

## Relatório Reflexivo

**Projeto:** Leafy (App de Cuidados com Plantas)

**Aluno:** Guilherme da Silva

**Assunto:** Relatório Reflexivo da Fase 2 (Avatar) - Implementação Guiada por IA

---

### 1. Introdução e Planejamento

**Objetivo:** O objetivo desta fase era implementar a funcionalidade de "Avatar do Utilizador" conforme o PRD da Fase 2. Isto envolvia não só a seleção de imagem, mas também requisitos complexos de privacidade (LGPD), acessibilidade (A11Y) e a criação de testes unitários e de widget.

**Contexto:** Esta fase foi iniciada após a correção da Fase 1 (Onboarding/LGPD), que tinha recebido uma nota de 41/100. O repositório Git foi estruturado corretamente (com *branches* main/dev) antes de começar a Fase 2.

**Estratégia de IA:** Adotei uma metodologia de "Pair Programming" (Programação em Par) com a IA (Gemini).

- **Eu (Aluno) atuei como "Piloto":** Defini os requisitos através de *prompts* detalhados, validei o código gerado, testei a funcionalidade no emulador e reportei todos os erros, *warnings* ou dificuldades (ex: enviando *logs* de erro ou *screenshots*).
- **A IA (Gemini) atuou como "Navegador":** Gerou o código-base, sugeriu a arquitetura, explicou conceitos complexos (como LGPD e processamento de imagem) e depurou os erros que eu reportava.

A IA ajudou a planear a implementação dividindo-a em 5 Pull Requests (PRs) lógicos:

1. **PR 1:** Tornar o Avatar Clicável (UI + Acessibilidade)
  2. **PR 2:** Seleção de Imagem e Gestão de Permissões
  3. **PR 3:** Processamento (LGPD/Compressão) e Persistência de Dados
  4. **PR 4:** Exibição da Imagem e Lógica de *Fallback*
  5. **PR 5:** Escrita de Testes Unitários e de Widget
- 

### 2. Execução Guiada por IA (PR 1 ao PR 5)

**PR 1: Avatar Clicável (UI/A11Y)**

- **Prompt (Aluno):** Solicitei à IA que tornasse o CircleAvatar (com as iniciais "GU") clicável, garantindo uma área de toque mínima de 48dp (A11Y) e adicionando Semantics (descrição para leitores de ecrã).
- **Resposta (IA):** A IA sugeriu corretamente que a HomeScreen deveria ser convertida de StatelessWidget para StatefulWidget, para permitir futuras mudanças de estado (como a mudança da imagem). A IA gerou o código usando InkWell para o clique e Semantics para a acessibilidade.
- **Dificuldade/Correção:** A primeira geração de código resultou em múltiplos erros. A IA diagnosticou que o problema foi uma falha na sua instrução: ela pediu para *modificar* o ficheiro, mas a mudança de StatelessWidget para StatefulWidget exigia uma *substituição completa* do ficheiro. Ao solicitar à IA "me dê o código inteiro corrigido", ela forneceu o ficheiro home\_screen.dart completo e funcional, resolvendo o problema.

## PR 2: Seleção de Imagem e Permissões

- **Prompt (Aluno):** Pedi à IA para adicionar as dependências image\_picker e permission\_handler, gerar um showModalBottomSheet (com opções de Câmera/Galeria) e implementar a lógica de pedido de permissão.
- **Resposta (IA):** A IA gerou a função \_onAvatarTapped (para mostrar o *sheet*) e a função \_pickImage (para lidar com Permission.camera e Permission.photos).
- **Dificuldade/Correção:** Ocorreu uma dificuldade crítica. A Câmera funcionou, mas a Galeria falhava (não pedia permissão). A IA inicialmente diagnosticou incorretamente que as permissões do AndroidManifest.xml estavam corretas.
- **Resolução:** O problema só foi resolvido quando eu, o aluno, enviei um *screenshot* (imagem) das definições de permissão do app no emulador. A imagem provou que a permissão da Galeria nem sequer estava a ser listada. Com esta nova informação visual, a IA identificou o seu próprio erro e forneceu a correção exata: adicionar as permissões READ\_MEDIA\_IMAGES e READ\_EXTERNAL\_STORAGE ao AndroidManifest.xml. Após esta correção (e a desinstalação/reinstalação do app), o fluxo da galeria funcionou perfeitamente.

## PR 3: Processamento (LGPD) e Persistência

- **Prompt (Aluno):** Solicitei o núcleo da funcionalidade de privacidade: adicionar image e path\_provider, e atualizar o PrefService e a \_pickImage para:

1. Remover metadados EXIF/GPS (requisito LGPD).
  2. Comprimir a imagem (qualidade 80%).
  3. Salvar o ficheiro processado localmente.
  4. Salvar o *caminho* desse ficheiro no PrefsService.
- **Resposta (IA):** A IA gerou todo o código. Ela explicou que o processo de decodificar (`img.decodeImage`) e re-codificar (`img.encodeJpg`) a imagem naturalmente descarta os metadados EXIF/GPS. Além disso, a IA usou o `path_provider` (`getApplicationDocumentsDirectory`) para salvar a imagem numa pasta interna e privada do app, em vez da galeria pública.
  - **Reflexão:** A IA foi crucial aqui ao explicar *porquê* a foto *não deveria* aparecer na galeria pública do telemóvel, validando que este comportamento cumpria o requisito de privacidade (LGPD).

#### PR 4: Exibição da Imagem e Lógica de *Fallback*

- **Prompt (Aluno):** Pedi à IA para finalmente exibir a imagem na UI. Isto exigia (1) carregar o caminho salvo quando a tela iniciasse (`initState`), (2) atualizar a UI imediatamente após a seleção (`setState`), e (3) modificar o `CircleAvatar` para exibir a `FileImage` (se existisse) ou o *fallback* "GU" (se fosse nulo).
- **Resposta (IA):** A IA integrou a variável de estado `_avatarFilePath`, criou a função `_loadAvatar` (chamada no `initState`) e atualizou o `_pickImage` com `setState`. O `CircleAvatar` foi modificado com um `backgroundImage` condicional.
- **Reflexão:** O teste de "Revogação" (limpar o avatar ao revogar o consentimento) funcionou na primeira tentativa. Isto demonstrou a capacidade de planeamento da IA, pois no PR 3, ela já tinha "antecipado" esta necessidade e modificou a função `revokePolicy` no `PrefsService` para limpar o caminho do avatar (`_kAvatarPath`).

#### PR 5: Escrita de Testes Unitários e de Widget

- **Prompt (Aluno):** Pedi os testes unitários e de widget, sugerindo a biblioteca `mockito`.
- **Resposta (IA):** A IA gerou um Teste Unitário funcional (`prefs_service_test.dart`) que passou de primeira.
- **Dificuldade/Correção:** Esta foi a etapa com maior atrito. A IA falhou **múltiplas vezes** em gerar um Teste de Widget (`widget_test.dart`) funcional.

1. **Falha 1 (Mockito v4):** A IA gerou código para uma versão antiga do mockito, causando erros de *null-safety* (type 'Null' is not a subtype...).
  2. **Falha 2 (getIt.reset):** A tentativa de correção da IA usou getIt.reset(), o que destruiu a instância de injeção de dependência e causou um novo erro (Object/factory... is not registered).
- **Resolução:** Após várias tentativas de depuração falhadas, a IA propôs abandonar o mockito (que estava a ser problemático) e substituí-lo por uma abordagem mais simples: uma classe "Fake" (FakePrefsService). Esta abordagem alternativa, onde criámos um serviço falso em vez de um "mock", funcionou perfeitamente e resolveu todos os erros, levando ao resultado All 3 tests passed!.

---

### 3. Conclusão

A parceria Aluno-IA foi fundamental para o sucesso do projeto. A IA demonstrou uma capacidade excecional para gerar código complexo (processamento de imagem, lógica de permissões, arquitetura de serviços) a uma velocidade que seria impossível para mim sozinho, especialmente num prazo tão curto.

A maior fraqueza da IA revelou-se na configuração de testes de *widget* com dependências (getIt e mockito), onde o seu conhecimento parecia estar desatualizado ou incorreto, exigindo um ciclo intenso de depuração (Aluno reporta erro -> IA tenta correção).

No entanto, a capacidade da IA de diagnosticar os seus próprios erros (especialmente com a ajuda de *screenshots* ou *logs* de erro) e de propor soluções alternativas (como trocar mockito por um "Fake") provou ser a sua maior força. O projeto cumpriu 100% dos requisitos do PRD, incluindo os complexos requisitos de LGPD e Acessibilidade.

### 4. Apêndice: Histórico de Prompts (Fase 2)

Histórico de interação entre o Aluno (Eu) e a IA (Gemini) para a implementação da Fase 2.

---

#### PR 1: Avatar Clicável (UI/A11Y)

[Aluno] Prompt 1: "Ok, estou na branch feature/fase2-avator. Vamos começar o PR 1 (Tornar Avatar Clicável). Por favor, gere para mim o código para:

1. Modificar o `UserAccountsDrawerHeader` na `home_screen.dart` para que o `CircleAvatar` (com as iniciais 'GU') se torne clicável.
2. Garantir que a área de toque seja de no mínimo 48x48dp (requisito de Acessibilidade), usando `InkWell` ou `GestureDetector`.
3. Adicionar `Semantics` ao widget clicável, com a *label* 'Avatar do usuário' e a *hint* 'Toque para alterar sua foto de perfil'.
4. Ao ser tocado, o widget deve chamar uma função `_onAvatarTapped()` (por enquanto, deixe a função vazia, apenas com um `print('Avatar tocado');`);).

**[IA] Resposta:** A IA gerou o código e explicou que, para isso, a `HomeScreen` precisaria ser convertida de `StatelessWidget` para `StatefulWidget`, para suportar a nova função `_onAvatarTapped`.

**[Aluno] Dificuldade:** "asim que eu coleí o código deu muitos erros"

**[IA] Correção:** A IA diagnosticou o erro: "Eu cometi um erro na minha instrução... A correção é simples: você precisa **apagar 100%** do conteúdo do arquivo `home_screen.dart` e **colar** o código novo e completo que eu gerei abaixo." (IA gerou o ficheiro `home_screen.dart` completo e corrigido como `StatefulWidget`).

**[Aluno] Validação:** "ok agora está rodando normal"

---

## PR 2: Seleção de Imagem e Permissões

**[Aluno] Prompt 2:** "Ok, **PR 1 (Tornar Avatar Clicável)** está validado. O clique e as `Semantics` funcionam. Agora, vamos para o **PR 2 (Seleção de Imagem e Permissões)**. Por favor, gere o código para:

1. Adicionar as dependências `image_picker` e `permission_handler`...
2. Atualizar a função `_onAvatarTapped()` para mostrar um `showModalBottomSheet` com 'Tirar Foto' e 'Escolher da Galeria'.
3. Criar uma nova função `_pickImage(ImageSource source)` que use o `permission_handler` para pedir a permissão...
4. Por enquanto, apenas imprima o caminho (path) do arquivo selecionado."

**[IA] Resposta:** A IA gerou o comando `flutter pub add ...` e as instruções para modificar os ficheiros nativos `AndroidManifest.xml` (para Câmera) e `Info.plist` (para Câmera e Galeria). Em seguida, gerou o ficheiro `home_screen.dart` atualizado com o `showModalBottomSheet` e a lógica de permissões.

**[Aluno] Dificuldade:** "ok estou conseguindo clicar na icone de usuario e estou conseguindo tirar a foto mais nao tenho permição para abrir a galeria nem mudar o icone" (Aluno enviou a imagem *Captura de tela 2025-10-22 175813.png*)

**[IA] Correção:** A IA analisou a imagem e diagnosticou o seu próprio erro: "A sua imagem prova que... a permissão 'Fotos e vídeos' nem sequer aparece na lista... **Isto é um erro meu.** Eu disse-lhe na instrução anterior que o Android moderno não precisava de adicionar a permissão da galeria no AndroidManifest.xml. **Eu estava errado.**" (IA gerou as linhas xml corretas: `<uses-permission android:name="android.permission.READ_MEDIA_IMAGES" />` e `<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />` e instruiu a desinstalar/reinstalar o app.)

**[Aluno] Validação:** "ok estou conseguindo utilizar a camera e abrir a galeria vamos continuar"

---

### PR 3: Processamento (LGPD) e Persistência

**[Aluno] Prompt 3:** "Ok, PR 2 (Seleção de Imagem) está validado. Agora, vamos para o PR 3 (Processamento, LGPD e Persistência). Por favor, gere o código para:

1. Adicionar as dependências image e path\_provider...
2. Atualizar o PrefsService... para salvar e ler o caminho do avatar (setAvatarPath e getAvatarPath).
3. Atualizar a implementação PrefsServiceImpl...
4. Atualizar a função \_pickImage... para: a. Usar path\_provider... b. Usar a biblioteca image para... remover os metadados EXIF/GPS (LGPD), e comprimir para JPG com 80% de qualidade. c. Salvar este novo arquivo... d. Usar o PrefsService... para salvar o *caminho*... e. Mostrar um SnackBar 'Avatar atualizado com sucesso'."

**[IA] Resposta:** A IA gerou o comando flutter pub add image path\_provider path. Gerou o ficheiro prefs\_service.dart atualizado (com set/getAvatarPath e a atualização na revokePolicy). Gerou o ficheiro home\_screen.dart atualizado com a lógica completa de processamento (img.decodeImage/img.encodeJpg), salvamento (path\_provider) e feedback ao utilizador (showLoadingDialog, showSuccessSnackBar).

**[Aluno] Validação:** "Ok, PR 3 (Processamento e Persistência) está validado."

---

### PR 4: Exibição da Imagem e Lógica de *Fallback*

**[Aluno] Prompt 4:** "Ok, **PR 3** está validado. Agora, vamos para o **PR 4 (Exibição e Fallback)**. Por favor, gere o código para:

1. ...adicionar uma variável de estado `_avatarFilePath`.
2. Adicionar um `initState()` que use o `PrefsService.getAvatarPath()` para carregar o caminho salvo...
3. Modificar o `CircleAvatar`... a. Se `_avatarFilePath` **não** for nulo, deve exibir a imagem... (`FileImage`). b. Se `_avatarFilePath` **for** nulo, deve exibir as iniciais "GU" (o *fallback*).
4. Atualizar a função `_pickImage` para... usar `setState`... e a UI mudar *imediatamente*."

**[IA] Resposta:** A IA gerou o ficheiro `home_screen.dart` final, adicionando a variável de estado `_avatarFilePath`, a função `_loadAvatar()` chamada no `initState()`, a linha `setState(() { _avatarFilePath = newPath; })` dentro da `_pickImage`, e a lógica condicional no `CircleAvatar` (`backgroundImage: _avatarFilePath != null ? ...`).

**[Aluno] Dificuldade:** "ok avatar atualizado utilizando a camera mais eu nao tenho nem uma foto no album e mesmo utilizando a opicao do ap de foto ele nao salva no album"

**[IA] Correção/Explicação:** A IA explicou que isto não era um erro, mas sim o **comportamento correto de LGPD**: "O que o nosso código faz (no PR 3):... Salva a foto numa **pasta privada, interna da app** (`getApplicationDocumentsDirectory()`). **Só o seu app 'Leafy' pode ver essa foto**.... Você cumpriu o requisito de privacidade perfeitamente." A IA também instruiu como adicionar fotos à galeria do emulador (usando o Chrome) para fins de teste.

**[Aluno] Validação:** "Ok, **PR 4 (Exibição e Fallback)** está validado. A imagem aparece (da câmara e galeria), persiste após reiniciar e é limpa na revogação. A funcionalidade está completa."

---

## **PR 5: Testes (Unitário e Widget)**

**[Aluno] Prompt 5:** "Ok, **PR 4** está validado. Agora, vamos para o **PR 5 (Testes Unitários e de Widget)**... Por favor, gere o código para:

1. Adicionar a dependência `mockito`.
2. Criar um `MockPrefsService`.
3. Gerar um **Teste Unitário**...

4. Gerar um **Teste de Widget** (widget\_test.dart)... que: a. Verifique se o Text('GU') (fallback) é exibido... b. Simule um toque (tap) no avatar. c. Verifique se o showModalBottomSheet apareceu."

**[IA] Resposta:** A IA gerou o comando flutter pub add dev:mockito, o ficheiro de teste unitário prefs\_service\_test.dart (que passou) e o ficheiro widget\_test.dart (que falhou).

**[Aluno] Dificuldade:** *(Aluno envia o log de erro do flutter test) type 'Null' is not a subtype of type 'Future<bool>' Bad state: Cannot call 'when' within a stub response*

**[IA] Correção 1 (Falha):** A IA diagnosticou incorretamente que o problema era um async na função setUp e removeu-o.

**[Aluno] Dificuldade:** *(Aluno envia o mesmo log de erro, provando que a correção falhou)*

**[IA] Correção 2 (Falha):** A IA diagnosticou que o problema era a sintaxe antiga do mockito (v4 vs v5) e atualizou as chamadas when() para when(() => ...).

**[Aluno] Dificuldade:** "continua com erro"

**[IA] Correção 3 (Sucesso):** A IA concluiu que o mockito estava a ser problemático com o getIt e propôs uma nova estratégia: "Vamos abandonar o mockito no teste de widget... Vamos usar uma abordagem mais simples e robusta: criar um '**Fake**' (um 'Serviço Falso')". *(IA gerou um novo widget\_test.dart usando uma classe FakePrefsService em vez do mockito).*

**[Aluno] Dificuldade (Final):** *(Aluno envia um último log de erro) Bad state: GetIt: Object/factory... is not registered inside GetIt.*

**[IA] Correção 4 (Sucesso Final):** A IA diagnosticou o erro final de getIt: "O SplashScreen está a tentar usar o getIt antes de o nosso teste ter tido tempo de 'injetar' o FakePrefsService." *(IA gerou o widget\_test.dart final, removendo getIt.reset() e garantindo que o FakePrefsService era registado corretamente antes do teste rodar).*

**[Aluno] Validação:** PS C:\Cflutter\_projetos\Leafy> flutter test 00:03 +4: All tests passed!