelo entregador
- Métodos indisponíveis aparecem desativados com motivo

RF-035 / RF-036 / RF-037 - Receitas médicas e medicamentos prescritos

Objetivo: Permitir requisição de medicamentos prescritos sem deslocação à farmácia.

Dependências:

- RF-034 — botão "Requisitar produtos" inicia o fluxo
- RF-024 — medicamentos aprovados adicionados ao carrinho existente
- RFF-010 — farmácia valida a receita e atualiza estado do pedido

Critérios de aceitação:

- Submissão bloqueada se faltar CC ou receita
- Utilizador pode rever e refazer cada fotografia antes de submeter
- Após submissão, estado "Em análise" apresentado
- Erro no envio mostra opção "Tentar novamente" sem perder fotografias
- Medicamentos indisponíveis informados individualmente e não adicionados

RF-017 / RF-018 - Histórico e repetição de pedidos

Objetivo: Reduzir esforço cognitivo ao permitir repetir compras frequentes rapidamente.

Dependências:

- RF-019 — repetição implica validação de stock
- RF-023 / RF-024 — repetição cria novo carrinho
- RFF-005 — stock atual consultado no momento da repetição

Critérios de aceitação:

- Histórico ordenado do mais recente para o mais antigo
- Carrinho atual substituído apenas com confirmação
- Total recalculado com preços atuais
- Produtos sem stock removidos com aviso
- Resumo de alterações apresentado antes de confirmar

Entregador

RFE-003 a RFE-006 - Ciclo de estados da entrega

Objetivo: Garantir gestão simples e rastreável do ciclo de vida de cada entrega.

Dependências:

- RFE-001 / RFE-002 — entregador precisa de ver e aceder ao detalhe antes de aceitar
- RFE-008 — confirmação de pagamento ocorre na conclusão
- RFE-009 — prova de entrega registada ao marcar "Entregue"
- RFF-010 — estado "Pronto" do fornecedor precede atribuição ao entregador
- RNFE-006 — sistema garante sequência lógica e impede duplicações

Critérios de aceitação:

- Só é possível aceitar entrega no estado "Atribuída"
- Só é possível marcar "Em percurso" após aceitar
- Só é possível marcar "Entregue" ou "Falhada" após "Em percurso"
- Cada transição regista hora automaticamente
- "Falhada" exige motivo obrigatório
- Impossível marcar "Entregue" duas vezes

RFE-008 - Confirmação de pagamento em dinheiro

Objetivo: Garantir que o entregador cobra o valor correto e regista a confirmação.

Dependências:

- RF-015 — utilizador seleciona "Dinheiro na entrega" no checkout
- RFE-005 — confirmação de pagamento acompanha o estado "Entregue"
- RFE-002 — método de pagamento visível no detalhe da entrega

Critérios de aceitação:

- Valor a cobrar apresentado de forma destacada quando método é dinheiro
- Botão "Confirmar pagamento recebido" visível apenas para pagamento em dinheiro
- Impossível marcar "Entregue" sem confirmar pagamento quando método é dinheiro
- Campo de confirmação não aparece para pagamentos digitais

Fornecedor

RFF-005 / RFF-006 - Gestão e validação de stock

Objetivo: Manter stock atualizado em tempo real para evitar pedidos inválidos.

Dependências:

- RF-012 — visibilidade de produtos no lado do utilizador depende do stock
- RFF-008 / RNFF-007 — ao aceitar pedido, stock é revalidado e decrementado
- RNFF-006 — alterações de stock registadas para auditoria

Critérios de aceitação:

- Atualização por valor absoluto ou ajuste (ex.: +5, -3)
- Valores negativos ou não numéricos rejeitados com mensagem clara
- Stock atualizado refletido imediatamente na listagem do utilizador
- Cada alteração registada com data/hora e utilizador responsável
- Impossível aceitar pedido com stock insuficiente

RFF-008 / RFF-009 / RFF-010 - Ciclo de vida do pedido

Objetivo: Permitir gestão completa do pedido desde receção até expedição.

Dependências:

- RFF-005 / RFF-006 — stock validado e decrementado ao aceitar
- RFF-011 — identificador de expedição gerado após aceitação
- RFE-003 — entregador só recebe entrega após pedido "Pronto"
- RNFF-006 — mudanças de estado registadas para auditoria
- RNFF-007 — impossível aceitar pedido com stock insuficiente

Critérios de aceitação:

- Aceitação com um clique passa pedido a "Aceite/Em preparação"
- Recusa exige motivo obrigatório e bloqueia avanço para preparação
- Sequência obrigatória: Aceite → Em preparação → Pronto
- Data/hora registada em cada mudança de estado
- Identificador de expedição gerado e visível após aceitação

3.1.5 Casos de Uso/*User Stories*

Caso de Uso 1: Registo de Cliente

Atores: Cliente

Objetivo: Criar uma conta de forma simples.

Fluxo Principal:

1. O cliente toca em "Criar Conta"

2. A app solicita nome, email, telefone e endereço
3. O cliente insere as informações
4. A app envia código de verificação por SMS
5. O cliente digita o código
6. A conta é criada com sucesso

Fluxo Alternativo:

- O cliente já possui conta → redirecionado para login

Caso de Uso 2: Registo de Entregador

Atores: Entregador

Objetivo: Criar conta como profissional de entrega.

Fluxo Principal:

1. O entregador acede a "Registar como Entregador"
2. Insere dados pessoais e informações do veículo
3. Sistema valida documentos (carta de condução, seguro)
4. Código SMS enviado para confirmação
5. Entregador digita código
6. Conta ativada com permissão para aceitar entregas

Fluxo Alternativo:

- Documentos inválidos → rejeitado com indicação do motivo

Caso de Uso 3: Registo de Fornecedor

Atores: Fornecedor

Objetivo: Registar farmácia/loja no sistema.

Fluxo Principal:

1. Fornecedor acede a "Registar Farmácia/Loja"
2. Insere dados da empresa e localização
3. Envia documentos de licença comercial
4. Sistema valida informações
5. Código SMS enviado
6. Conta de fornecedor criada e painel acessível

Fluxo Alternativo:

- Documentos incompletos → solicitação de reenvio

Caso de Uso 4: Criar Produto no Catálogo

Atores: Fornecedor

Objetivo: Adicionar novo produto ao catálogo.

Fluxo Principal:

1. Fornecedor acede ao painel e seleciona "Novo Produto"
2. Insere nome, descrição, preço e categoria
3. Define stock inicial
4. Adiciona imagem do produto
5. Sistema valida dados
6. Produto criado e disponível para compra

Fluxo Alternativo:

- Dados inválidos (ex.: preço negativo) → exibe erro e pede correção

Caso de Uso 5: Editar Informação de Produto

Atores: Fornecedor

Objetivo: Modificar dados de um produto existente.

Fluxo Principal:

1. Fornecedor seleciona produto no catálogo
2. Toca em "Editar"
3. Modifica preço, descrição ou categoria
4. Confirma alterações
5. Sistema atualiza produto
6. Alteração é registada no histórico

Fluxo Alternativo:

- Produto já foi comprado → aviso de que alterações afetam pedidos futuros

Caso de Uso 6: Atualizar Stock de Produto

Atores: Fornecedor

Objetivo: Modificar quantidade disponível de um produto.

Fluxo Principal:

1. Fornecedor acede à gestão de stock
2. Seleciona produto
3. Insere nova quantidade
4. Sistema valida se valor é válido (não negativo)
5. Stock atualizado
6. Feedback de sucesso exibido

Fluxo Alternativo:

- Quantidade negativa inserida → sistema rejeita e exibe mensagem de erro
- Stock abaixo do mínimo definido → alerta de stock crítico enviado

Caso de Uso 7: Aceitar Pedido

Atores: Fornecedor, Cliente

Objetivo: Fornecedor aceita pedido do cliente.

Fluxo Principal:

1. Fornecedor recebe notificação de novo pedido
2. Revê detalhes e stock disponível
3. Toca em "Aceitar Pedido"
4. Estado muda para "Aceite"
5. Sistema reserva stock
6. Cliente recebe notificação de confirmação

Fluxo Alternativo:

- Stock insuficiente → fornecedor oferece alternativa ou rejeita

Caso de Uso 8: Recusar Pedido com Motivo

Atores: Fornecedor, Cliente

Objetivo: Fornecedor rejeita pedido indicando razão.

Fluxo Principal:

1. Fornecedor revê pedido
2. Seleciona "Recusar"
3. Escolhe motivo (stock indisponível, horário fechado, etc.)

4. Pode adicionar mensagem personalizada
5. Pedido marcado como "Recusado"
6. Cliente notificado com razão

Fluxo Alternativo:

- Cliente pode tentar novo pedido após recusa

Caso de Uso 9: Atualizar Estado de Pedido

Atores: Fornecedor, Entregador, Cliente

Objetivo: Acompanhar progresso do pedido.

Fluxo Principal:

1. Fornecedor aceita pedido (estado: "Aceite")
2. Começa preparação (estado: "Em Preparação")
3. Finaliza preparação (estado: "Pronto")
4. Entregador recolhe (estado: "Em Trânsito")
5. Entrega realizada (estado: "Entregue")
6. Cliente confirma receção

Fluxo Alternativo:

- Problema durante preparação → estado "Cancelado" com notificação

Caso de Uso 10: Consultar Histórico de Pedidos

Atores: Fornecedor, Cliente

Objetivo: Visualizar pedidos anteriores.

Fluxo Principal:

1. Utilizador acede à secção "Histórico"
2. Sistema exibe lista de pedidos processados
3. Pode filtrar por data, estado ou valor
4. Seleciona pedido para ver detalhes
5. Informações completas exibidas
6. Pode exportar em PDF se necessário

Fluxo Alternativo:

- Sem pedidos no histórico → mensagem informativa exibida

Caso de Uso 11: Gerar Identificador de Expedição

Atores: Fornecedor, Entregador

Objetivo: Criar código para rastreio da entrega.

Fluxo Principal:

1. Fornecedor marca pedido como "Pronto"
2. Sistema gera identificador único (QR code ou código numérico)
3. Fornecedor exibe/imprime identificador
4. Entregador scaneia ou insere código
5. Sistema confirma correspondência
6. Entrega pode prosseguir

Fluxo Alternativo:

- Código inválido → entregador notificado de erro

Caso de Uso 12: Efetuar Compra com Carrinho

Atores: Cliente, Fornecedor

Objetivo: Cliente compra múltiplos produtos de um ou vários fornecedores.

Fluxo Principal:

1. Cliente adiciona produtos ao carrinho
2. Revê itens e preços
3. Procede ao checkout
4. Insere endereço de entrega
5. Confirma pagamento
6. Pedidos enviados aos fornecedores correspondentes
7. Cliente recebe confirmação

Fluxo Alternativo:

- Produto sai de stock → cliente notificado para remover ou substituir
- Pagamento falha → sistema oferece nova tentativa

Caso de Uso 13: Entregador Recolhe e Entrega Pedido

Atores: Entregador, Cliente, Fornecedor

Objetivo: Entregador completa ciclo de entrega.

Fluxo Principal:

1. Entregador recebe notificação de pedido pronto
2. Vai até fornecedor e recolhe pacote
3. Scaneia identificador de expedição
4. Sistema confirma recolha
5. Entregador viaja até cliente
6. Scaneia código no local de entrega
7. Cliente assina/confirma receção
8. Pedido marcado como "Entregue"

Fluxo Alternativo:

- Cliente não está em casa → entregador agenda nova tentativa
- Pacote danificado → relatório enviado ao fornecedor

Caso de Uso 14: Visualizar Dashboard de Vendas

Atores: Fornecedor

Objetivo: Acompanhar desempenho e métricas.

Fluxo Principal:

1. Fornecedor acede ao painel principal
2. Dashboard exibe resumo de vendas
3. Gráficos mostram produtos mais vendidos
4. Alertas de stock crítico visíveis

Fluxo Alternativo:

- Sem dados disponíveis → mensagem "Sem vendas neste período"

Caso de Uso 15: Visualizar Perfil e Histórico Pessoal

Atores: Cliente, Entregador, Fornecedor

Objetivo: Aceder a dados e histórico pessoal.

Fluxo Principal:

1. Utilizador toca em "Perfil"
2. Visualiza dados pessoais (nome, telefone, endereço)
3. Pode editar informações

4. Acede a histórico de transações
5. Consulta avaliações/feedback
6. Pode desativar conta se desejar

Fluxo Alternativo:

- Dados desatualizados → utilizador atualiza e confirma por SMS

3.2 Modelação

Users — dados comuns a todos os tipos de utilizador

Tabela 2 - Tabela User

Campo	Tipo	Descrição
id	**STRING (PK)**	**UID gerado pelo Firebase Auth**
nome	**STRING**	**Nome do utilizador**
email	**STRING**	**Email de autenticação**
telefone	**STRING**	**Contacto telefónico**
foto_base64	**STRING**	**Fotografia em base64**
tipo	**STRING**	**'cliente' | 'entregador' | 'fornecedor'**
createdAt	**TIMESTAMP**	**Data de registo**

Clientes — dados específicos dos utilizadores do tipo Cliente

Tabela 3 - Tabela Cliente

Campo	Tipo	Descrição
id	**STRING (PK)**	**Identificador único**

Campo	Tipo	Descrição
userId	STRING (FK → users)	Referência ao utilizador
morada	STRING	Morada de entrega

Entregadores — dados específicos dos utilizadores do tipo Entregador

Tabela 4 - Tabela Entregador

Campo	Tipo	Descrição
id	STRING (PK)	Identificador único
userId	STRING (FK → users)	Referência ao utilizador
veiculo	STRING	Tipo de veículo
matricula	STRING	Matrícula do veículo

Fornecedores — dados específicos dos utilizadores do tipo Fornecedor

Tabela 5 - Tabela Fornecedor

Campo	Tipo	Descrição
id	STRING (PK)	Identificador único
userId	STRING (FK → users)	Referência ao utilizador
nomeEmpresa	STRING	Nome da empresa
morada	STRING	Morada da empresa

Campo	Tipo	Descrição
nif	STRING	NIF da empresa
tipoLoja	STRING	Tipo de loja (ex.: supermercado, farmácia)

Produtos

Tabela 6 - Tabela Produtos

Campo	Tipo	Descrição
id	STRING (PK)	Identificador único
fornecedorId	STRING (FK → users)	Fornecedor responsável
nome	STRING	Nome do produto
preco	DOUBLE	Preço unitário
quantidade	INT	Stock disponível
tipo	STRING	'loja' | 'farmacia'

Pedidos

Tabela 7 - Tabela Pedidos

Campo	Tipo	Descrição
id	STRING (PK)	Identificador único
clienteId	STRING (FK → users)	Cliente que fez o pedido
entregadorId	STRING (FK → users)	Entregador atribuído

Campo	Tipo	Descrição
data	**TIMESTAMP**	**Data do pedido**
estado	**STRING**	**'criado' | 'aceite' | 'em preparação' | 'pronto' | 'expedido'**
moradaLoja	**STRING**	**Ponto de partida**
moradaCliente	**STRING**	**Ponto de destino**
codigo	**STRING**	**Identificador de expedição**
metodoPagamento	**STRING**	**Método de pagamento escolhido**

Pedido_Items

Armazena os produtos de cada pedido como um snapshot (nome, preço, quantidade), garantindo o histórico mesmo que os dados originais do produto mudem.

Tabela 8 - Tabela Pedidos_Itens

Campo	Tipo	Descrição
id	**STRING (PK)**	**Identificador único**
pedidoId	**STRING (FK → pedidos)**	**Pedido a que pertence**
produtoId	**STRING**	**Referência ao produto**
nome	**STRING**	**Nome no momento da compra**
preco	**DOUBLE**	**Preço no momento da compra**
quantidade	**INT**	**Quantidade encomendada**

O modelo apresentado encontra-se já numa fase final de desenvolvimento, refletindo uma estrutura consolidada e alinhada com os requisitos definidos para a solução. Nesta etapa, as

alterações esperadas serão pontuais e de natureza incremental, podendo surgir pequenos ajustes ou otimizações decorrentes da implementação ou validação prática.

Ainda assim, não se antecipam mudanças estruturais significativas, mantendo-se o modelo estável e adequado aos objetivos do projeto.

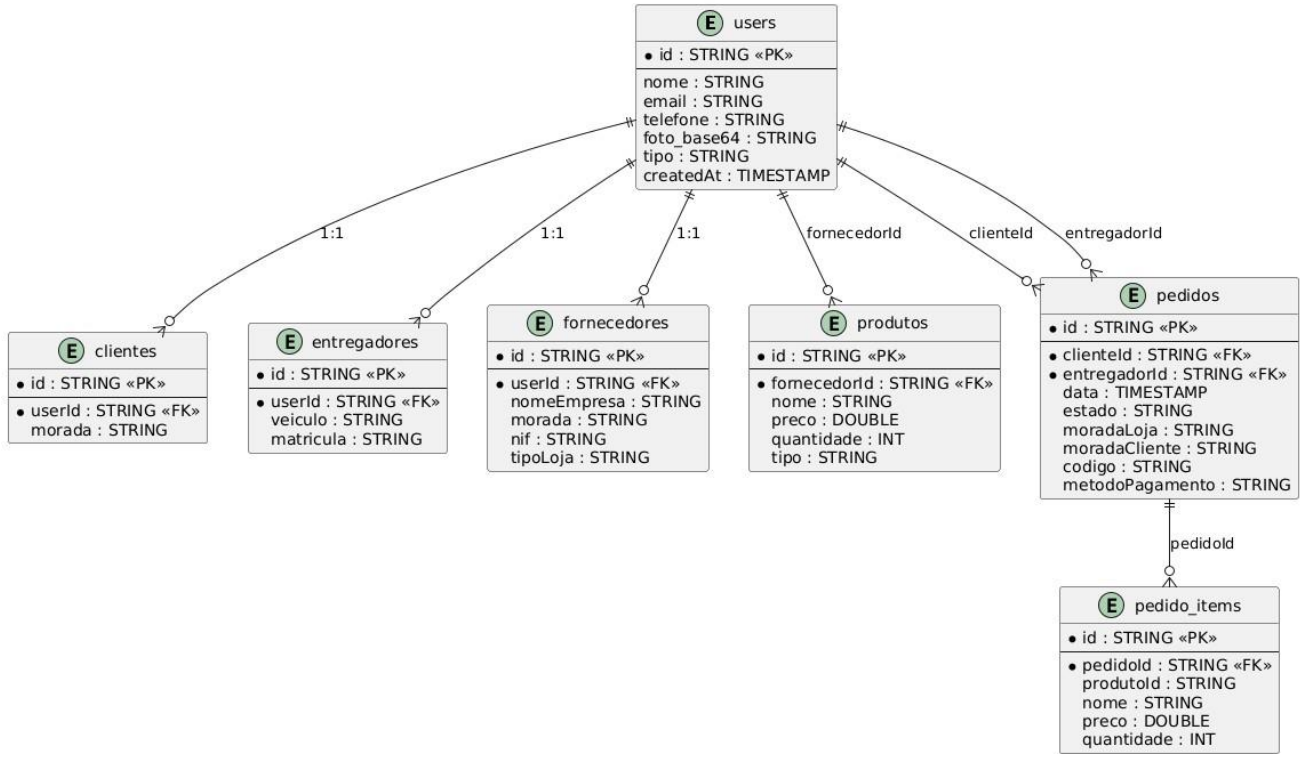

Figura 1 - Modelo BD

3.3 Protótipos de Interface/Mapa aplicacional

Os protótipos de interface apresentados na primeira entrega tinham como objetivo ilustrar, de forma preliminar, a estrutura e navegação da aplicação numa fase em que ainda não existia implementação.

Nesta fase do projeto, esses protótipos encontram-se superados pela evolução do desenvolvimento, sendo substituídos pelos ecrãs reais da aplicação, que refletem com maior fidelidade as decisões tomadas durante a implementação.

Mapa Aplicacional

O mapa aplicacional apresenta a estrutura de navegação da aplicação organizada por perfil de utilizador, ilustrando os ecrãs disponíveis e os fluxos de interação para cada tipo de conta.

O ponto de entrada comum a todos os utilizadores é o ecrã de **Splash / Login**, a partir do qual é possível registar conta ou recuperar senha. Após autenticação, o utilizador é direcionado para um ecrã de **seleção de papel**, que encaminha para a área correspondente ao seu perfil — Cliente, Entregador ou Fornecedor. Existe ainda um ecrã partilhado de **edição de perfil**, acessível a todos os tipos de utilizador.

Área do Cliente

O fluxo do cliente organiza-se em quatro módulos principais:

- O módulo de **Explorar** permite navegar entre lojas, aceder aos detalhes de cada estabelecimento e consultar o detalhe de produtos.
- O módulo de **Encomenda** cobre o processo de compra completo, desde o carrinho e checkout até à confirmação de pagamento, rastreamento da encomenda e avaliação da experiência.
- O módulo de **Histórico** permite consultar pedidos anteriores, ver o detalhe de cada um e repetir encomendas.
- O módulo de **Perfil** dá acesso às definições, edição de dados pessoais e logout.

Área do Entregador

O fluxo do entregador divide-se em três módulos:

- O módulo de **Pedidos** apresenta os pedidos disponíveis para recolha, com detalhe de cada pedido e opção de aceitar ou recusar, podendo cancelar antes de confirmar.
- O módulo de **Entrega Ativa** suporta a navegação via Mapbox GPS, o acompanhamento da entrega em curso, a confirmação de entrega e o resumo final.
- O módulo de **Histórico / Perfil** agrega o histórico de entregas realizadas, estatísticas de ganhos, edição de perfil e logout.

Área do Fornecedor

O fluxo do fornecedor organiza-se em quatro módulos:

- O módulo de **Loja** permite gerir as informações da loja, consultar dados do estabelecimento e aceder à análise de vendas.
- O módulo de **Produtos** cobre a gestão do catálogo, listagem de produtos, adição e edição de produtos e gestão de categorias.
- O módulo de **Pedidos** permite consultar pedidos recebidos, ver o detalhe de cada um, confirmar ou rejeitar e marcar como pronto para entrega.
- O módulo de **Perfil** dá acesso à edição de dados e logout.

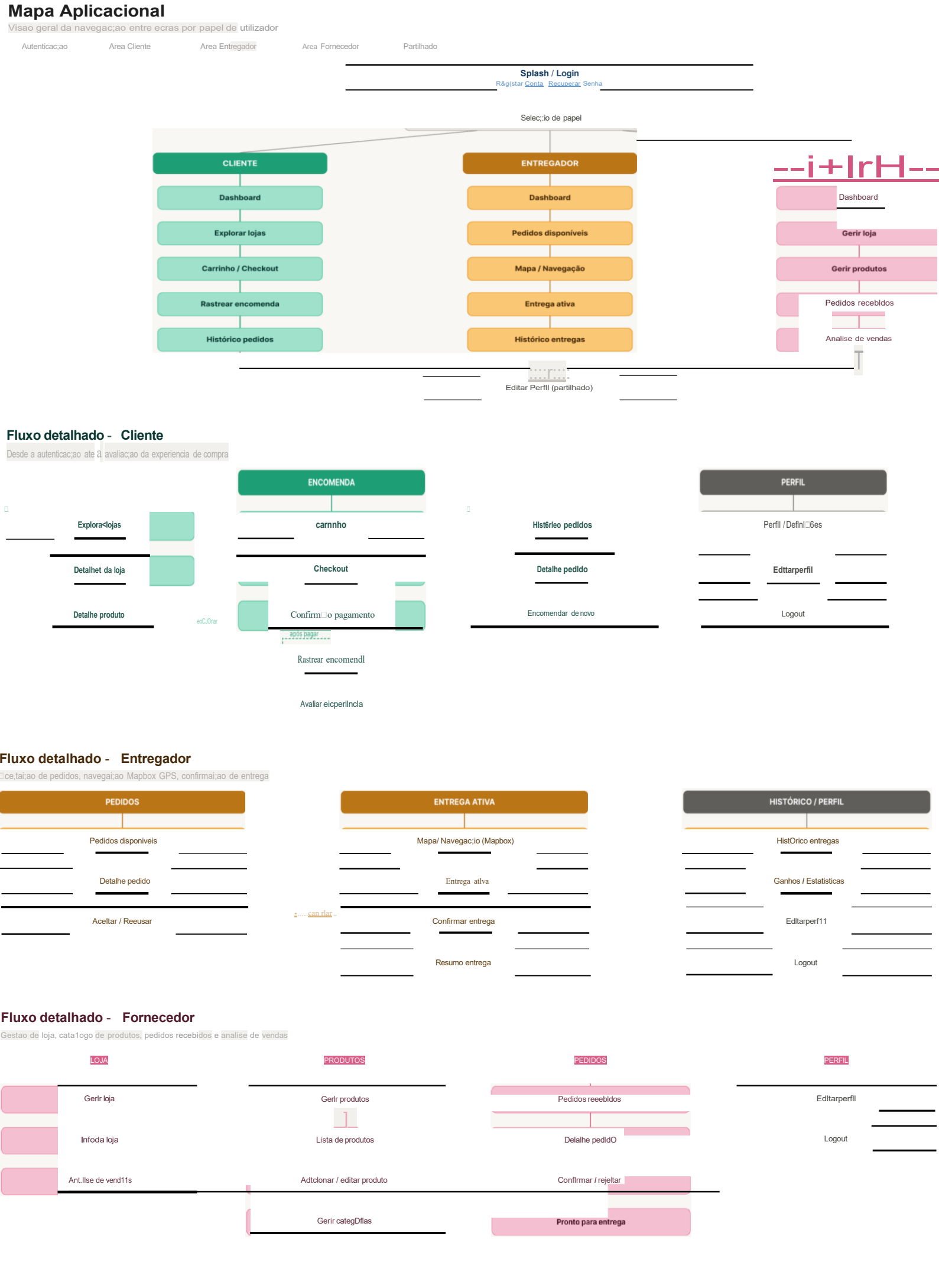

Figura 2 - Mapa Aplicacional

4 Solução Proposta

4.1 Apresentação

A solução proposta consiste numa aplicação móvel destinada a diferentes perfis de utilizador — cliente, fornecedor e entregador — desenvolvida para servir populações com baixa literacia digital residentes em zonas remotas.

A aplicação permite realizar compras de bens essenciais, requisitar medicamentos, gerir catálogos e stocks, e acompanhar entregas, tudo de forma simples e intuitiva.

A aplicação é organizada nos seguintes módulos principais:

Cliente

- Navegação Simplificada
- Interação com Lojas
- Gestão de Carrinho de Compras
- Histórico de Pedidos
- Requisição de Medicamentos
- Centro de Ajuda

Fornecedor

- Gestão de Catálogo
- Gestão de Stock
- Gestão de Pedidos
- Expedição

Entregador

- Gestão de Entregas
- Navegação e Mapa
- Comunicação e Incidentes
- Histórico de Entregas

Cada módulo foi definido com base nos requisitos funcionais e não funcionais identificados anteriormente.

Nesta fase do projeto, a solução encontra-se em desenvolvimento, tendo sido então concluído a fase de especificação e modelação.

Dado a tratar-se de um relatório intercalar, esta definição encontra-se numa fase avançada, podendo ainda sofrer ajustes pontuais em função das necessidades identificadas durante as próximas etapas.

As secções seguintes apresentam a arquitetura implementada, as tecnologias e as ferramentas selecionadas para o desenvolvimento.

Link demo vídeo: https://www.youtube.com/watch?v=Q6qRmlFFcH0

4.2 Arquitetura

Nesta fase, a arquitetura evoluiu de uma proposta conceptual para uma implementação funcional. A aplicação encontra-se agora integrada com o Firebase, que serve de backend e base de dados em tempo real, substituindo a abordagem cliente-centrada descrita na entrega anterior.

A lógica da aplicação está distribuída por camadas bem definidas, com o Firestore a assegurar o armazenamento persistente de dados e o Firebase Authentication a gerir a autenticação dos diferentes tipos de utilizador.

A arquitetura adotada segue uma abordagem em camadas, onde cada camada tem responsabilidades bem delimitadas e comunica apenas com as camadas adjacentes, promovendo a separação de responsabilidades, a manutenibilidade do código e a escalabilidade da solução.

- **Camada de Modelos:** Estruturas de dados partilhadas entre todas as camadas.
- **Camada de Base de Dados:** Armazenamento persistente de dados através do Firebase/Firestore.
- **Camada de Serviços:** Lógica de negócio e acesso a dados.
- **Camada de Repositórios:** Gestão de estado.
- **Camada de UI:** Apresentação e interação com os utilizadores.

4.3 Tecnologias e Ferramentas Utilizadas

Nesta fase de desenvolvimento, já se encontram bem definidas as tecnologias a serem usadas no desenvolvimento do projeto:

- **Flutter:** Framework da Google que permite criar aplicações móveis multiplataforma (Android e iOS) a partir de uma única base de código, produzindo interfaces fluidas e responsivas com a linguagem Dart.
- **Dart:** Linguagem de programação utilizada pelo Flutter, com tipagem estática e estrutura orientada a objetos, adequada à arquitetura em camadas adotada.
- **Firebase:** Plataforma da Google que serve de backend à aplicação, dispensando a necessidade de desenvolver e manter um servidor próprio. Os serviços utilizados são:
    - **Authentication:** Gestão de autenticação para os três perfis de utilizador: cliente, fornecedor e entregador.
    - **Cloud Firestore:** Base de dados NoSQL em tempo real, escolhida pela sua flexibilidade, sincronização automática e escalabilidade.
    - **Storage:** Armazenamento de ficheiros como imagens de produtos e fotografias de prova de entrega.

- **Android Studio:** Ambiente de desenvolvimento integrado (IDE) utilizado para desenvolver, testar e depurar a aplicação, com suporte nativo ao Flutter e emulação de dispositivos Android.
- **GitHub:** Plataforma de controlo de versões utilizada para gerir o código fonte do projeto, permitindo o acompanhamento do histórico de alterações e a organização do desenvolvimento.

4.4 Ambientes de Teste e de Produção

4.4.1 Ambiente de Teste

O ambiente de teste utilizado durante o desenvolvimento da solução Longe com Tudo é composto por um emulador Android gerido pelo Android Studio, complementado pela estação de trabalho do desenvolvedor que serve de host. Não foram utilizados dispositivos físicos dedicados em fase de teste.

4.4.1.1 Estação de Trabalho (Host do Emulador)

O emulador Android corre sobre a máquina do desenvolvedor, que disponibiliza os recursos de hardware reais ao processo de emulação. Os requisitos mínimos recomendados para suportar o emulador com aceleração por hardware são:

Recurso	Especificação Utilizada
Sistema Operativo	Windows 10/11 (64-bit)
CPU (host)	x86_64 com suporte a Intel HAXM ou Hyper-V
RAM (host)	≥ 16 GB recomendado (8 GB mínimo)
Espaço em disco	≥ 20 GB (SDK Android + AVD + projeto)
Placa gráfica	Com suporte a OpenGL 2.0+ (para hw.gpu.mode=host)
Conectividade	Acesso à internet para Firebase, Mapbox e pub.dev
IDE	Android Studio (com Flutter e Dart plugins)
Flutter SDK	Canal stable

Dart SDK	Incluído no Flutter SDK

4.4.1.2 Emulador Android (AVD)

O dispositivo virtual utilizado para testes funcionais é um Pixel 9 Pro XL emulado, configurado com as seguintes especificações extraídas do ficheiro config.ini do AVD:

Parâmetro	Valor
Perfil de dispositivo	Google Pixel 9 Pro XL
Arquitetura CPU (emulada)	x86_64
Número de cores (vCPU)	4
RAM	2048 MB (2 GB)
Heap da VM Dalvik/ART	256 MB
Armazenamento interno (data)	12 GB
Cartão SD (emulado)	512 MB
Resolução do ecrã	1344 × 2992 px
Densidade do ecrã	480 dpi (xxhdpi)
Versão Android (API)	Android 16 (API 36)
Imagem do sistema	Google APIs + Play Store (x86_64)
Aceleração gráfica	Aceleração por host (hw.gpu.mode=host)
GPS	Emulado (hw.gps=yes)
Sensores	Acelerómetro, giroscópio, magnetómetro, luz, pressão, proximidade

Câmara traseira	Cena virtual (virtualscene)
Câmara frontal	Emulada
Latência de rede (emulada)	Nenhuma (runtime.network.latency=none)
Velocidade de rede (emulada)	Máxima (runtime.network.speed=full)
Google Play Store	Ativo

4.4.1.3 Serviços Backend em Teste

Durante a fase de testes, a aplicação liga-se diretamente ao projeto Firebase de produção (não existe um ambiente Firebase separado de staging). As operações de teste sobre o Firestore e a Firebase Authentication são realizadas com contas e dados de teste criados especificamente para esse fim, sem impacto nos dados reais de utilizadores.

- Firebase Authentication: contas de teste para os três perfis (cliente, entregador, fornecedor).
- Cloud Firestore: coleções de teste com lojas, produtos e encomendas fictícias.
- Mapbox SDK: token de acesso configurado via flutter_dotenv (.env não incluído no controlo de versões).
- Conectividade: testada com ConnectivityProvider simulando modo offline através da desativação do adaptador de rede do emulador.

4.4.2 Ambiente Produtivo

O ambiente produtivo da solução Longe com Tudo assenta numa arquitetura cliente-servidor distribuída, combinando uma aplicação móvel nativa (Flutter) com uma plataforma de backend gerida em cloud (Firebase / Google Cloud Platform). A solução não requer infraestrutura própria de servidores, recorrendo integralmente a serviços geridos de terceiros para garantir escalabilidade, disponibilidade e segurança.

4.4.2.1 Dispositivos Cliente (Aplicação Móvel)

A aplicação móvel é distribuída através das plataformas Google Play Store (Android) e Apple App Store (iOS). Os requisitos mínimos de hardware e software para execução produtiva são:

Requisito	Android	iOS

Versão mínima do SO	Android 6.0 (API 23)	iOS 13.0
Processador	ARMv7 / ARM64, ≥ 1,2 GHz	Apple A9 ou superior
Memória RAM	≥ 2 GB	≥ 2 GB
Armazenamento (instalação)	≈ 60–80 MB	≈ 60–80 MB
Armazenamento (cache local)	≈ 50–150 MB	≈ 50–150 MB
Conectividade de rede	Wi-Fi ou dados móveis ≥ 5 Mbps	Wi-Fi ou dados móveis ≥ 5 Mbps
GPS / Localização	Obrigatório (hardware)	Obrigatório (hardware)
Câmara	Opcional (fotografia de perfil)	Opcional (fotografia de perfil)

4.4.2.2 Backend e Serviços Cloud (Firebase / GCP)

O backend é inteiramente suportado pela plataforma Firebase, gerida pela Google Cloud Platform. Não existe qualquer servidor físico ou virtual gerido pela equipa. Os serviços Firebase utilizados em produção são:

Serviço Firebase	Função	Recursos / Limites
Firebase Authentication	Autenticação (e-mail/password) dos três perfis de utilizador	Ilimitado (sem custo adicional)
Cloud Firestore	Base de dados NoSQL em tempo real (utilizadores, lojas, produtos, encomendas)	1 GiB gratuito; além disso $0,18/GiB/mês
Firebase Cloud Messaging	Notificações push para entregadores e clientes	Sem custo

Firebase Hosting (opcional)	Distribuição de assets estáticos ou painel admin web	10 GB/mês gratuitos

A escalabilidade dos recursos de computação e armazenamento é gerida automaticamente pela GCP, sem necessidade de dimensionamento manual.

4.4.2.3 Recursos de Rede

A comunicação entre o cliente e os serviços Firebase é realizada via HTTPS (TLS 1.2+) e WebSockets (sincronização em tempo real do Firestore). Os requisitos estimados de largura de banda por tipo de utilizador são:

- Cliente em navegação (consulta de lojas/produtos): ≈ 1–3 Mbps downstream.
- Entregador com rastreio GPS ativo (Mapbox + Firestore writes): ≈ 2–5 Mbps downstream / 0,5 Mbps upstream.
- Upload de fotografia de perfil (base64, ≈ 200 KB comprimida): pontual, ≈ 1–2 Mbps.
- Latência recomendada para rastreio em tempo real: ≤ 150 ms RTT.

A solução não requer infraestrutura de rede dedicada. A conectividade é assegurada pelas operadoras móveis e/ou redes Wi-Fi disponíveis nos locais de operação.

4.4.2.4 Serviços de Terceiros

Serviço	Fornecedor	Utilização	Modelo de Custo
Mapbox Maps SDK	Mapbox, Inc.	Mapas, rotas e navegação GPS em tempo real	Gratuito até 50 000 loads/mês
Mapbox Directions API	Mapbox, Inc.	Cálculo de rotas entregador → destino	Gratuito até 100 000 req/mês
Google Play Services Location	Google LLC	Coordenadas GPS no dispositivo Android	Incluído no SO Android
Apple Core Location	Apple Inc.	Coordenadas GPS no dispositivo iOS	Incluído no iOS
connectivity_plus	Flutter Community	Deteção de estado de rede no dispositivo	Open-source, sem custo

4.4.2.5 Resumo de Recursos em Produção

Componente	Recurso	Valor Estimado
Dispositivo móvel	CPU	≥ 1,2 GHz (ARM64 ou Apple A9+)
Dispositivo móvel	RAM	≥ 2 GB
Dispositivo móvel	Armazenamento	≈ 130–230 MB (instalação + cache)
Dispositivo móvel	Rede	≥ 5 Mbps / latência ≤ 150 ms
Cloud Firestore (backend)	Armazenamento inicial	< 1 GiB (plano gratuito)
Cloud Firestore (backend)	Escalabilidade	Automática (gerida pela GCP)
Mapbox	Pedidos mensais estimados	≤ 50 000 loads (plano gratuito)
FCM (notificações)	Capacidade	Ilimitado (sem custo)

A solução não requer a aquisição ou gestão de hardware de servidor próprio, drones, robôs ou outros artefactos físicos para além dos dispositivos móveis dos utilizadores finais. Toda a infraestrutura de backend é provisionada e escalada automaticamente pelos fornecedores de serviços cloud identificados.

4.5 Abrangência

A solução proposta integra conhecimentos adquiridos ao longo do curso de Engenharia Informática, combinando diferentes áreas científicas e unidades curriculares que contribuem diretamente para o desenvolvimento do projeto “Longe, com Tudo”.

Embora o projeto se encontre ainda numa fase inicial, já é possível identificar de forma clara quais os tópicos e competências académicas relevantes que sustentam esta solução.

- **Computação Móvel**
  Os princípios de desenvolvimento de aplicações móveis, a estruturação de interfaces simples e acessíveis, e a organização de navegação adequada a utilizadores com baixa literacia digital constituem a base funcional do projeto. As boas práticas de UI/UX,

gestão de estado e interação com serviços externos serão aplicadas no desenvolvimento do protótipo.

- **Engenharia de Software**
  A definição de requisitos, gestão do ciclo de vida do software, identificação de stakeholders, criação de user stories, casos de uso e validação através de critérios de aceitação são diretamente derivados dos conteúdos desta unidade curricular. A organização modular em epics, features e requisitos decorre igualmente dos métodos formais estudados.
- **Bases de Dados**
  A modelação da informação referente a estabelecimentos, produtos, histórico de compras, carrinho e pedidos envolve conceitos de normalização, integridade referencial e desenho de modelos de dados. A futura construção do diagrama entidade-relação e da base de dados em 3FN aplica aprendizagens nucleares desta área.
- **Interação Humano-Máquina**
  A necessidade de criar uma aplicação extremamente simples, acessível e orientada a utilizadores com limitações visuais, cognitivas ou motoras implica aplicar princípios de Interação humano-máquina e acessibilidade estudados ao longo do curso.
- **Engenharia de requisitos e testes**
  Muito semelhante a engenharia de software, esta foi a disciplina que nos introduziu ao trabalho necessário antes de começar a desenvolver o projeto em si (definir requisitos, etc.), sendo a partir desta disciplina que as bases destas componentes foram desenvolvidas.

4.6 Componentes

1. Camada de Modelos

Os modelos definem as estruturas de dados principais da aplicação, cada um com métodos para converter dados para armazenamento (`toMap`) e reconstruir objetos a partir da base de dados (`fromMap`):

- **Produto** — Armazena informações de produtos (id, nome, preço, fornecedorId, quantidade, tipo)
- **PedidoItem** — Item individual de um pedido (produtoId, nome, preço, quantidade)
- **Pedido** — Representa uma encomenda (id, data, itens, moradas, código)
- **Loja** — Informações de lojas e farmácias (id, nome, foto, morada, telefone, produtos, tipo)

O campo `tipo` em `Loja` distingue entre `loja` e `farmácia`, e em `Produto` distingue entre produtos de loja e de farmácia, eliminando a necessidade de modelos separados.

2. Camada de Serviços

Os serviços encapsulam a lógica de comunicação com o Firestore e o Firebase Authentication.

Autenticação

A autenticação é gerida por três serviços distintos, cada um responsável pelo registo e login de um tipo de utilizador. Os dados comuns são guardados na coleção `users` e os dados específicos nas respetivas coleções:

- **AuthServiceCliente** — Registo e autenticação de clientes, com dados específicos guardados em `clientes`
- **AuthServiceEntregador** — Registo e autenticação de entregadores, com dados específicos guardados em `entregadores`
- **AuthServiceFornecedor** — Registo e autenticação de fornecedores, com dados específicos (`nomeEmpresa`, `nif`, `tipoLoja`) guardados em `fornecedores`

Serviços de Dados

Os serviços realizam operações CRUD sobre as entidades principais e gerem todo o ciclo de vida dos pedidos:

- **SupplierService** — Operações CRUD de produtos (adicionar, alterar, eliminar, buscar por fornecedor)
- **LojaService** — Busca e gestão de lojas e farmácias, distinguidas pelo campo `tipo`
- **PedidoService** — Gestão de pedidos, desde a criação a partir dos itens do carrinho até à atualização de estados (criado, aceite, recusado, em preparação, pronto, expedido), incluindo geração de identificadores únicos para expedição

3. Camada de Repositórios

Os repositórios atuam como intermediários entre os serviços e a interface, gerindo o estado local da aplicação. Herdam de `ChangeNotifier` para notificar a UI de alterações e implementam estados de carregamento e mensagens de erro:

- **ProdutoRepository** — Sincronização de produtos com o Firestore
- **CarrinhoRepository** — Gestão local do carrinho em memória, limpo após checkout
- **HistoricoRepository** — Carregamento e gestão do histórico de pedidos
- **LojaRepository** — Cache de lojas e farmácias, filtradas por `tipo`

4. Camada de Base de Dados

A base de dados é gerida pelo Firebase/Firestore e organizada nas seguintes coleções:

- **users** — Dados comuns a todos os utilizadores (id, email, nome, telefone, foto, tipo)
- **clientes** — Dados específicos de clientes (userId, morada de entrega)
- **entregadores** — Dados específicos de entregadores (userId, veículo, carta de condução)
- **fornecedores** — Dados específicos de fornecedores (userId, nif, nomeEmpresa, tipoLoja)

- **produtos** — Catálogo de produtos (id, nome, preço, fornecedorId, quantidade, tipo)
- **pedidos** — Encomendas (id, clienteId, entregadorId, estado, data, morada, código)
- **pedido_items** — Itens individuais de cada pedido (id, pedidoId, produtoId, nome, preço, quantidade)

5. Camada de Interface do Utilizador (UI)

A camada de UI é responsável por apresentar os dados ao utilizador e captar as suas interações, atualizando automaticamente sempre que os dados mudam nos repositórios.

Páginas do Cliente:

- Lojas e farmácias (listagem e detalhe)
- Carrinho de compras
- Histórico de pedidos
- Requisição de medicamentos
- Perfil
- Centro de Ajuda

Páginas do Fornecedor:

- Gestão de catálogo de produtos
- Gestão de stock
- Gestão de pedidos
- Expedição

Páginas do Entregador:

- Gestão de entregas
- Navegação e mapa
- Comunicação e incidentes
- Histórico de entregas

A interface chama métodos nos repositórios e serviços, mostra estados de carregamento enquanto os dados são obtidos e apresenta mensagens de erro quando algo corre mal.

Fluxo de Dados

Exemplo 1: Adicionar produto ao carrinho

- O utilizador clica em "Adicionar ao Carrinho"
- A interface chama o método adicionar no CarrinhoRepository
- O repositório verifica se o produto já existe e atualiza a quantidade

- O repositório notifica os listeners
- A interface atualiza automaticamente

Exemplo 2: Criar pedido

- O utilizador clica em "Confirmar Compra"
- A interface chama o `PedidoService`
- O serviço guarda o pedido no Firestore
- O repositório atualiza o histórico
- O carrinho é limpo
- A interface mostra confirmação

Exemplo 3: Carregar histórico

- A página de histórico abre
- A interface chama o `HistoricoRepository`
- O repositório ativa o estado de carregamento
- O serviço consulta o Firestore
- O repositório guarda os dados e desativa o estado de carregamento
- A interface mostra os pedidos

4.7 Interfaces

A aplicação, Longe, com tudo, serve três perfis de utilizador distintos — cliente, entregador e fornecedor — cada um com o seu próprio conjunto de ecrãs e fluxos de navegação. A interface foi desenvolvida em Flutter, com widgets Material Design, numa linguagem visual coerente baseada em azul como cor primária. As secções seguintes apresentam os ecrãs mais representativos de cada perfil, com descrição funcional, detalhes de implementação e decisões de design tomadas.

4.7.1 Autenticação — Ecrãs Comuns aos Três Perfis

4.7.1.1 Menu de Seleção de Perfil

Figura 3 — Menu de seleção de perfil

O ecrã de boas-vindas é o ponto de entrada da aplicação e apresenta as três opções de perfil: Cliente, Entregador e Fornecedor. Cada opção é representada por um card com ícone ilustrativo, nome e breve descrição da função.

Implementação

Este ecrã é apresentado pela SplashPage que, após verificar o estado de autenticação Firebase, redireciona automaticamente o utilizador já autenticado para o ecrã correto sem passar por este menu. Cada card é um ListTile dentro de um Card com bordas arredondadas. A navegação para o login ou registo de cada perfil é feita via Navigator.push().

Decisões de Design

- A separação dos três perfis no ecrã de entrada evita confusão entre fluxos de autenticação distintos e permite diferenciar visualmente as rotas desde o início.
- Os ícones têm cores distintas por perfil — azul para cliente, verde para entregador, rosa para fornecedor — reforçando a identidade visual de cada papel ao longo de toda a aplicação. (Por implementar)

4.7.1.2 Login e Registo

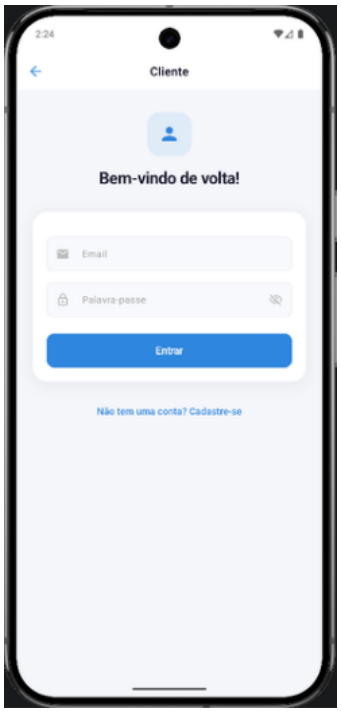

Figura 4 — Login (Cliente)

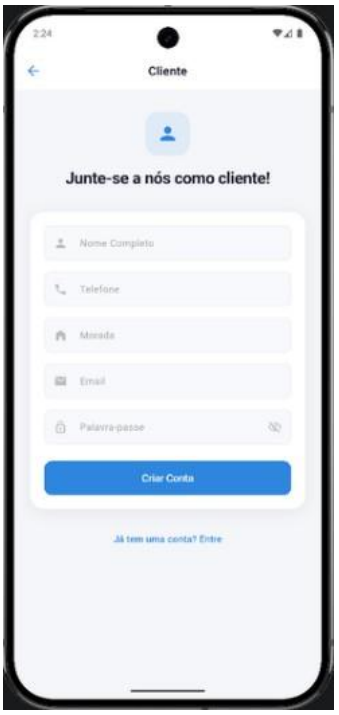

Figura 5 — Registo (Cliente)

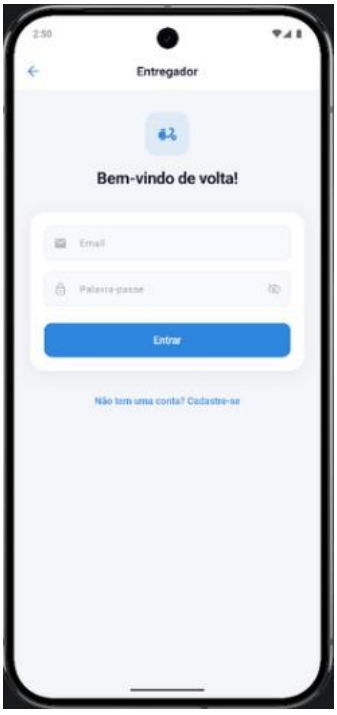

Figura 6 — Login (Entregador)

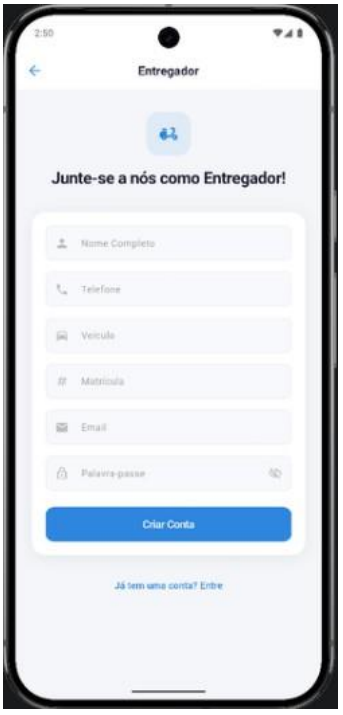

Figura 7 — Registo (Entregador)

Os ecrãs de login e registo partilham o mesmo layout base entre os três perfis: ícone do perfil no topo, título de boas-vindas, formulário com campos TextField dentro de um Card branco com sombra, botão primário de ação e link de navegação entre login e registo.

Implementação

A autenticação usa Firebase Authentication (e-mail/password). O login chama FirebaseAuth.instance.signInWithEmailAndPassword() num bloco try/catch/finally: o catch captura FirebaseAuthException e apresenta a mensagem de erro, e o finally garante que o estado de loading é sempre reposto, evitando que o botão fique bloqueado em caso de erro de rede.

O registo cria a conta com createUserWithEmailAndPassword() e de seguida cria um documento na coleção utilizadores do Firestore com os campos do formulário e o campo tipo (cliente, entregador ou fornecedor), lido pela SplashPage para determinar o redirecionamento pós-login.

A lógica de seleção de fotografia de perfil é partilhada pelos três formulários de registo através do mixin PerfilFotoMixin, que usa image_picker com compressão (imageQuality: 50, maxWidth: 300). A imagem é armazenada em base64 no Firestore, evitando o custo adicional do Firebase Storage.

Decisões de Design

- O registo do entregador inclui campos adicionais de Veículo e Matrícula, necessários para identificação operacional, que não existem nos outros perfis.
- A reutilização do layout base entre perfis minimiza a duplicação de código: apenas o ícone, cor e campos específicos variam entre perfis.
- A password é ocultada por defeito, com botão de visibilidade (visibility/visibility_off) gerido por um booleano de estado local.

4.7.2 Perfil Cliente

4.7.2.1 Dashboard do Cliente

Figura 8 — Dashboard do Cliente

O dashboard do cliente é o hub central de navegação após o login. Apresenta quatro atalhos principais em grelha: Perfil, Lojas, Farmácia e Enviar Receita Médica. Na parte inferior são apresentados dois cards contextuais: "Deseja comprar novamente?" (histórico de pedidos) e "Enviar receita médica".

Implementação

Implementado como StatelessWidget com Column principal que organiza a AppBar azul (título "Menu" e ícone de ajuda) seguida de uma GridView.count com 2 colunas para os atalhos principais. O botão de ajuda (?) no canto superior direito navega para o Centro de Ajuda.

Decisões de Design

- A opção Farmácia foi incluída para suportar encomendas de produtos farmacêuticos, diferenciando este tipo de loja e justificando o ecrã de Receita Médica.
- Os cards de ação rápida na base reduzem o número de toques para as ações mais frequentes.

4.7.2.2 Lista de Lojas e Produtos

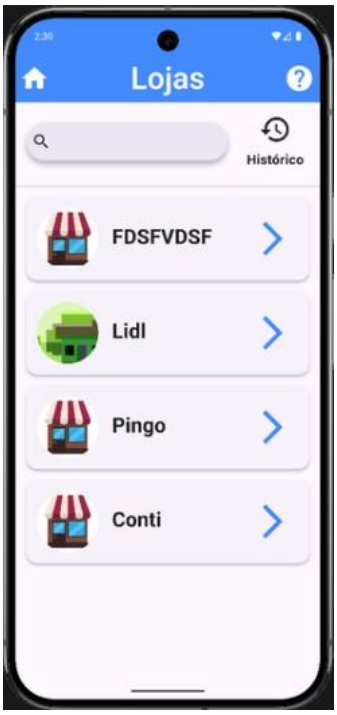

Figura 9 — Lista de Lojas

Figura 10 — Produtos da Loja (loja sem produtos)

O ecrã das Lojas apresenta todas as lojas disponíveis obtidas em tempo real do Firestore. Cada loja é um ListTile com fotografia circular, nome e seta de navegação. A barra de pesquisa no topo filtra lojas por nome. O ecrã de produtos (Figura 8) apresenta os artigos da loja selecionada com acesso rápido ao Carrinho e Histórico — neste caso vazio por a loja de teste não ter produtos adicionados.

Implementação

Ambos os ecrãs utilizam StreamBuilder sobre queries Firestore para apresentar dados em tempo real. A pesquisa é implementada com filtro local sobre a lista em memória, sem queries adicionais ao Firestore. As fotos das lojas são carregadas a partir de strings base64 com Image.memory(base64Decode(foto)).

Decisões de Design

- StreamBuilder em vez de FutureBuilder garante atualização em tempo real quando o fornecedor altera dados, sem reload manual.
- Pesquisa local em memória foi preferida à pesquisa server-side para evitar latência adicional e custos de leitura Firestore, sendo suficiente para o volume de dados esperado.

4.7.2.3 Carrinho de Compras

Figura 11 — Carrinho de Compras

O ecrã do Carrinho apresenta os itens selecionados, o total acumulado e dois botões de ação: Finalizar (submeter pedido) e Esvaziar (limpar carrinho). O total é atualizado em tempo real conforme os itens mudam.

Implementação

O estado do carrinho é gerido pelo CarrinhoRepository (ChangeNotifier) registado no Provider tree raiz. O ecrã usa Consumer<CarrinhoRepository> para reconstruir apenas a secção afetada quando o estado muda. O botão Finalizar cria um documento Pedido no Firestore e invoca carrinho.limpar(). A lista de itens é exposta como UnmodifiableListView, impedindo modificações externas ao repositório.

Decisões de Design

- A separação entre lógica do carrinho (CarrinhoRepository) e UI segue o padrão estabelecido: ChangeNotifier para estado local da app, StreamBuilder para estado Firestore.

4.7.2.4 Perfil do Cliente

Figura 12 — Perfil do Cliente

O Perfil apresenta fotografia, nome, morada, telefone e número de Cartão de Cidadão, com botões "Editar perfil" e "Log out". Os dados são carregados pelo UtilizadorProvider, que os mantém em cache após o login.

Implementação

Os dados são injetados via Provider.of<UtilizadorProvider>(context). O botão Editar navega para EditarPerfilPage e ao regressar atualiza os dados com .then((_) => _loadUserData()), garantindo que a UI reflete as alterações sem necessidade de logout. O botão Log out invoca FirebaseAuth.instance.signOut() e redireciona para o menu inicial.

Decisões de Design

- Armazenamento de fotos em base64 no Firestore (em vez do Firebase Storage) por razões de custo. A imagem é comprimida com imageQuality: 50 e maxWidth: 300 para caber dentro do limite de 1MB por documento Firestore.

4.7.2.5 Centro de Ajuda

Figura 13 — Centro de Ajuda

Figura 14 — Envio de Receita Médica

O Centro de Ajuda (Figura 11) oferece dois mecanismos de contacto: chamada telefónica (url_launcher abre a app de chamadas com número pré-definido) e envio de mensagem escrita com limite de 250 caracteres. O contador é atualizado em tempo real via TextEditingController.

O ecrã de Receita Médica (Figura 12) permite ao cliente anexar uma imagem da receita (galeria ou câmara via image_picker) com pré-visualização antes do envio. O botão "Enviar Receita" está desativado (onPressed: null) enquanto não existe imagem selecionada, fornecendo feedback visual imediato ao utilizador.

4.7.3 Perfil Entregador

4.7.3.1 Mapa e Rastreio GPS

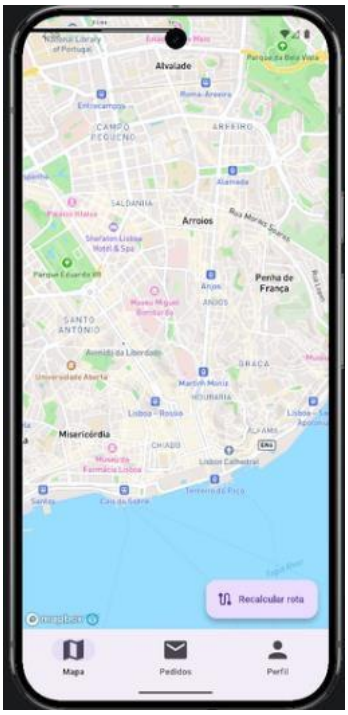

Figura 15 — Mapa do Entregador (Mapbox)

O ecrã de Mapa é o ecrã central do entregador. Apresenta um mapa interativo Mapbox centrado na localização atual, com visualização de rota até ao destino e botão "Recalcular rota". A barra de navegação inferior dá acesso a Mapa, Pedidos e Perfil.

Implementação

O mapa usa mapbox_maps_flutter. A localização é obtida em tempo real via geolocator e escrita no Firestore, permitindo ao cliente seguir a entrega. A rota é calculada pela Mapbox Directions API e desenhada como polyline via layer GeoJSON. O token Mapbox é carregado via flutter_dotenv a partir de ficheiro .env excluído do controlo de versões.

Durante o desenvolvimento foram corrigidos bugs críticos: conflitos de namespace no AndroidManifest, memory leaks nos annotation managers (resolvidos com dispose() explícito), IDs de source/layer duplicados ao navegar entre ecrãs, e URLs de ícones inválidas nos marcadores.

Decisões de Design

- Mapbox em vez de Google Maps pelo plano gratuito mais generoso (50 000 loads/mês) e maior flexibilidade de personalização do estilo do mapa.
- Token gerido via flutter_dotenv para evitar exposição no repositório, reconhecendo que a extração do token de um APK compilado permanece uma limitação de segurança em produção.

4.7.3.2 Perfil do Entregador

Figura 16 — Perfil do Entregador

O perfil do entregador mostra fotografia, nome, avaliação média (4.95 estrelas), taxa de aceitação (90%) e taxa de cancelamento (2%). Três botões de ação: Editar, Online/Offline e Logout. A secção Estatísticas apresenta o total de entregas feitas (20) e canceladas (1).

Implementação

O toggle Online/Offline atualiza o campo disponível no documento Firestore do entregador, determinando se recebe novos pedidos. As estatísticas são lidas de campos agregados no documento, atualizados sempre que um pedido muda de estado.

Decisões de Design

- A exibição de métricas de desempenho no perfil serve como feedback imediato e motivação para o entregador.
- O toggle Online/Offline tem posição de destaque nos botões de ação por ser uma funcionalidade crítica para a operação.

Nota: O ecrã de Pedidos do entregador (listagem de pedidos disponíveis e atribuídos) está previsto na arquitetura, mas não se encontra implementado na versão atual da aplicação.

4.7.4 Perfil Fornecedor

4.7.4.1 Dashboard do Fornecedor

Figura 17— Dashboard do Fornecedor

O dashboard do fornecedor apresenta um resumo de desempenho da loja: Produtos Vendidos (245), Receita Total (€2.450), Encomendas (38) e Avaliação (4.8), em grelha de 4 cards com ícones coloridos. A secção "Atividades Recentes" lista os eventos mais recentes (nova encomenda, avaliação recebida). A barra de navegação inferior dá acesso a Home, Menu e Perfil.

Implementação

As estatísticas são lidas do documento do fornecedor no Firestore com FutureBuilder (leitura única). As atividades recentes usam StreamBuilder sobre uma subcoleção atividades, ordenada por timestamp decrescente com limit(10).

Decisões de Design

- FutureBuilder para estatísticas e StreamBuilder para atividades recentes otimiza leituras Firestore: as estatísticas não requerem atualização em tempo real, enquanto as atividades beneficiam de notificação imediata.

4.7.4.2 Gestão de Produtos

Figura 18 — Gestão de Produtos (sem produtos adicionados)

O ecrã de Gestão de Produtos permite ao fornecedor adicionar e remover produtos da sua loja. O ID do fornecedor é apresentado em destaque. Os botões "+ Adicionar Produto" e "Remover" estão sempre visíveis. A lista mostra "Nenhum produto adicionado" por a loja de teste ainda não ter produtos registados; a funcionalidade de adição está implementada e não funcional, ainda falta adicionar leitura de documentos Excel, assim o fornecedor consegue adicionar vários produtos de uma só vez.

Implementação

Os produtos são armazenados numa subcolecção produtos dentro do documento da loja no Firestore (lojas/{lojaId}/produtos/{produtoId}). A lista usa StreamBuilder para atualizar em tempo real. O botão "+ Adicionar Produto" abre um AlertDialog com campos para nome, preço e quantidade. O botão "Remover" ativa o modo de seleção múltipla para eliminação.

Decisões de Design

- Subcoleções Firestore em vez de arrays no documento principal evitam o limite de 1MB por documento e permitem queries independentes sobre produtos.
- O ID do fornecedor é apresentado no ecrã para facilitar o debugging e a identificação da loja em contexto de desenvolvimento.

5 Testes e Validação

5.1 Abordagem e Justificação

A estratégia de testes adotada para o projeto Longe, com tudo segue a pirâmide de testes clássica, combinando três níveis de cobertura — testes unitários, testes de integração e testes de widget — com o objetivo de garantir a qualidade da solução a diferentes granularidades, desde a lógica de negócio isolada até ao comportamento dos componentes de interface de utilizador.

A escolha desta abordagem justifica-se pela natureza da solução: uma aplicação móvel Flutter com backend Firebase, em que a lógica de negócio (modelos de dados, cálculo de totais, gestão do carrinho) é completamente desacoplada dos serviços externos, permitindo a sua validação autónoma e determinística. Os testes não dependem de ligação à internet, de contas Firebase ou de emuladores de serviços cloud, o que os torna reprodutíveis em qualquer ambiente de desenvolvimento.

A ausência de testes automáticos sobre os serviços Firebase (Authentication, Firestore) e sobre a integração com o Mapbox é intencional e justificada: estes serviços requerem mocks ou emuladores dedicados que introduziriam complexidade de configuração desproporcional ao âmbito do projeto. A validação destes componentes é realizada manualmente, conforme descrito na secção 5.4.

5.1.1 Modelo de Análise de Risco

Para priorizar os testes a implementar, foi realizada uma análise de risco baseada na probabilidade de falha e no impacto no utilizador final:

Tabela 9 - Análise de Risco

Componente	Probabilidade de Falha	Impacto	Prioridade	Cobertura
Cálculo de totais do carrinho	Média	Alto (financeiro)	Crítica	Automatizada (unitário)
Modelos de dados (Pedido, Produto)	Baixa	Alto (integridade de dados)	Alta	Automatizada (unitário)
Fluxo completo de compra	Média	Alto (experiência de utilizador)	Alta	Automatizada (integração)
Widgets do carrinho	Baixa	Médio (UI)	Média	Automatizada (widget)

Autenticação Firebase	Baixa	Alto (acesso)	Alta	Manual
Rastreio GPS / Mapbox	Média	Alto (funcionalidade central)	Alta	Manual
Conectividade / modo offline	Média	Médio (resiliência)	Média	Manual
Upload de fotografia de perfil	Baixa	Baixo	Baixa	Manual

5.2 Conjunto de Testes Automatizados

O conjunto de testes automatizados é composto por 44 testes distribuídos por três categorias, organizados na pasta test/ do projeto Flutter. Todos os testes seguem o padrão Arrange-Act-Assert, utilizam setUp/tearDown para garantir isolamento entre testes, e são independentes entre si — podem ser executados em qualquer ordem e em qualquer ambiente com Flutter SDK instalado.

Comando de execução: flutter test

5.2.1 Testes Unitários — Modelos de Dados (test/models/)

Os testes unitários dos modelos validam a criação, serialização e lógica de negócio dos quatro modelos principais da aplicação. Total: 22 testes.

Ficheiro	Modelo	Casos de Teste	Validações Principais
loja_model_test.dart	Loja	3	Criação com todos os campos; lista de produtos vazia por defeito; inicialização com produtos
produto_model_test.dart	Produto	6	Criação com parâmetros; ID único gerado automaticamente; serialização toMap(); preço

			decimal; IDs distintos entre instâncias
pedido_item_model_test.dart	PedidoItem	6	Criação com parâmetros; cálculo de subtotal (preço × quantidade); serialização toMap()/fromMap() reversível
pedido_model_test.dart	Pedido	5	Criação completa; cálculo de total agregado; cálculo de totalItens; serialização toMap() com userId; consistência com múltiplos itens

Destaque para os testes de cálculo financeiro: o modelo PedidoItem valida que subtotal = preço × quantidade, e o modelo Pedido valida que total = soma de todos os subtotais dos itens. Estes testes garantem que nenhuma alteração futura ao código introduza erros de arredondamento ou de lógica nos valores apresentados ao utilizador.

5.2.2 Testes Unitários — Repositório do Carrinho (test/repositories/)

O CarrinhoRepository é a peça central da lógica de negócio do lado do cliente, implementando o padrão ChangeNotifier para notificar a UI de alterações ao estado do carrinho. Total: 11 testes.

Tabela 10 - Testes Unitários

Caso de Teste	Descrição	Resultado Esperado
Carrinho inicia vazio	Instanciar CarrinhoRepository sem operações	itens vazio; total == 0.0
Adicionar um produto	adicionar(produto, quantidade: 2)	1 item; total correto
Adicionar múltiplos produtos	Adicionar dois produtos distintos	2 itens; total = soma dos subtotais
Adicionar mesmo produto acumula	adicionar() duas vezes com mesmo produto	1 item; quantidade = soma; total atualizado

Cálculo total correto	Produtos com preços decimais e quantidades variadas	Total = (1.50×2) + (5.0×1) = 8.0
Remover item do carrinho	remover(item) após adicionar	Carrinho vazio; total == 0.0
Remover mantém outros itens	remover() apenas um de dois itens	1 item restante; total atualizado
Limpar carrinho	limpar() com múltiplos itens	Carrinho vazio; total == 0.0
Lista itens é imutável	Tentar adicionar à lista retornada por itens	Lança UnsupportedError
Notifica ao adicionar	addListener + adicionar()	Listener chamado 1 vez por operação
Notifica ao remover/limpar	addListener + remover() ou limpar()	Listener chamado 1 vez

O teste de imutabilidade da lista de itens é particularmente relevante para a arquitetura da aplicação: garante que componentes UI não conseguem alterar o estado do carrinho diretamente, sendo obrigados a passar pelas operações públicas do repositório, mantendo a consistência do estado.

5.2.3 Testes de Integração — Fluxo de Compra (test/integration/)

Os testes de integração validam fluxos de negócio completos, combinando múltiplos modelos e o repositório do carrinho. Total: 5 testes.

Tabela 11 - Testes de Integração

Cenário	Passos do Fluxo	Validações
Fluxo completo de compra	Criar loja → adicionar produtos → adicionar ao carrinho → criar Pedido → limpar carrinho	Total do pedido = 6.50€; totalItens = 4; moradas corretas; carrinho limpo após pedido
Múltiplas lojas com carrinho separado	Compra em loja 1 → limpar → compra em loja 2	Pedidos com moradas distintas; totais

		independentes; carrinho reutilizável
Adicionar, remover e finalizar	Adicionar 2 produtos → remover 1 → criar pedido	Total reflete apenas item mantido; totalItens correto
Vários itens da mesma loja	4 produtos distintos de 1 loja → criar pedido	Total = 10.50€; totalItens = 6; morada da loja correta
Consistência em operações múltiplas	Adicionar → adicionar mesmo produto → remover → adicionar → limpar	Estado do carrinho correto após cada operação

5.2.4 Testes de Widget (test/widgets/)

Os testes de widget validam o comportamento dos componentes de interface de utilizador, verificando renderização correta e resposta a interações. Total: 6 testes.

Tabela 12 - Testes de Widgets

Ficheiro	Widget	Casos de Teste
carrinho_item_widget_test.dart	CarrinhoItemWidget	Exibição de nome, preço e quantidade; botão de remover funcional (callback disparado); renderização de múltiplos itens em ListView
simple_widgets_test.dart	TotalPriceWidget, BotaoAcao	Formatação de preço com 2 casas decimais (€10.50, €5.00, €0.00); exibição de texto e ícone no botão; botão clicável via Key; renderização de cor

5.2.5 Resumo do Conjunto de Testes Automatizados

Tabela 13 - Resumo dos testes

Categoria	Localização	Nº Testes	Dependências Externas

Testes Unitários — Modelos	test/models/	22	Nenhuma
Testes Unitários — Repositório	test/repositories/	11	Nenhuma
Testes de Integração	test/integration/	5	Nenhuma
Testes de Widget	test/widgets/	6	Flutter test framework
TOTAL		44	

5.3 Critérios de Aceitação

Os critérios de aceitação do conjunto de testes automatizados são os seguintes:

- Todos os 44 testes devem passar sem falhas (flutter test com resultado 44 passed, 0 failed).
- Nenhum teste deve depender de estado externo (Firebase, rede, sistema de ficheiros).
- O tempo total de execução do conjunto deve ser inferior a 60 segundos em qualquer máquina com Flutter SDK instalado.
- A lista de itens do CarrinhoRepository deve ser imutável externamente (UnsupportedError em tentativas de modificação direta).
- Os cálculos de subtotal, total e totalItens devem ser exatos para os valores de teste definidos (sem erros de arredondamento em virgula flutuante).

5.4 Testes Manuais — Validação Operacional

Para além do conjunto de testes automatizados, foram realizados testes manuais sobre os componentes que dependem de serviços externos (Firebase, Mapbox, GPS). Estes testes foram executados no emulador Android Pixel 9 Pro XL (API 36) descrito na secção 4.4.1, ligado ao projeto Firebase de desenvolvimento.

5.4.1 Autenticação e Gestão de Utilizadores

Tabela 14 - Testes de Autenticação

Caso de Teste	Procedimento	Resultado Esperado

Registo de novo utilizador (cliente)	Preencher formulário com e-mail e password válidos; selecionar perfil cliente	Conta criada no Firebase Auth; documento criado no Firestore (coleção utilizadores); redireciona para home do cliente
Registo de fornecedor	Selecionar perfil fornecedor no registo	Documento Firestore com tipo=fornecedor; acesso ao painel de gestão de loja
Registo de entregador	Selecionar perfil entregador	Documento Firestore com tipo=entregador; acesso ao painel de entregas
Login com credenciais válidas	Introduzir e-mail e password corretos	Autenticação bem-sucedida; UtilizadorProvider carregado; navegação para home correta por perfil
Login com credenciais inválidas	Introduzir password errada	Mensagem de erro apresentada; sem navegação
Edição de perfil	Alterar nome e foto de perfil	Dados atualizados no Firestore; UI reflete alteração após Navigator.push().then()
Persistência de sessão	Fechar e reabrir a app	Utilizador continua autenticado; dados do UtilizadorProvider recarregados

5.4.2 Gestão de Lojas e Produtos (Perfil Fornecedor)

Tabela 15 - Testes de gestão de lojas

Caso de Teste	Procedimento	Resultado Esperado
Criar loja	Preencher nome, morada, telefone e foto	Documento criado na subcoleção lojas do fornecedor no Firestore

Adicionar produto	Introduzir nome, preço e quantidade	Produto adicionado à subcoleção produtos da loja; visível via StreamBuilder
Editar produto	Alterar preço de produto existente	Documento Firestore atualizado; UI reflete alteração em tempo real
Eliminar produto	Confirmar eliminação de produto	Documento removido do Firestore; produto desaparece da lista
Visualização em tempo real	Editar produto com app aberta	StreamBuilder atualiza a lista sem reload manual

5.4.3 Rastreio GPS e Mapas (Mapbox)

Tabela 16 - Testes do Mapa

Caso de Teste	Procedimento	Resultado Esperado
Visualização do mapa	Abrir ecrã de mapa com token Mapbox válido (.env)	Mapa renderiza corretamente; sem erros de token
Localização GPS do entregador	Ativar GPS no emulador; abrir rastreio	Coordenadas obtidas via GPS; marcador do entregador aparece no mapa
Cálculo de rota	Definir origem (entregador) e destino (cliente)	Rota calculada via Mapbox Directions API; trajeto visível no mapa
Atualização em tempo real	Mover coordenadas GPS do emulador	Marcador do entregador atualiza posição; Firestore regista coordenadas
Gestão de memória	Navegar entre ecrãs com mapa múltiplas vezes	Sem memory leaks; annotation managers não duplicados

5.4.4 Verificação de Recursos (Ambiente de Teste)

Em linha com os requisitos da secção 4.4, foram verificados os seguintes recursos durante os testes operacionais no emulador:

Tabela 17 - Ambiente de Teste

Recurso	Verificação Realizada	Resultado
RAM (emulador: 2 GB)	Execução de fluxos completos sem crash por falta de memória	OK — heap VM de 256 MB não excedido em operação normal
Armazenamento (emulador: 12 GB)	Instalação da app + dados de teste Firestore	OK — ocupação < 200 MB
GPS (emulador)	Coordenadas simuladas via Android Studio → Extended Controls → Location	OK — localização recebida pelo flutter_geolocator
Rede (emulador: full speed, 0ms latência)	Sincronização Firestore e pedidos Mapbox	OK — latência < 100 ms em operação local
Firebase Auth & Firestore	Operações CRUD com contas de teste	OK — todas as operações concluídas com sucesso
Mapbox SDK	Carregamento de tiles e cálculo de rota	OK — dentro do limite gratuito (< 50 000 loads/mês)
ConnectivityProvider	Toggle de rede no emulador	OK — banner offline/online com latência < 1 s

5.5 Limitações e Trabalho Futuro

O conjunto de testes atual cobre a lógica de negócio local de forma abrangente, mas apresenta as seguintes limitações identificadas:

- Firebase não mockado: os testes automatizados não cobrem operações de leitura/escrita no Firestore nem a Firebase Authentication. A implementação futura de mocks com firebase_core_mocks e fake_cloud_firestore permitiria cobrir estes cenários de forma automática.
- Sem testes E2E: a ausência de testes end-to-end (usando o package integration_test da Flutter) significa que os fluxos de navegação completos entre ecrãs só são validados manualmente.

- Sem testes de performance: não foram implementados benchmarks para operações críticas como o carregamento de listas de produtos ou a latência do rastreio GPS.
- Participação de terceiros: a validação operacional foi realizada apenas pelo desenvolvedor. A participação de utilizadores reais dos três perfis (cliente, entregador, fornecedor) em testes de aceitação de utilizador (UAT) está prevista como passo seguinte.

6 Método e Planeamento

6.1 Planeamento inicial

O planeamento do projeto foi definido tendo como referência o calendário sugerido pelo professor orientador, que apresenta uma divisão faseada do trabalho ao longo do ano letivo.

Este planeamento serve como guia estruturado para organizar o desenvolvimento da solução e garantir que todas as etapas são realizadas de forma progressiva e coerente com os objetivos do TFC.

Tabela 18 - Tabela de Planeamento

Fase	Período indicativo	Objetivos principais
1.Enquadramento e Investigação	Set. – Out. 2025	-Definir problema e objetivos -Levantar estado da arte -Estruturar metodologia e planeamento
2.Desenho e Modelação da Solução	Out. – Nov. 2025	-Especificar requisitos, arquitetura -Criar protótipo inicial
3. Desenvolvimento e Testes	Dez. 2025 – Mar. 2026	-Desenvolver e integrar componentes - Validar requisitos e metodologias
4.Validação, Conclusão e Apresentação	Abr. – Jun. 2026	- Testar e validar resultados - Redigir relatório final e preparar defesa

Este planeamento inicial constitui uma orientação geral para o desenvolvimento do projeto, sendo suficientemente flexível para permitir ajustes futuros.

6.2 Análise Crítica ao Planeamento

À data de entrega deste relatório (12 de abril de 2026), o projeto encontra-se na fase 3, com a fase de especificação concluída e o desenvolvimento em curso.

As tarefas realizadas até ao momento incluem:

- Levantamento e definição completa de requisitos funcionais e não funcionais, macro e micro, para os três perfis de utilizador
- Definição da arquitetura da solução e das tecnologias utilizadas
- Estruturação da base de dados no Firestore, incluindo a definição das coleções e respetivos campos
- Implementação parcial dos módulos do cliente e do fornecedor

As tarefas ainda em curso ou por concluir são:

- Finalização da implementação da estrutura da base de dados e validação dos campos
- Implementação do módulo do entregador (receção de pedidos, atualização de estados, histórico de entregas)
- Testes de integração entre os três perfis de utilizador

Desvios ao planeamento

Regista-se um atraso de aproximadamente um mês em relação ao planeamento inicial, situando o fim efetivo da fase de desenvolvimento em abril de 2026 em vez de março de 2026. Este atraso deveu-se essencialmente a dois fatores:

- **Identificação tardia dos três perfis de utilizador** — na primeira entrega, a solução foi conceptualizada apenas com o perfil de cliente. A necessidade de incluir os perfis de fornecedor e entregador só foi claramente identificada numa fase posterior, o que implicou rever e expandir significativamente toda a especificação de requisitos, casos de uso e modelação de dados.
- **Dificuldade na definição da estrutura da base de dados** — chegar a um consenso sobre a organização das coleções no Firestore, nomeadamente a separação entre dados comuns e dados específicos por tipo de utilizador, revelou-se um desafio que consumiu tempo considerável da fase de especificação.

Apesar do atraso registado, considera-se que o projeto se encontra em condições de ser concluído dentro do prazo de entrega final em junho de 2026, dado que a especificação está consolidada e o desenvolvimento dos módulos em falta é tecnicamente viável no tempo disponível.

Questionário de AI

DEISI_TFC_FormUsoIA .docx

Bibliografia

[DEISI24] DEISI, Regulamento de Trabalho Final de Curso, Out. 2024.

[DEISI24b] DEISI, www.deisi.ulusofona.pt, Out. 2024.

[TaWe20] Tanenbaum,A. e Wetherall,D., *Computer Networks*, 6ª Edição, Prentice Hall, 2020.

[ULHT21] Universidade Lusófona de Humanidades e Tecnologia, www.ulusofona.pt, acedido em Out. 2024.

Glossário

LEI	Licenciatura em Engenharia Informática
LIG	Licenciatura em Informática de Gestão
TFC	Trabalho Final de Curso