

Receitas do Guaxinim

Apresentação II

João Rezende, Lucas Branco e Pedro Caetano

Objetivo principal do aplicativo

Principais Funcionalidades:

- Variedade de receitas categorizadas (como massas, sobremesas, saladas, entre outras)
- Interface intuitiva e visualmente atrativa
- Maneira prática e organizada de encontrar receitas culinárias para o dia a dia
- Salvar suas receitas favoritas para acesso rápido

Público-alvo:

- Pessoas interessadas em cozinhar em casa
- Iniciantes até cozinheiros mais experientes

Motivação e diferenciais

Oferecer uma solução simples, acessível e organizada para quem busca cozinhar em casa, com foco em uma experiência intuitiva e leve.

Referências para o desenvolvimento

- **TudoGostoso**
- **Tasty**

Similaridades

- Imagens dos pratos
- Variedade culinária
- Divisão em categorias

Diferenciais

- Simplificações
- Criar suas próprias receitas



Organização do código

```
lib
├── core
├── data
│   └── datasources
│       ├── comment_remote_datasource.dart
│       ├── recipe_local_datasource.dart
│       ├── recipe_remote_datasource.dart
│       ├── review_remote_data_source.dart
│       └── review_remote_datasource.dart
│   └── db
│       └── app_database.dart
│   └── repositories
│       ├── comment_repository_impl.dart
│       ├── recipe_repository_impl.dart
│       ├── review_repository_impl.dart
│       └── storage_repository.dart
├── domain
│   ├── entities
│   │   ├── comment.dart
│   │   ├── profile.dart
│   │   ├── recipe.dart
│   │   └── review.dart
│   └── repositories
│       ├── comment_repository.dart
│       ├── recipe_repository.dart
│       └── review_repository.dart
```

```
features
├── auth
├── comments
├── profile
│   ├── view
│   │   └── profile_page.dart
│   └── viewmodel
│       ├── favorite_recipes_viewmodel.dart
│       ├── my_recipes_viewmodel.dart
│       └── profile_viewmodel.dart
├── recipes
├── reviews
└── main.dart
```

Arquitetura MVVM

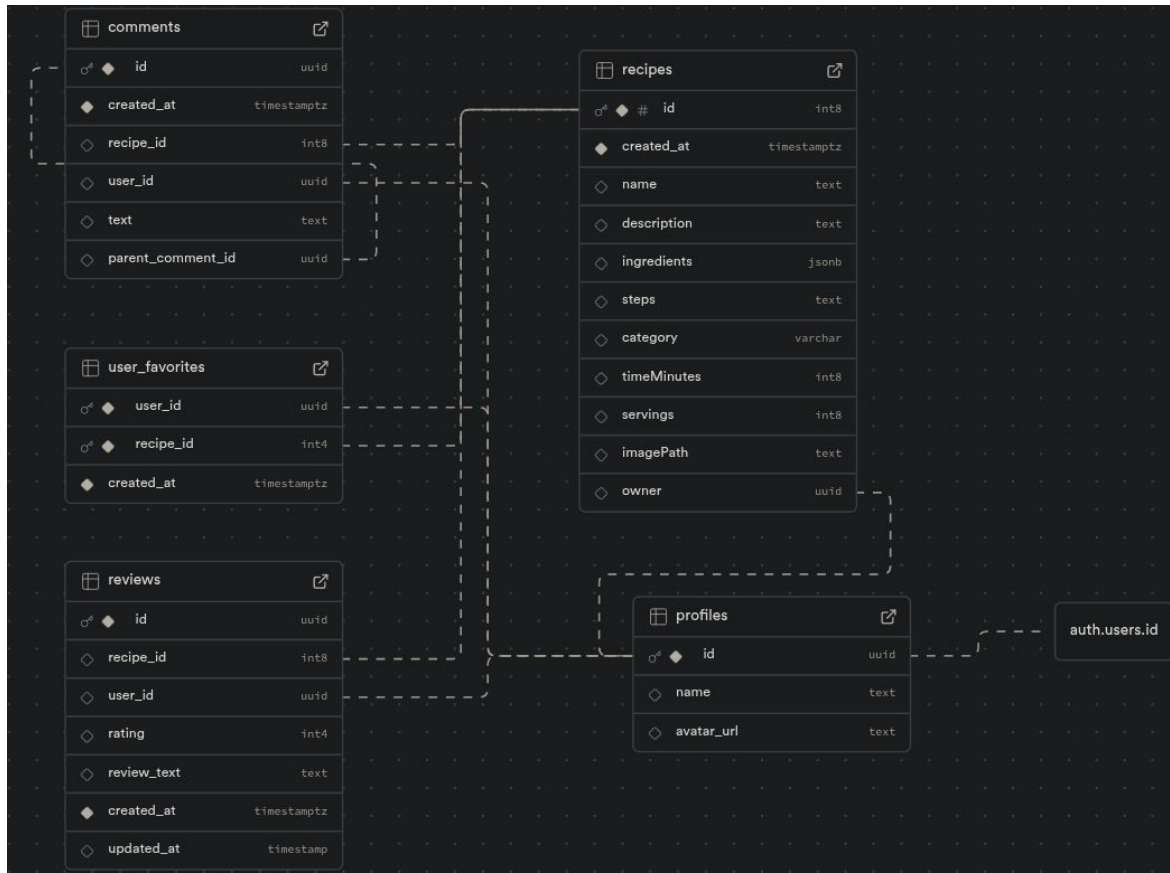
```
integration_test
├── app_integration_test.dart
├── recipe_review_test.dart
├── ios
├── lib
├── linux
├── macos
├── test
│   ├── domain
│   │   ├── entities
│   │   │   └── review_test.dart
│   ├── features
│   │   ├── recipes
│   │   │   └── view
│   │   │       ├── recipe_detail_page_widgets_test.dart
│   │   │       ├── recipe_unit_test.dart
│   │   │       └── recipe_widget_test.dart
```

Dependências

Riverpod

```
cupertino_icons: ^1.0.8
flutter_riverpod: ^2.6.1
go_router: ^16.2.0
sqflite: ^2.4.2
sqflite_common_ffi: ^2.3.6
path_provider: ^2.1.5
path: ^1.9.1
image_picker: ^1.2.0
supabase_flutter: ^2.10.0
flutter_dotenv: ^6.0.0
intl: ^0.20.2
```

Comunicação com supabase



Testes

Testes de unidade

- Teste de review (testando funções da entidade)
 - `copyWith` should update specified fields and keep others
- Teste de criação de receita (map para entidade, entidade para map)

```
// Teste para o factory constructor 'fromMap' com o formato de dados atual.  
test('Deve criar uma Receita a partir de um Map (formato atual)', () {
```

- Teste de filtrar receitas corretamente ao usar a busca.

```
await viewModel.setSearch('Bolo');  
  
// ASSERT  
final state = container.read(homeVmProvider);  
expect(state.categorizedRecipes.length, 1);  
expect(state.categorizedRecipes.containsKey('Sobremesas'), isTrue);  
expect(state.categorizedRecipes['Sobremesas']!.first.name, 'Bolo de Chocolate');
```

Testes

Testes de widgets

- Teste do widget das estrelas das reviews
- Teste do widget de card de receita
 - Informações exibidas estão corretas?
 - Botão de favoritar dispara função?
- Teste do widget com validação visual (tema/cor) e funcional (toques) do Recipe Card com imagens de rede mockadas.

Testes

Testes de integração

- Teste de criar ou atualizar uma review em uma receita
- Teste de login
- Teste de fluxo criação de uma receita

Apresentação do aplicativo

Dificuldades enfrentadas

- Novas tecnologias
- Pouco tempo
- Ideia do app pouco original

Possíveis melhorias

- Integração com LLMs
 - Gerar/modificar receitas
 - Recomendações de cozinha
 - Recomendações de receitas