Composição do Time:

Nome	Responsabilidade Principal
Jáder	Scrum Master / Dev
Giulia	Product Owner / UI/UX
Gabrielly	Dev (Wireframes ou Código)
João	Dev (Wireframes ou Código)
Karen	Documentação e Testes
Gustavo	Dev (Interação e Apresentação)

Historias do usuário:

- **Criar um Plano de Treino**
  - O usuário pode adicionar exercícios com nome, repetições e tempo de duração.
  - O treino pode ser salvo e reutilizado em outros dias.
  - O usuário pode editar ou excluir exercícios do plano.
- **Registrar Progresso Diário**
  - O usuário pode marcar cada exercício como concluído.
  - Deve haver um resumo diário mostrando quais treinos foram feitos.
  - O sistema pode sugerir ajustes com base no progresso.
- **Receber Notificações de Treino**
  - O usuário pode definir horários para os lembretes.
  - O sistema deve enviar notificações automáticas.
  - O lembrete pode ser diário ou em dias específicos.

Trabalho:

Back-End (Joao e Gaby)

Os desenvolvedores Gabrielly e João encarregados de trabalhar no funcionamento da persistência do aplicativo desenvolveram as funcionalidades para garantir a persistência garantindo que o progresso diário e o plano de treino estejam de acordo e trabalhando entre si a partir das funções:

```dart
Future<void> _loadData() async {
    final prefs = await SharedPreferences.getInstance();
    // Carregar planos de treino
    final workoutPlansJson = prefs.getStringList('workoutPlans') ?? [];
    setState(() {
      _workoutPlans.clear();
      for (var json in workoutPlansJson) {
        _workoutPlans.add(WorkoutPlan.fromJson(jsonDecode(json)));
      }
    });
    // Carregar progresso diário
    final progressJson = prefs.getStringList('dailyProgress') ?? [];
    setState(() {
      _dailyProgress.clear();
      for (var json in progressJson) {
        _dailyProgress.add(DailyProgress.fromJson(jsonDecode(json)));
      }
    });
  }


  Future<void> _saveData() async {
    final prefs = await SharedPreferences.getInstance();
    // Salvar planos de treino
    final workoutPlansJson = _workoutPlans
        .map((plan) => jsonEncode(plan.toJson()))
        .toList();
    await prefs.setStringList('workoutPlans', workoutPlansJson);
    // Salvar progresso diário
    final progressJson = _dailyProgress
        .map((progress) => jsonEncode(progress.toJson()))
        .toList();
    await prefs.setStringList('dailyProgress', progressJson);
  }
```

> As funções de carregamento e salvamento foram reutilizadas para garantir a reusabilidade do código e melhorar a manutenção também, desta forma a persistência é a mesma para os arquivos de progresso dos treinos, lista dos exercícios e os alarmes, mudando apenas a forma dos JSON.

Foi escolhido também uma arquitetura baseada em POO, para melhorar na escalabilidade do projeto, garantindo um bom paradigma para construção do projeto.

```dart
class Exercise {
  final String name;
  final int repetitions;
  final int duration; // em minutos

  Exercise({
    required this.name,
    required this.repetitions,
    required this.duration,
  });

  Map<String, dynamic> toJson() {
    return {
      'name': name,
      'repetitions': repetitions,
      'duration': duration,
    };
  }
```

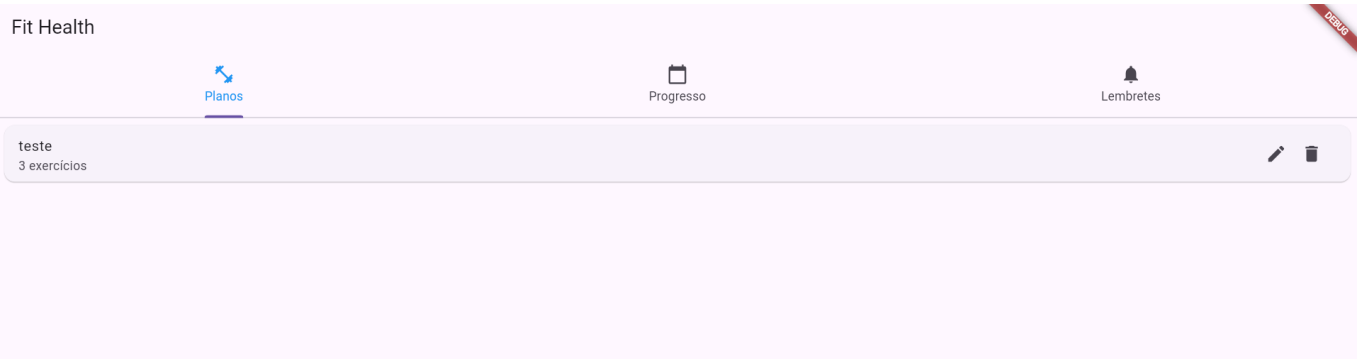
```dart
class DailyProgress {
  final DateTime date;
  final int workoutPlanId;
  final List<Exercise> completedExercises;

  DailyProgress({
    required this.date,
    required this.workoutPlanId,
    required this.completedExercises,
  });
```

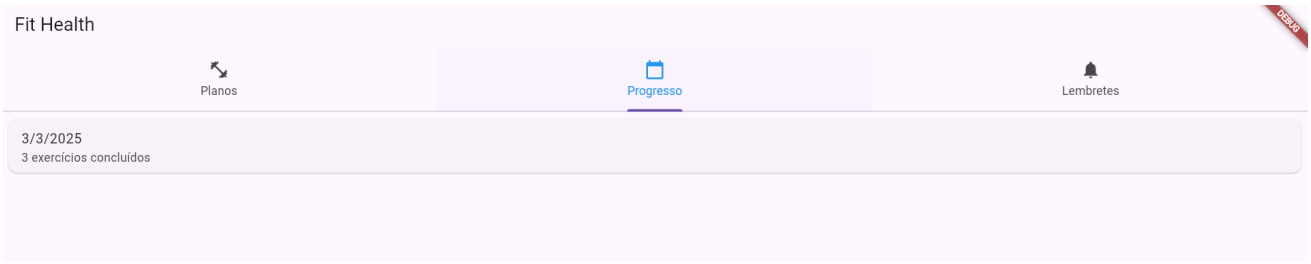
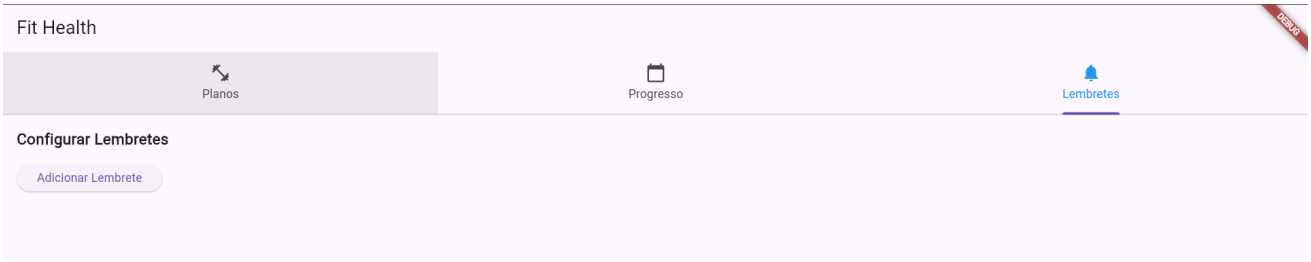
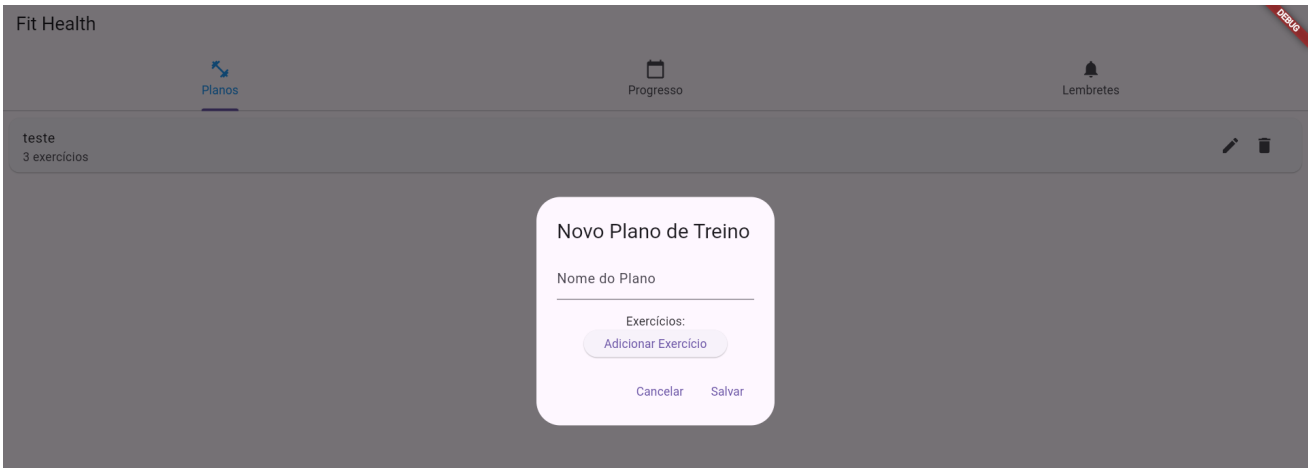
```dart
class WorkoutPlan {
  final String name;
  final List<Exercise> exercises;

  WorkoutPlan({
    required this.name,
    required this.exercises,
  });
```

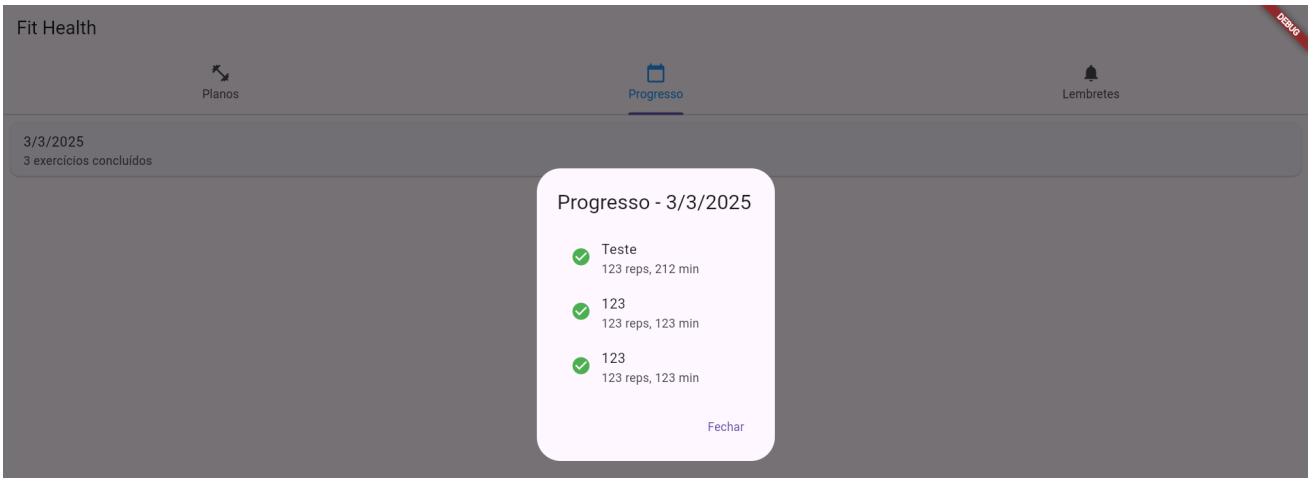
Front-End (Gustavo e Karen)

O front end idealizado por Gustavo e Karen que realizou os testes foi idealizado de uma maneira, porém acabou ficando um pouco diferente devido ao tempo e as também devido a

propria limitação do Flutter que não aceitou a portação direta dos arquivos exportados, então a Interface teve de ser feita de outra forma adequando funcionalidade sobre a estética do aplicativo entre si, mas ainda entregado um aplicativo viavel e intuitivo:



> O aplicativo é dado por uma interface onde podemos ver os planos criados e atualizar o progresso e também adicionar os lembretes, navegando por uma AppBar simples.
>
> 
>
>
> O progresso atualizando a partir do plano "Teste" visto na imagem passada.
>
> 
>
>
> Os lembretes funcionando, o usuário pode adicionar os lembretes para serem lembrados e motivados a treinar!
>
> 
>
>
> O usuário pode criar um novo plano de treino que deseja!



E depois ir checando seu progresso, marcando o que ja foi feito!

A partir da persistência feita por arquivos JSON, podemos garantir a integridade das informações do usuário!

Análises do Produto (Giulia)

A Product Owner inicialmente definiu que a Sprint seria conduzida com um protótipo no **Canvas em PowerPoint**, devido à limitação de tempo. No entanto, ao longo do desenvolvimento, houve uma reavaliação conjunta com o **Scrum Master**, e decidiu-se migrar para uma abordagem em código para construir um **MVP funcional**. Com essa mudança, o design previamente elaborado foi descartado para que a equipe pudesse apresentar uma versão operacional do aplicativo.

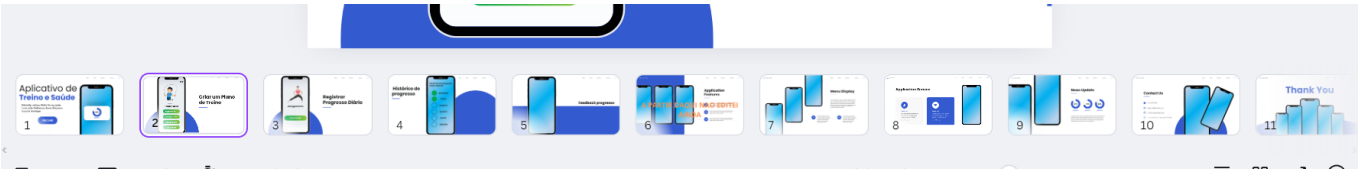


Imagem da interface que havia sido desenvolvida no Canvas.

A transição para o código trouxe alguns desafios, principalmente em relação à **usabilidade**, conforme relatado pelo **Product Owner**. Como resultado, foi necessário implementar melhorias na experiência do usuário, incluindo a **persistência de dados**, garantindo que as informações fossem salvas corretamente e que a navegação fosse mais intuitiva.

Facilitação do trabalho (Jáder)

O scrum master atuou como um facilitador para o desenvolvimento do código Garantindo que o processo Scrum seja seguido corretamente, assim como também ajudando no manuseio da framework escolhida para garantir o desenvolvimento eficiente, e solucionando problemas de dependencia gerados pela migração do codigo por conta da transição do uso dos dados que foi acarretado devido as melhoras sugeridas pela Product Owner, além disso o Scrum Master

atuou como um desenvolvimento na linha de frente responsável pela interatividade e navegçaão entre as janelas, responsavle também pela idealização das questões principais e configurações inicias do aplicativo de gerenciamento de estados

```
class _HomePageState extends State<HomePage> {
  final List<WorkoutPlan> _workoutPlans = [];
  final List<DailyProgress> _dailyProgress = [];
  final FlutterLocalNotificationsPlugin _notificationsPlugin =
FlutterLocalNotificationsPlugin();

  @override
  void initState() {
    super.initState();
    _initializeNotifications();
    _loadData();

  }

  Future<void> _initializeNotifications() async {
    const AndroidInitializationSettings initializationSettingsAndroid =
        AndroidInitializationSettings('@mipmap/ic_launcher');
    final InitializationSettings initializationSettings =
        InitializationSettings(android: initializationSettingsAndroid);
    await _notificationsPlugin.initialize(initializationSettings);
  }
```

E a Janela Principal:

```
@override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Fit Health'),
      ),

      body: DefaultTabController(
        length: 3,
        child: Column(
          children: [
            const TabBar(
              labelColor: Colors.blue,
              tabs: [
                Tab(icon: Icon(Icons.fitness_center), text: 'Planos'),
                Tab(icon: Icon(Icons.calendar_today), text: 'Progresso'),
                Tab(icon: Icon(Icons.notifications), text: 'Lembretes'),
```

```
            ],
          ),
          Expanded(
            child: TabBarView(
              children: [
                _buildWorkoutPlansTab(),
                _buildProgressTab(),
                _buildNotificationsTab(),
              ],
            ),
          ),
        ],
      ),
    ),
    floatingActionButton: FloatingActionButton(
      onPressed: () => _showAddWorkoutPlanDialog(context),
      child: const Icon(Icons.add),
      tooltip: 'Adicionar Plano de Treino',
    ),
  );
}
```