Apresentação: SkillSeeds - Features de Lições e Conquistas (IA-Assisted)

1. Sumário Executivo

Este documento detalha a implementação de duas novas features centrais para o aplicativo SkillSeeds: a Tela de Lições (vinculada às Trilhas) e a Tela de Conquistas (vinculada ao perfil do usuário).

O objetivo principal foi expandir a funcionalidade do aplicativo, migrando de uma tela principal estática para um sistema dinâmico que busca dados de um backend Supabase.

Para acelerar este desenvolvimento, utilizamos uma abordagem de "estudo guiado com IA". Definimos a arquitetura de dados (o padrão Entity/DTO/Mapper que já havíamos estabelecido) e o fluxo de navegação. Em seguida, fornecemos prompts detalhados à IA para gerar o código repetitivo (boilerplate) para os novos Repositórios, Providers (Riverpod) e Telas (UI), garantindo consistência com o restante do projeto.

Resultados:

- Implementamos com sucesso a LessonListScreen, que exibe lições dinâmicas por trilha.
- Implementamos com sucesso a AchievementListScreen, que exibe as conquistas do usuário.
- Validamos que o padrão de arquitetura (DTO/Mapper/Repository) é escalável.
- Confirmamos que a IA é uma ferramenta eficaz para acelerar a criação de features quando guiada por uma arquitetura sólida.

2. Arquitetura e Fluxo de Dados

A arquitetura segue um fluxo de dados claro, gerenciado pelo Riverpod, com uma separação estrita de responsabilidades.

Diagrama de Fluxo (ASCII) - Feature 1 (Tela de Lições)

```
(Usuário Clica na Trilha "Design" na HomeScreen)
            |
            v
     [UI: HomeScreen]
(Navega para /lessons com 'Track' como argumento)
            |
            v
   [UI: LessonListScreen]
(ref.watch(lessonsForTrackProvider(track.id)))
            |
            v
 [Provider: lessonsForTrackProvider (FutureProvider.family)]
      (chama o repositório)
            |
            v
     [Repository: LessonRepository]
      (chama o Supabase e o Mapper)
            |
            v
  [Supabase (Backend)] <--- (JSON) --- [LessonDTO]
 (Tabela 'lessons')              (Converte)
            ^                      |
            |                      v
            ------------------ [LessonMapper]
                                   |
                                   v
                             [Lesson (Entity)]
                              (Modelo do App)
          |
          v
(Retorna List para a UI)
```

Papel da IA no Fluxo

A IA (Assistente do VS Code) foi utilizada como uma ferramenta de geração de código padronizado. Ela não tomou decisões de arquitetura.

- **Input (Nosso Papel):** Fornecemos à IA (1) o **Contexto** (nossas classes Track, LessonDTO, LessonMapper existentes) e (2) a **Intenção** (ex: "crie um repositório que busca dados da tabela 'lessons' e os mapeia").
- **Output (Papel da IA):** A IA gerou o "primeiro rascunho" (boilerplate) das novas classes (LessonRepository), as atualizações nos providers.dart e a estrutura básica das novas telas (LessonListScreen), que nós então revisamos, ajustamos e integramos.

3. Feature 1: Tela de Lições da Trilha

3.1. Objetivo

Permitir que o usuário clique em uma Trilha (ex: "Design") na tela principal (HomeScreen) e seja levado a uma nova tela (LessonListScreen) que exibe a lista de todas as lições associadas àquela trilha, buscando os dados dinamicamente do Supabase.

3.2. Prompts Usados (Estudo Guiado com IA)

Para construir esta feature, dividimos o trabalho em 3 prompts:

Prompt 1: Criar o Repositório de Dados

> **Nosso Prompt para a IA:** "Olá! Estou expandindo meu app Flutter (SkillSeeds) e preciso buscar dados da minha nova tabela lessons no Supabase.
>
> Eu já tenho um LessonDTO (em lib/models/lesson_dto.dart) e um LessonMapper (em lib/mappers/lesson_mapper.dart).
>
> Por favor, crie um novo arquivo lib/repositories/lesson_repository.dart que contenha a classe LessonRepository.
>
> Esta classe deve:
>
> 1. Ter um construtor que recebe um SupabaseClient (igual ao meu TrackRepository existente).
> 2. Ter um método público: Future<List> getLessonsForTrack(int trackId).
> 3. Dentro deste método, você deve: a. Chamar o supabase.from('lessons').select() b. Filtrar o resultado usando .eq('track_id', trackId) c. Mapear a lista de Map<String, dynamic> resultante: primeiro para LessonDTO.fromMap e depois para LessonMapper.toEntity. d. Retornar a List."

Prompt 2: Atualizar os Providers (Riverpod)

> **Nosso Prompt para a IA:** "Ótimo. Agora, preciso expor esse novo repositório para o meu app usando Riverpod.
>
> Por favor, atualize meu arquivo lib/providers/providers.dart (vou fornecer o contexto dele) e adicione dois novos providers:
>
> 1. Um lessonRepositoryProvider (um Provider simples que assiste o supabaseClientProvider e retorna uma instância de LessonRepository).
> 2. Um lessonsForTrackProvider (um FutureProvider.family<List, int>). Este provider deve: a. Receber um int trackId como parâmetro da família. b. Assistir (watch) o lessonRepositoryProvider. c. Chamar e retornar o método repository.getLessonsForTrack(trackId)."

Prompt 3: Criar a Tela (UI)

> **Nosso Prompt para a IA:** "Perfeito. Por fim, preciso da nova tela para exibir esses dados.
>
> Crie um novo arquivo lib/screens/lesson_list_screen.dart para a classe LessonListScreen.
>
> A tela deve:
>
> 1. Ser um ConsumerWidget.
> 2. Ter uma AppBar.
> 3. No método build, ela deve primeiro extrair o argumento Track que foi passado pela navegação, usando ModalRoute.of(context)!.settings.arguments as Track.
> 4. O título da AppBar deve ser o track.name.
> 5. O body da tela deve usar ref.watch(lessonsForTrackProvider(track.id)) e um .when() para tratar os estados de loading (mostrar CircularProgressIndicator), error (mostrar Text('Erro')), e data (mostrar um ListView.builder com as lições)."

3.3. Exemplos de Entrada e Saída

- **Entrada (Exemplo 1):** Usuário clica na Trilha "Design" (ID 1).
  - **Saída:** A IA (simulada) gera o LessonRepository. O app navega para /lessons, chama lessonsForTrackProvider(1). O Supabase retorna 3 lições. A tela exibe: "O que é UI/UX?", "Princípios de Gestalt", "Quiz Rápido: UI/UX".
- **Entrada (Exemplo 2):** Usuário clica na Trilha "Desenvolvimento" (ID 2).
  - **Saída:** A IA (simulada) gera o LessonRepository. O app navega para /lessons, chama lessonsForTrackProvider(2). O Supabase retorna 1 lição. A tela exibe: "Atalhos do VS Code".
- **Entrada (Exemplo 3):** Usuário clica em uma Trilha (ID 3) que não tem lições.
  - **Saída:** A IA (simulada) gera o LessonRepository. O app chama lessonsForTrackProvider(3). O Supabase retorna uma lista vazia. A tela exibe: "Nenhuma lição encontrada para esta trilha."

3.4. Como Testar Localmente (Passo a Passo)

1. Garantir que o Supabase está configurado (SQL executado, .env preenchido).
2. Executar o aplicativo (flutter run -d chrome).
3. Se for um novo usuário, passar pelo fluxo de Onboarding e Consentimento (Observação: este fluxo está atualmente com um bug que impede a navegação para a Home).
4. *Assumindo que o bug do onboarding foi corrigido:* Na HomeScreen, clicar no card da trilha "Design".
5. **Verificação:** O app deve navegar para a nova tela. O título da AppBar deve ser "Design". O corpo da tela deve mostrar um *loading* e, em seguida, a lista de lições de Design cadastradas no Supabase.

3.5. Limitações e Riscos

- **Limitação (IA):** A IA gerou o código de busca, mas não implementou cache local (conforme o Guia 082). Cada vez que a tela é aberta, ela faz uma nova chamada ao Supabase, o que consome dados e é lento.
- **Risco (IA):** A IA gerou um tratamento de erro genérico (Text('Erro')). Em um app de produção, precisaríamos de um tratamento mais robusto (ex: um botão de "Tentar Novamente").
- **Risco (Privacidade):** Nenhum risco de privacidade identificado, pois os prompts não contêm chaves de API e a tabela lessons é pública (conforme nossa política RLS).

3.6. Código Gerado pela IA (Explicação)

A IA gerou o LessonRepository. O trecho mais importante é o método getLessonsForTrack:

```
// Comentário (Humano): Importamos as classes necessárias.
import 'package:supabase_flutter/supabase_flutter.dart';
import '../models/lesson.dart';
import '../models/lesson_dto.dart';
import '../mappers/lesson_mapper.dart';

// Comentário (Humano): A IA gerou esta classe conforme solicitado.
class LessonRepository {
  final SupabaseClient _supabase;

  LessonRepository(this._supabase);

  // Comentário (Humano): Este método foi gerado pela IA seguindo o prompt.
  Future<List<Lesson>> getLessonsForTrack(int trackId) async {
    try {
      // Comentário (IA): Busca dados da tabela 'lessons'.
      final data = await _supabase
          .from('lessons')
          .select()
          // Comentário (IA): Filtra pelo 'track_id' recebido.
          .eq('track_id', trackId);

      // Comentário (Humano): A IA converte a lista de Maps...
      final lessons = (data as List).map((map) {
        // Comentário (Humano): ...primeiro para DTO (espelho do banco)...
        final dto = LessonDTO.fromMap(map);
        // Comentário (Humano): ...e depois para Entity (modelo do app).
        return LessonMapper.toEntity(dto);
      }).toList();

      return lessons;
    } catch (e) {
      // Comentário (IA): Tratamento de erro básico.
      throw Exception('Erro ao buscar lições: $e');
    }
  }
}
```

4. Feature 2: Tela de Conquistas (Achievements)

4.1. Objetivo

Criar uma nova tela "Minhas Conquistas" e adicionar um link para ela no menu Drawer (menu lateral) da HomeScreen, permitindo ao usuário ver as conquistas que ele obteve (dados buscados do Supabase).

4.2. Prompts Usados (Estudo Guiado com IA)

O processo foi similar, mas dividido em 3 prompts:

Prompt 1: Criar o Repositório de Conquistas

> **Nosso Prompt para a IA:** "Preciso de outra feature. Crie um AchievementRepository (lib/repositories/achievement_repository.dart) baseado nas minhas entidades AchievementDTO e AchievementMapper (que já criei). A classe deve ter um método Future<List> getAchievements() que busca todos os dados da tabela achievements e os mapeia de DTO para Entity."

Prompt 2: Atualizar os Providers e Criar a Tela

> **Nosso Prompt para a IA:** "Agora, atualize lib/providers/providers.dart para adicionar um achievementRepositoryProvider e um achievementsProvider (um FutureProvider simples que chama getAchievements()).
>
> Em seguida, crie a tela lib/screens/achievement_list_screen.dart (um ConsumerWidget) com um AppBar ('Minhas Conquistas'). O body deve usar ref.watch(achievementsProvider).when(...) para exibir um ListView com o achievement.title e achievement.description."

Prompt 3: Atualizar o Menu (Drawer)

> **Nosso Prompt para a IA:** "Por fim, preciso que você modifique o Drawer no meu arquivo lib/screens/home_screen.dart.
>
> Adicione um novo ListTile com o título 'Minhas Conquistas' e o ícone Icons.emoji_events_outlined. O onTap deste ListTile deve fechar o drawer e navegar para a nova rota AppRoutes.achievements."

4.3. Como Testar Localmente (Passo a Passo)

1. Garantir que o Supabase está configurado (SQL executado, .env preenchido).
2. Executar o aplicativo.
3. Clicar no novo item de menu "Minhas Conquistas".
4. **Verificação:** O app deve navegar para a nova tela. O título da AppBar deve ser "Minhas Conquistas". O corpo deve mostrar um *loading* e, em seguida, a lista de conquistas cadastradas no Supabase (ex: "Primeira Lição").

5. Política de Branches e Commits (Metodologia AINDA não adotada)

Para esta entrega, adotei uma política de "feature branching" para isolar o desenvolvimento:

1. **Branch Principal:** main (protegida, representando a produção).
2. **Branch de Desenvolvimento:** Criei uma branch chamada feat/dynamic-content a partir da main.
3. **Commits Atômicos:** Registrei o progresso em commits pequenos e claros. O histórico do Git (resumido) seguiu este padrão:

```
$ git log --oneline
(HEAD -> feat/dynamic-content)
    feat: Adiciona Feature 1 (Tela de Lições) e Feature 2 (Tela de Conquistas)
    feat: Adiciona entidade Achievement (Entity, DTO, Mapper, Test)
    feat: Adiciona entidade UserProgress (Entity, DTO, Mapper, Test)
    feat: Adiciona entidade Step (Entity, DTO, Mapper, Test)
    feat: Adiciona entidade Lesson (Entity, DTO, Mapper, Test)
    refactor: Implementa padrão DTO e Mapper para Tracks
    ...
(origin/main, main)
```