ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

на прохождение производственной практики обучающегося направления подготовки/ специальности 09.03.02 Информационные системы и технологии

1. Фамилия, имя, отчество обучающегося: Воробьев Владислав Алексеевич

2. Группа: 4128

3. Тема индивидуального задания: Разработка проекта информационной системы справочника коктейлей

4. Исходные данные:
   - Провести анализ предметной области.
   - Разработать модели бизнес-процессов в BPMS. Представить алгоритмы выполнения бизнес-процессов в соответствии с созданными моделями.
   - Создать реляционную модель базы данных.

5. Содержание отчетной документации:

5.1. индивидуальное задание;

5.2. отчёт, включающий в себя:
   - титульный лист;
   - формулировку задачи и способов реализации;
   - описание технологии разработки проекта информационной системы;
   - выводы по результатам практики;
   - список использованных источников.

5.3. отзыв руководителя от профильной организации (при прохождении практики в профильной организации).

6. Срок представления отчёта на кафедру: «21» июля 2023 г.

Руководитель практики

зав. кафедрой №42, д.т.н., доц.		С.В. Мичурин
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

СОГЛАСОВАНО
Руководитель практики от профильной организации

должность	подпись, дата	инициалы, фамилия

Задание принял к исполнению:

Обучающийся		В.А. Воробьев
дата	подпись	инициалы, фамилия

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение высшего образования
«Санкт–Петербургский государственный университет
аэрокосмического приборостроения»

Кафедра информационных систем и технологий

ОТЧЁТ ПО ПРАКТИКЕ
ЗАЩИЩЁН С ОЦЕНКОЙ

РУКОВОДИТЕЛЬ

Зав. кафедрой №42, д.т.н., доцент		С.В. Мичурин
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЁТ ПО ПРАКТИКЕ

вид практики производственная

тип практики технологическая (проектно-технологическая)

на тему индивидуального задания Разработка проекта информационной системы справочника коктейлей

выполнен Воробьев Владислав Алексеевич
фамилия, имя, отчество обучающегося в творительном падеже

по направлению подготовки 09.03.02 Информационные системы и технологии
код наименование направления

наименование направления

направленности 04 Информационные технологии
код наименование направленности

в медиаиндустрии
наименование направленности

Обучающийся группы № 4128		В.А. Воробьев
номер	подпись, дата	инициалы, фамилия

Санкт-Петербург 2023

СОДЕРЖАНИЕ

1 Анализ предметной области

Для разработки информационной системы была выбрана сфера кулинарии, а именно рецепты коктейлей. В ходе анализа предметной области были оценены приложения конкурентов - самым сильным и популярным является Inshaker (URL: https://ru.inshaker.com/).

Были сделаны такие выводы:

1. Пользователей интересуют ИС, которые сочетают информацию о коктейлях и информацию об ингредиентах.
2. Пользователи не очень активно рассматривают социальный аспект ИС (комментарии, лайки и т.д.)
3. Пользователей интересует просмотр рецептов в offline, так и online.
4. Большинство пользователей пользуется ИС не регулярно.
5. Пользователей интересуют ИС в которых встроена система рекомендаций.

В данной ИС можно выделить 2 вида объектов:

1. Коктейли: название, описание, ингредиенты, шаги.
2. Ингредиент: название, тип, описание.

В ходе проектирования было решено сделать ИС как мультиплатформенное приложение (Windows/Linux, Android/Ios).

Можем выделить прецеденты использования для пользователя (см. рис. 1).

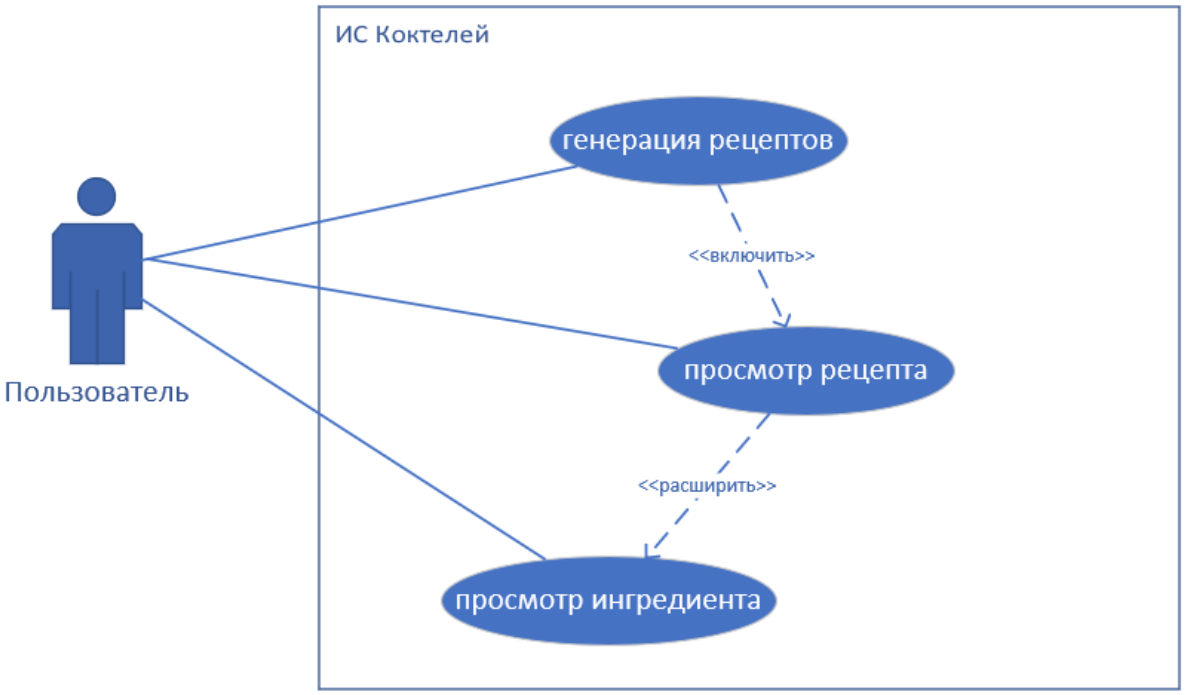

Рисунок 1 – Use-Case UML диаграмма

2 Проектирование ИС

Для реализации информационной системы было решено написать приложение на языке программирования Dart с использованием фреймворка Flutter [1].

Также build проекта был нацелен на следующие платформы:

1. Основные
    a. Android
    b. Windows
2. Дополнительные
    a. Ios
    b. Linux

Была разработана Mindmap для наброска функционала ИС (см. рис. 2), модель бизнес-процессов с использованием нотации BPMN (см. рис. 3), а также мокапы экранов в Figma (см. рис. 4) [2].

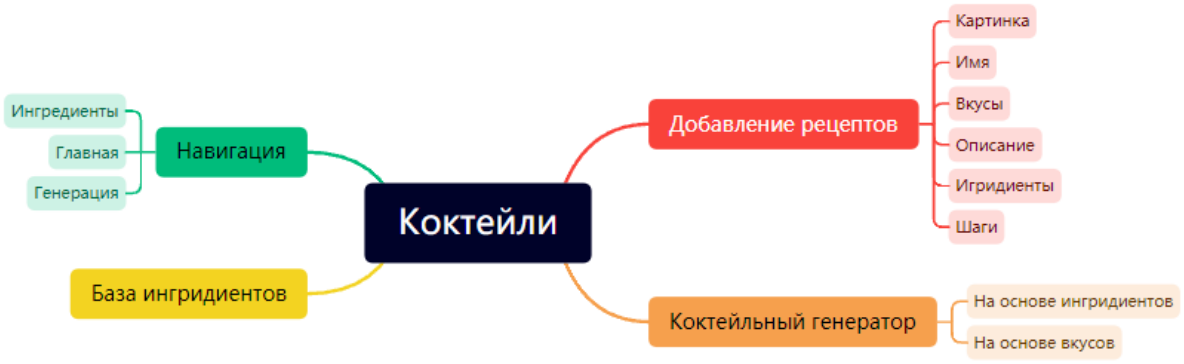

Рисунок 2 – Mindmap ИС

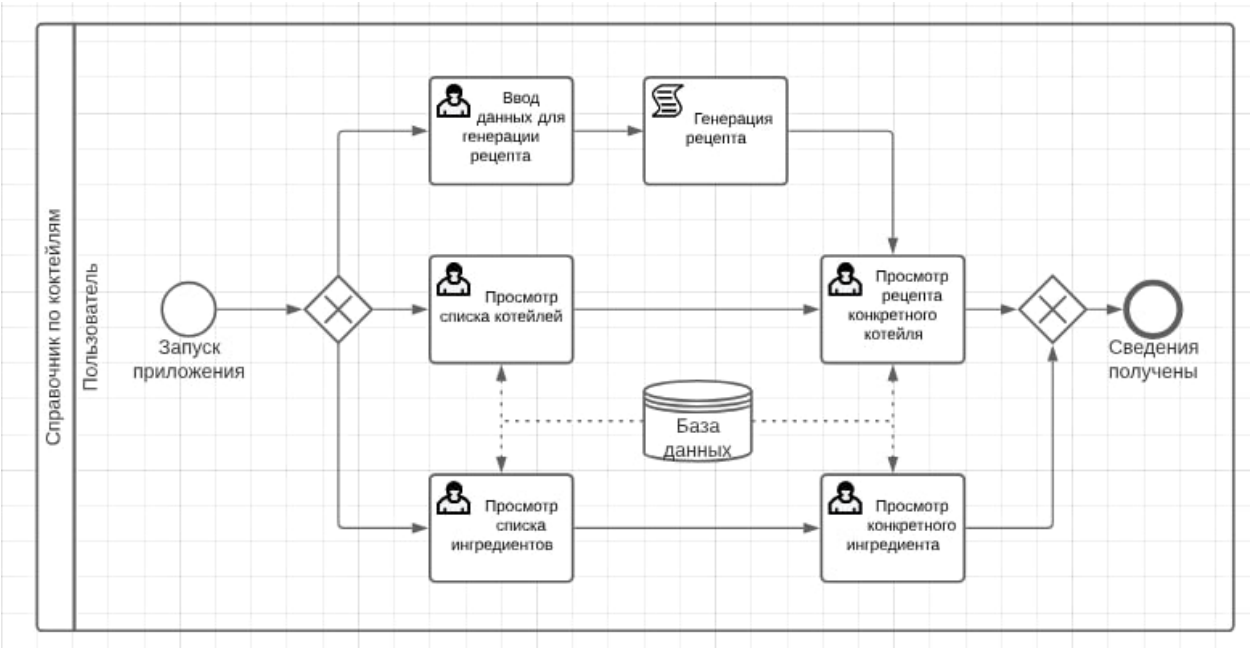

Рисунок 3 – BPMN диаграмма поиска рецептов

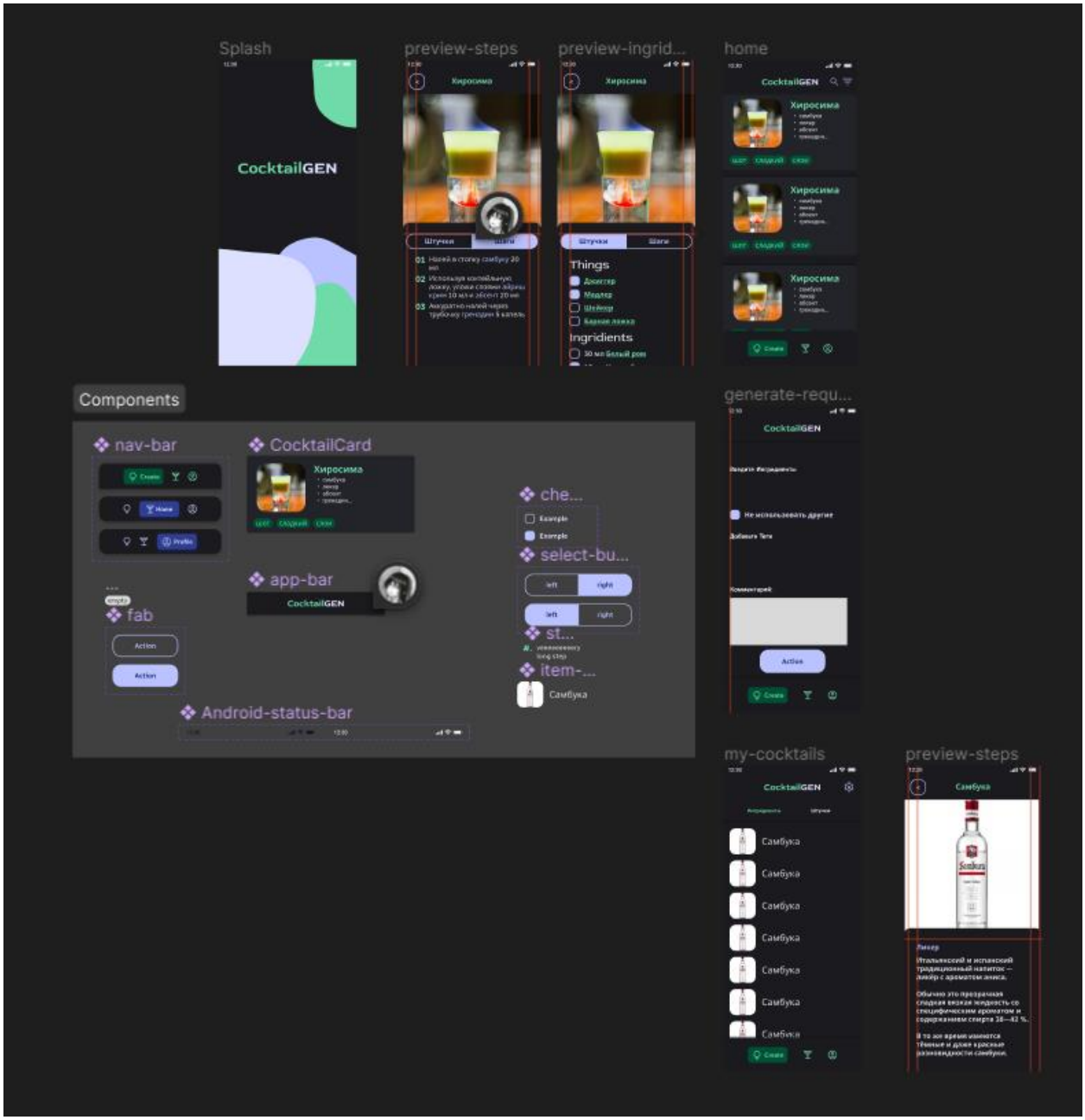

Рисунок 4 – Мокапы в Figma

3 Ход работы

При проектировании ИС было решено выбрать базу данных Isar [3]. Выбор пал именно на эту NoSQL технологию по ряду причин:

1. Прекрасная поддержка и большая популярность Isar в сообществе.
2. Интеграция с кодом Dart посредством код-гена, как итог – быстрый цикл разработки и высокая читаемость.
3. Существует свой инспектор, который позволяет редактировать базу данных через визуальный интерфейс.
4. Более легкая по сравнению с реляционными базами данных, и более быстрая и удобная в сравнении с аналогичными NoSql базами данных.
5. Из “коробки” доступен импорт/экспорт БД в JSON.
6. Поддержка Windows, Ios, Android платформ.

После выбора СУБД было решено спроектировать ER-диаграмму (см. рис. 5)

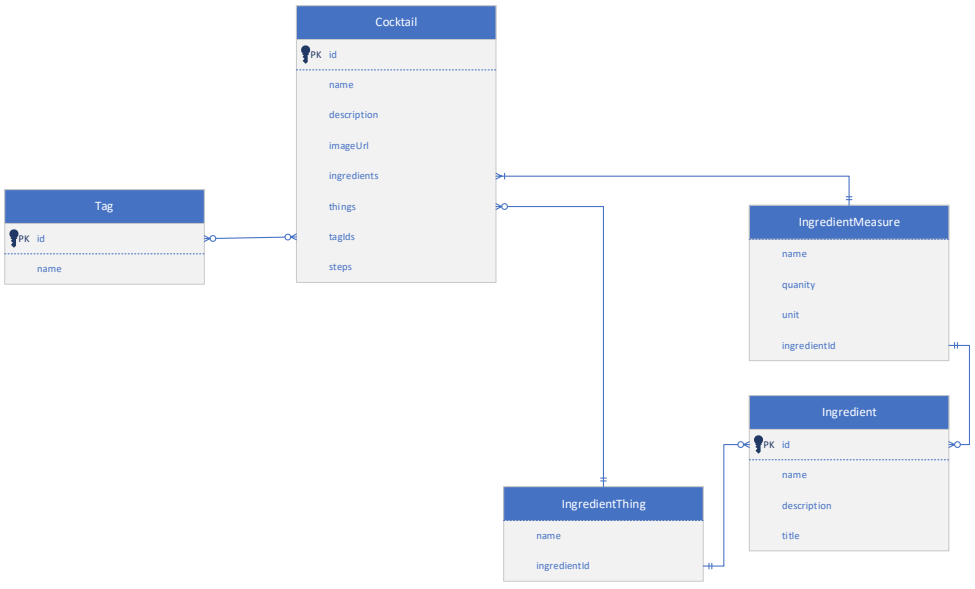

Рисунок 5 – ER-диаграмма базы данных

Стоит отметить следующий аспект, который может показать неоднозначным. IngredientThing и IngredientMeasure объявляют поле name и при этом связаны с Ingredient, которое также объявляет поле name. Это было сделано, так как name в IngredientThing и IngredientMeasure носит уточняющий характер (пример: Ingredient: Ром – IngredientThing: Plantation), а поле ingredientId нужно для возможности получить экземпляр Ingredient.

Далее представлен код моделей (см. рис 6-10) и пример заполненной БД (см. рис. 11 – 13).

cocktail_gen - cocktail_isar.dart

```dart
@collection
class CocktailIsar {
  final Id id;
  final String name;
  final String description;
  final String imageUrl;
  final List<Id> tagsIds;
  final List<IngredientMeasureIsar> ingredients;
  final List<IngredientThingIsar> things;
  final List<String> steps;

  CocktailIsar({
    this.id = Isar.autoIncrement,
    required this.name,
    required this.description,
    required this.imageUrl,
    required this.tagsIds,
    required this.ingredients,
    required this.things,
    required this.steps,
  });
}

```

Рисунок 6 – Модель “Cocktail”

cocktail_gen - ingredient_isar.dart

```dart
@collection
class IngredientIsar {
  final Id id;
  final String name;
  final String description;
  final String type;
  final String imageUrl;

  IngredientIsar({
    this.id = Isar.autoIncrement,
    required this.name,
    required this.description,
    required this.type,
    required this.imageUrl,
  });
}

```

Рисунок 7 – Модель “Ingredient”

cocktail_gen - ingredient_measure_isar.dart

```dart
enum MeasureUnitsIsar {
  none,
  spoons,
  pieces,
  milliliters,
  oz,
  leaves,
  drops,
}

@embedded
class IngredientMeasureIsar {
  late String name;
  late double quantity;
  @enumerated
  late MeasureUnitsIsar unit;
  late int ingredientId;
}

```

Рисунок 8 – Модель “IngredientMeasure”

cocktail_gen - ingredient_thing_isar.dart

```dart
@embedded
class IngredientThingIsar {
  late String name;
  late int ingredientId;
}

```

Рисунок 9 – Модель “IngredientThing”

cocktail_gen - tag_isar.dart

```dart
@collection
class TagIsar {
  final Id id;
  final String name;

  TagIsar(
    this.name, {
    this.id = Isar.autoIncrement,
  });
}

```

Рисунок 10 – Модель “Tag”

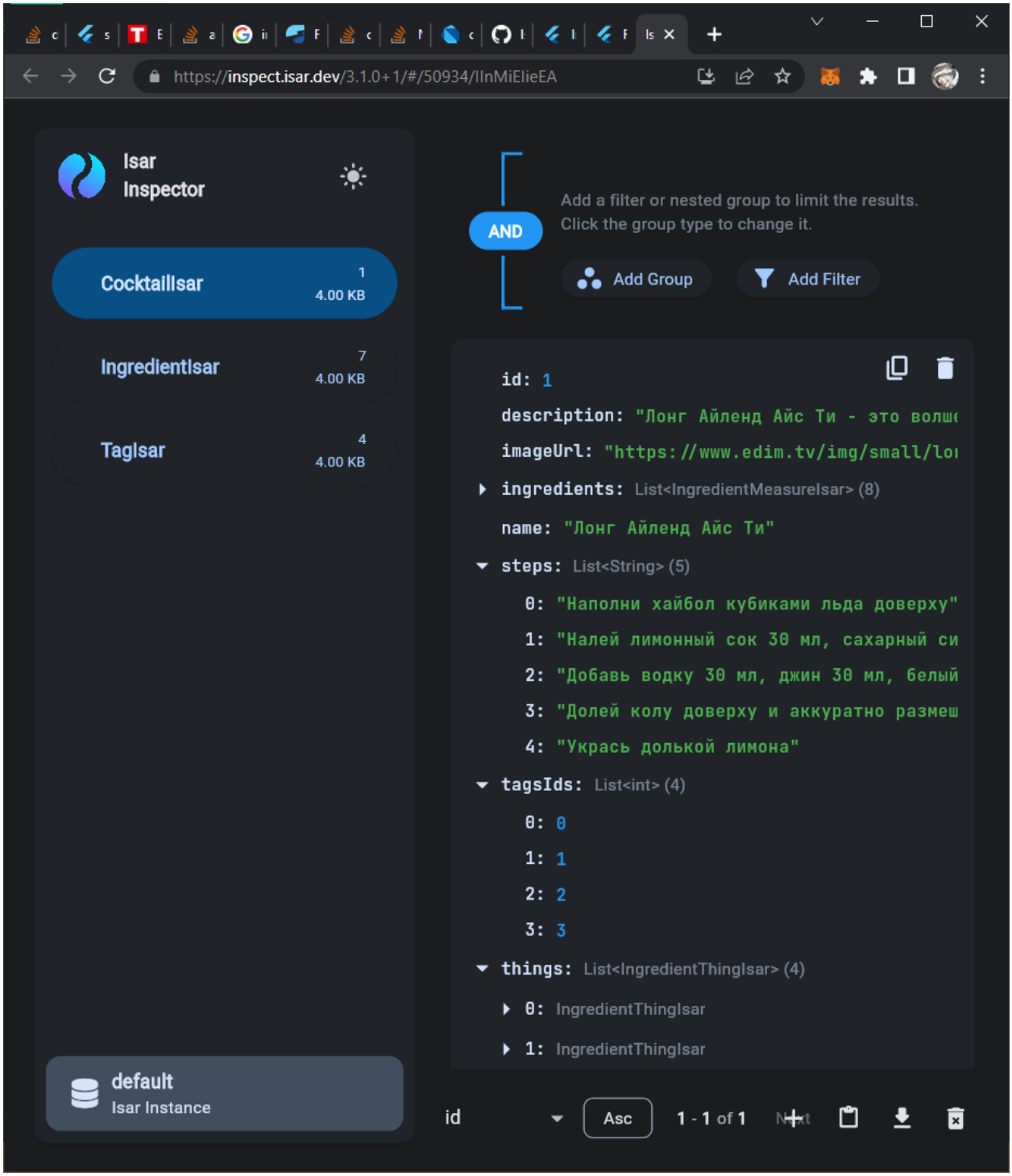

Рисунок 11 – Таблица “Cocktail”

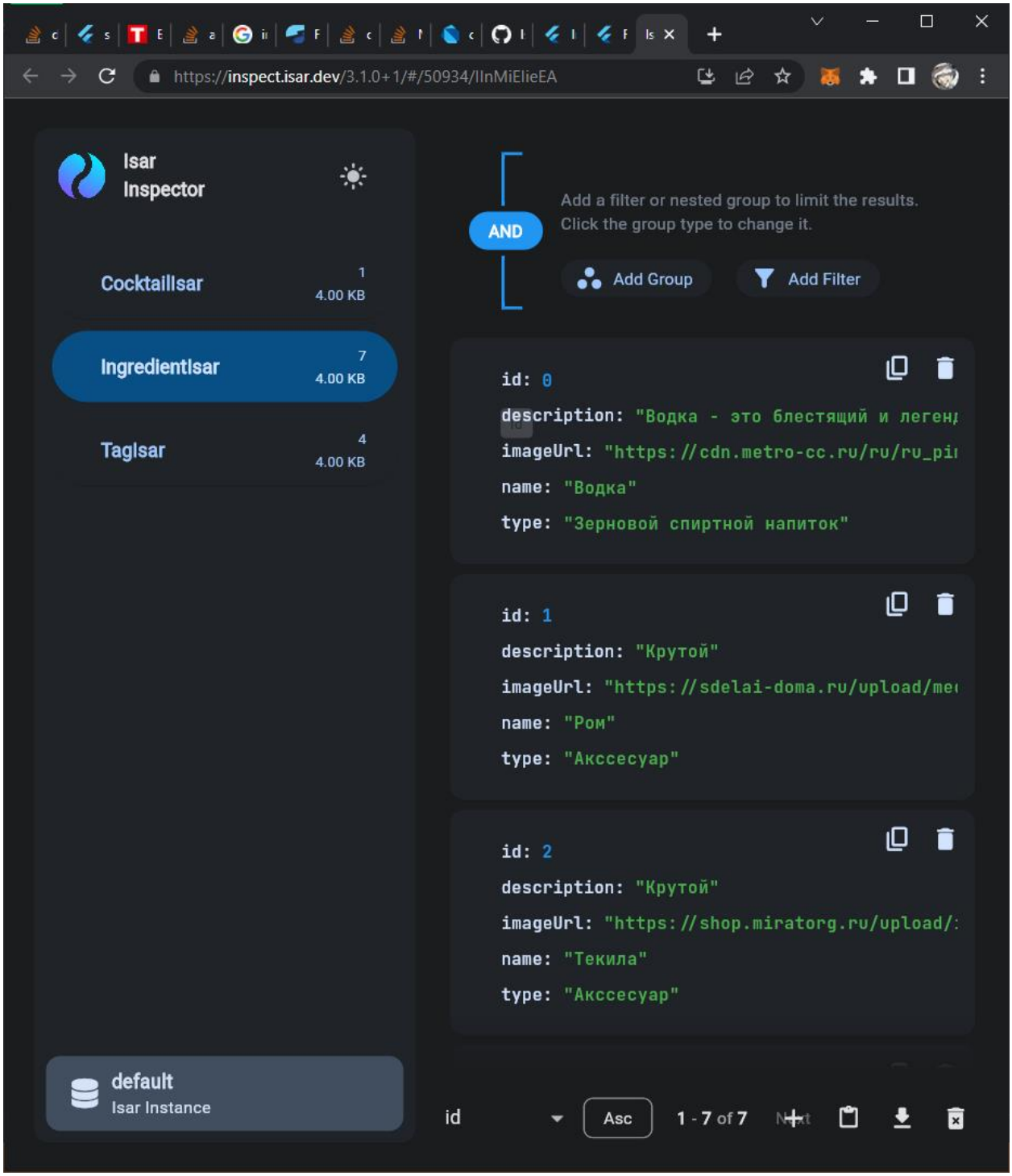

Рисунок 12 – Таблица “Ingredient”

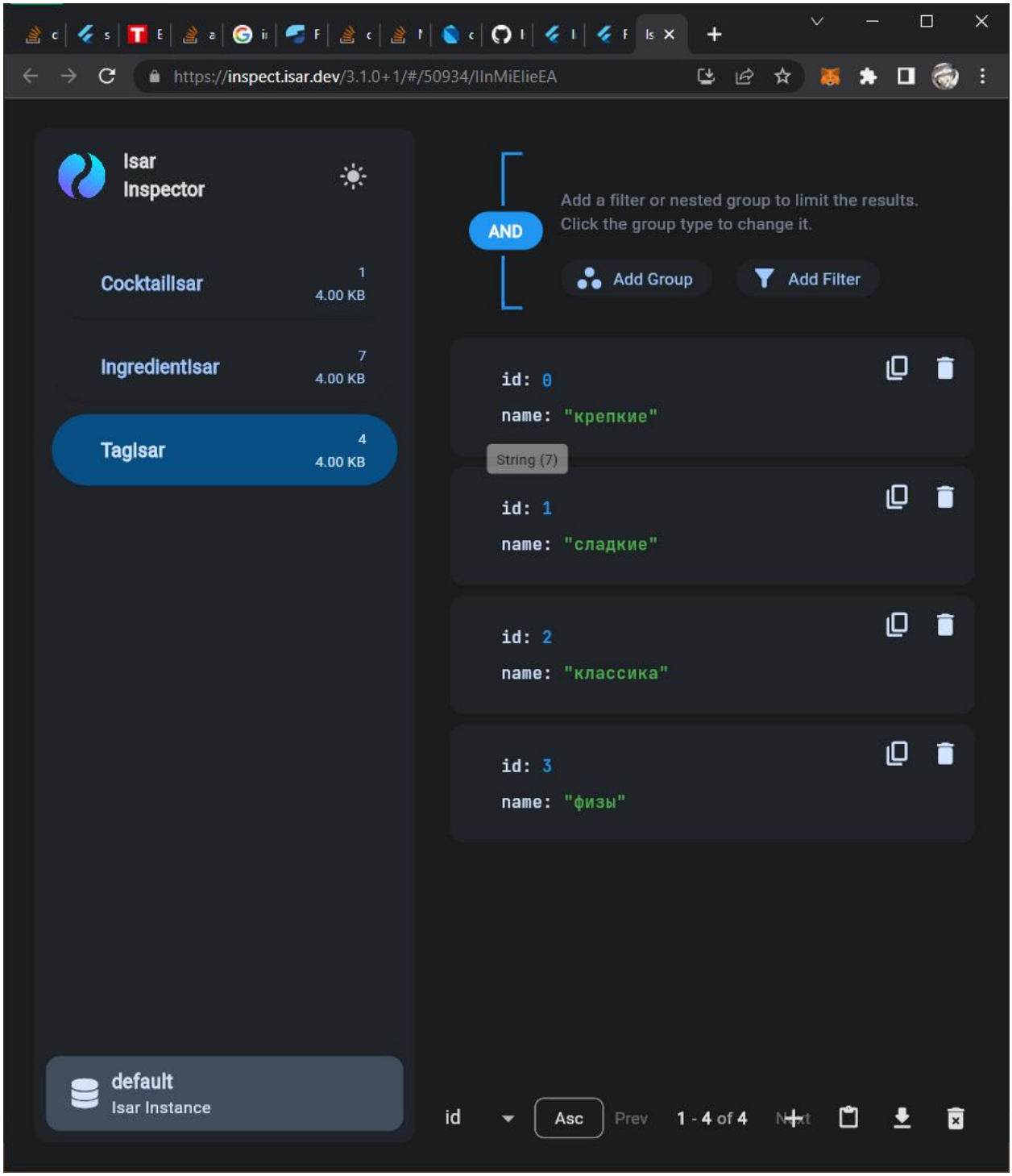

Рисунок 13 – Таблица “Tag”

Инициализацию базы данных было решено сделать при помощи функции импорта JSON снимка базы данных (см. рис. 14)

cocktail_gen - TagIsar.json

```json
[
    {
        "id": 0,
        "name": "крепкие"
    },
    {
        "id": 1,
        "name": "сладкие"
    },
    {
        "id": 2,
        "name": "классика"
    },
    {
        "id": 3,
        "name": "физы"
    }
]
```

Рисунок 14 – Пример JSON для таблицы “Tag”

После реализации базы данных началась реализация интерфейса и бизнес-логики приложения. В рамках разработки приложения были приняты следующие решения:

1) Методология разработки: Feature Driven Development

2) Архитектура проекта: Clean Architecture

Эти решения обоснованы тем, что это приложение является проектом маленького масштаба, поэтому разделение слоев приложения – отчетливо и просто. Также это позволяет гибко подстраиваться под график и ситуации.

Итоговый программный стек состоит из:

1 Framework: Flutter.

2 State Management: Riverpod [4].

3 DI: GetIt.

4 DB: Isar.

5 Navigation: AutoRoute.

6 VCS: Git с хостингом GitHub.

Первыми были реализованы главный экран, реализованный как PageView (см. рис. 15 – 18).

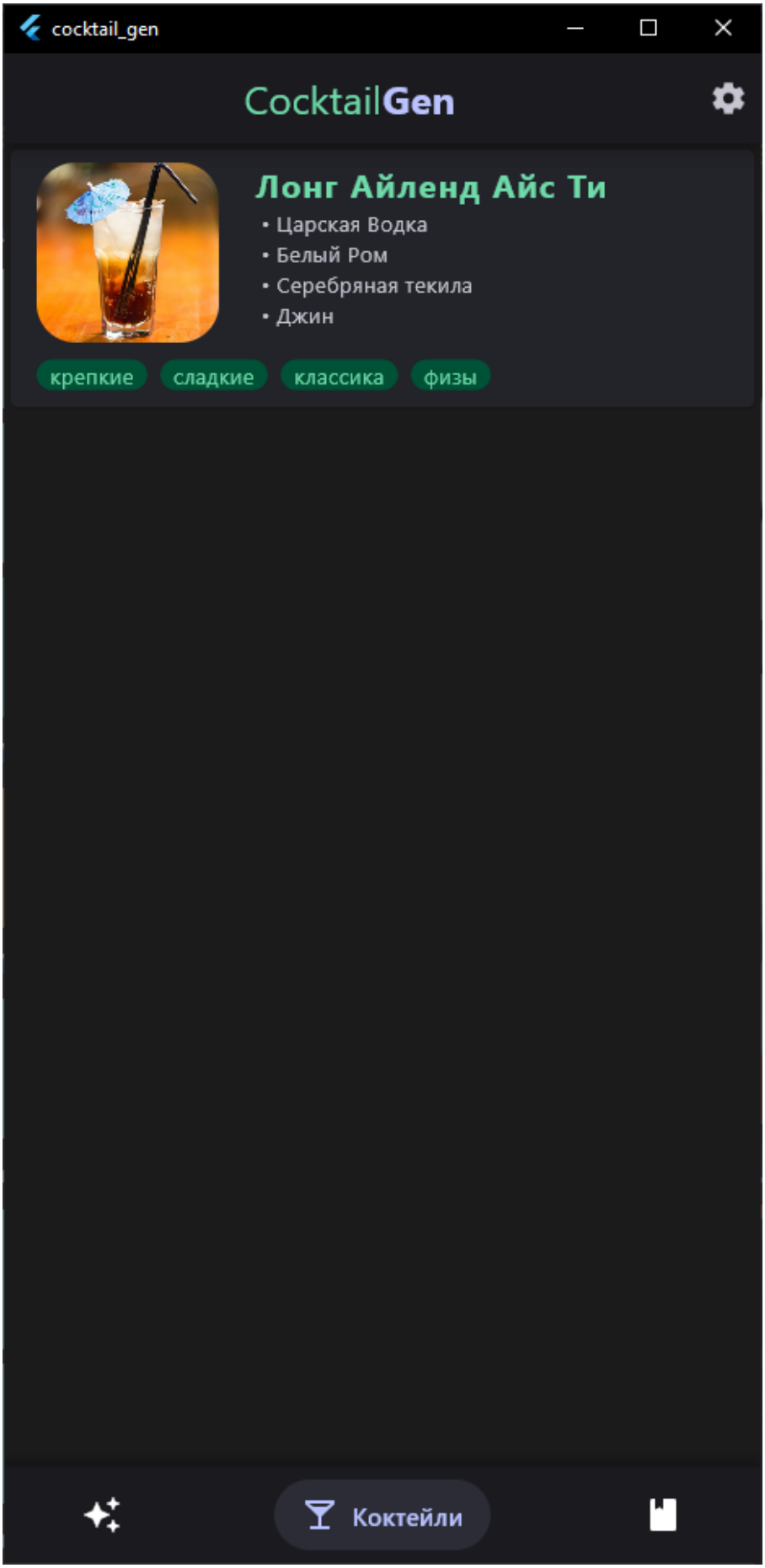

Рисунок 15 – Экран списка коктейлей

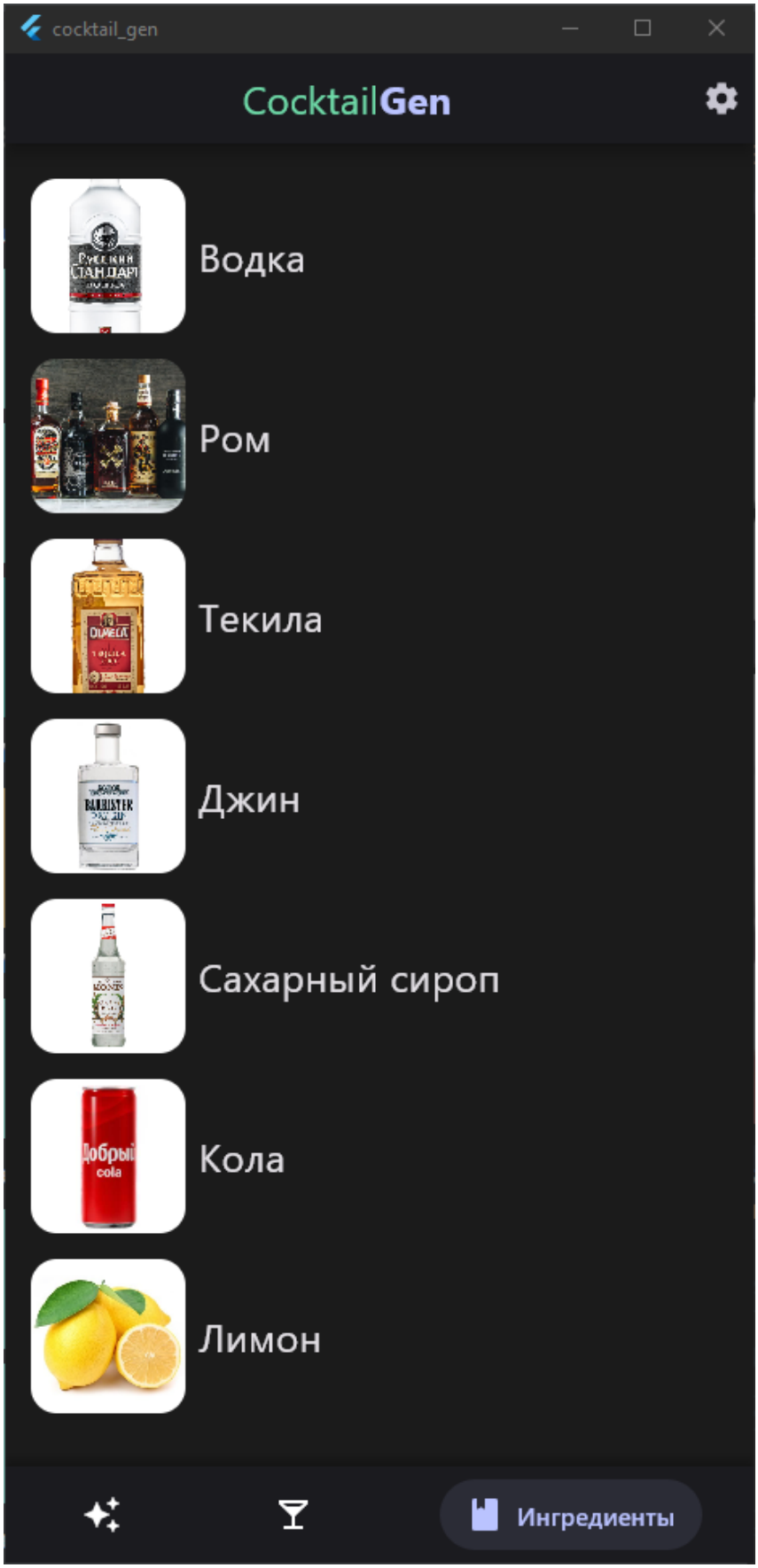

Рисунок 16 – Экран ингредиентов

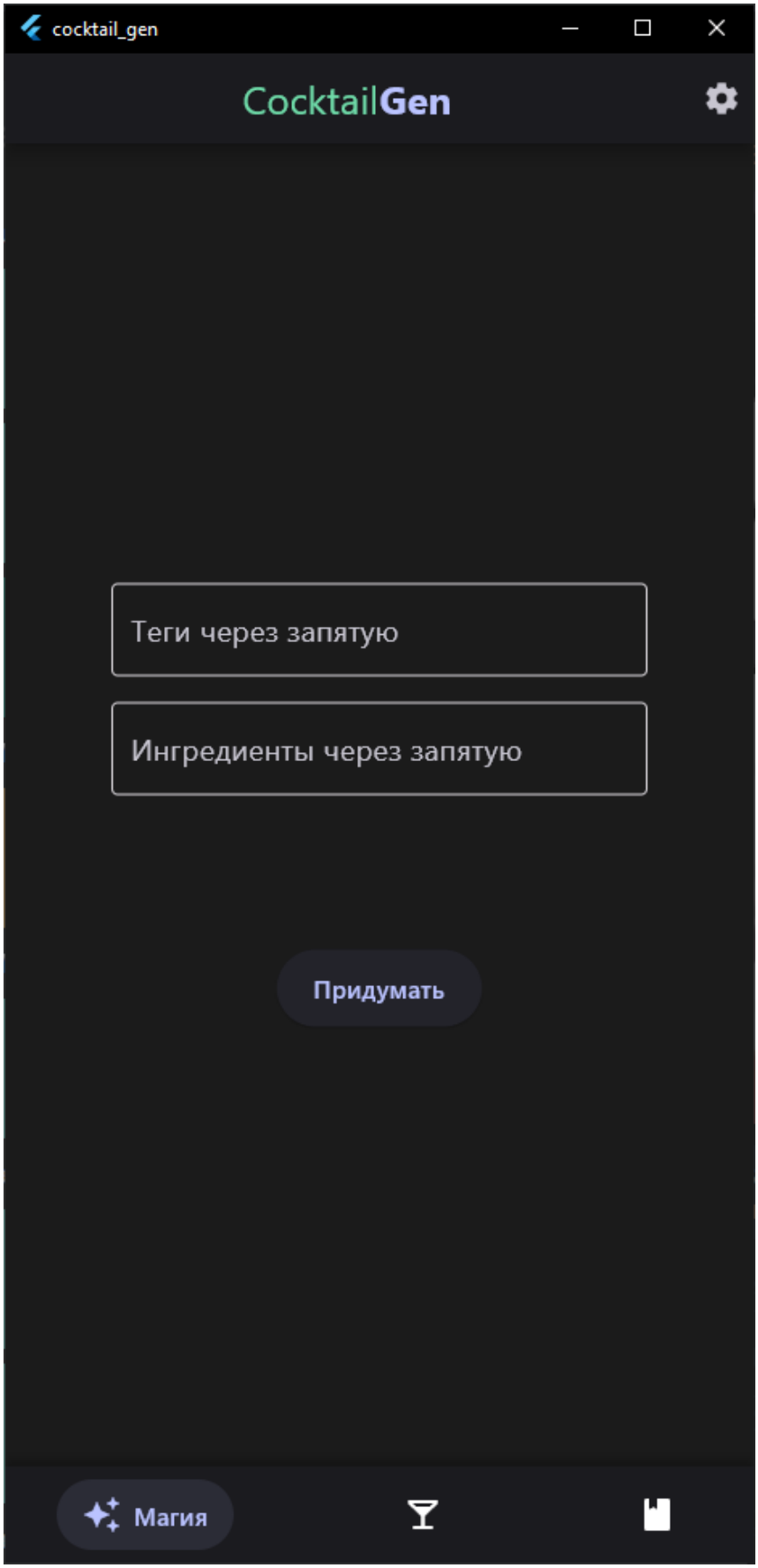

Рисунок 17 – Экран генерации

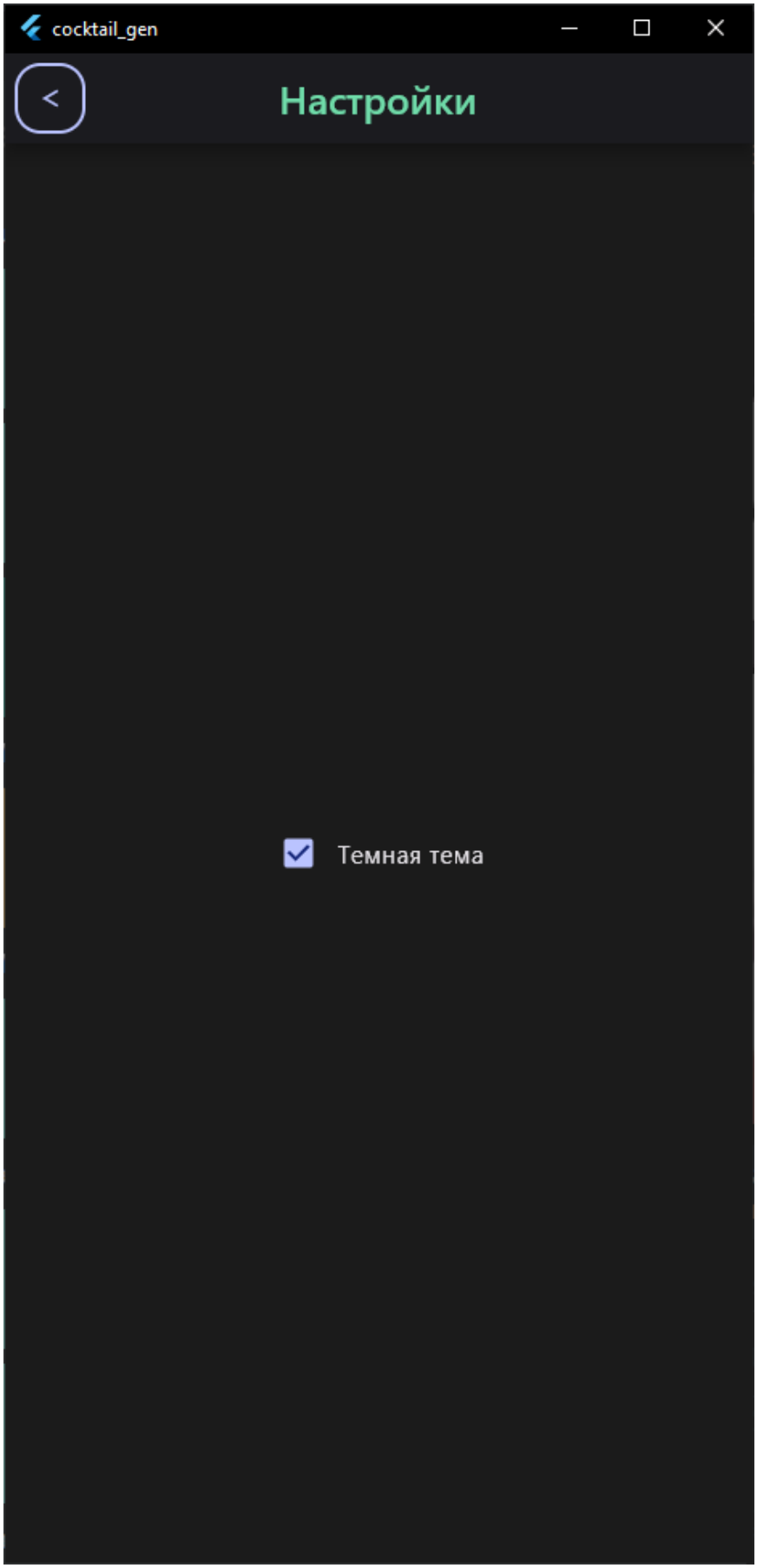

Рисунок 18 – Экран настроек

После этого были реализованы экраны просмотра коктейлей/ингредиентов (см. рис. 19 – 21).

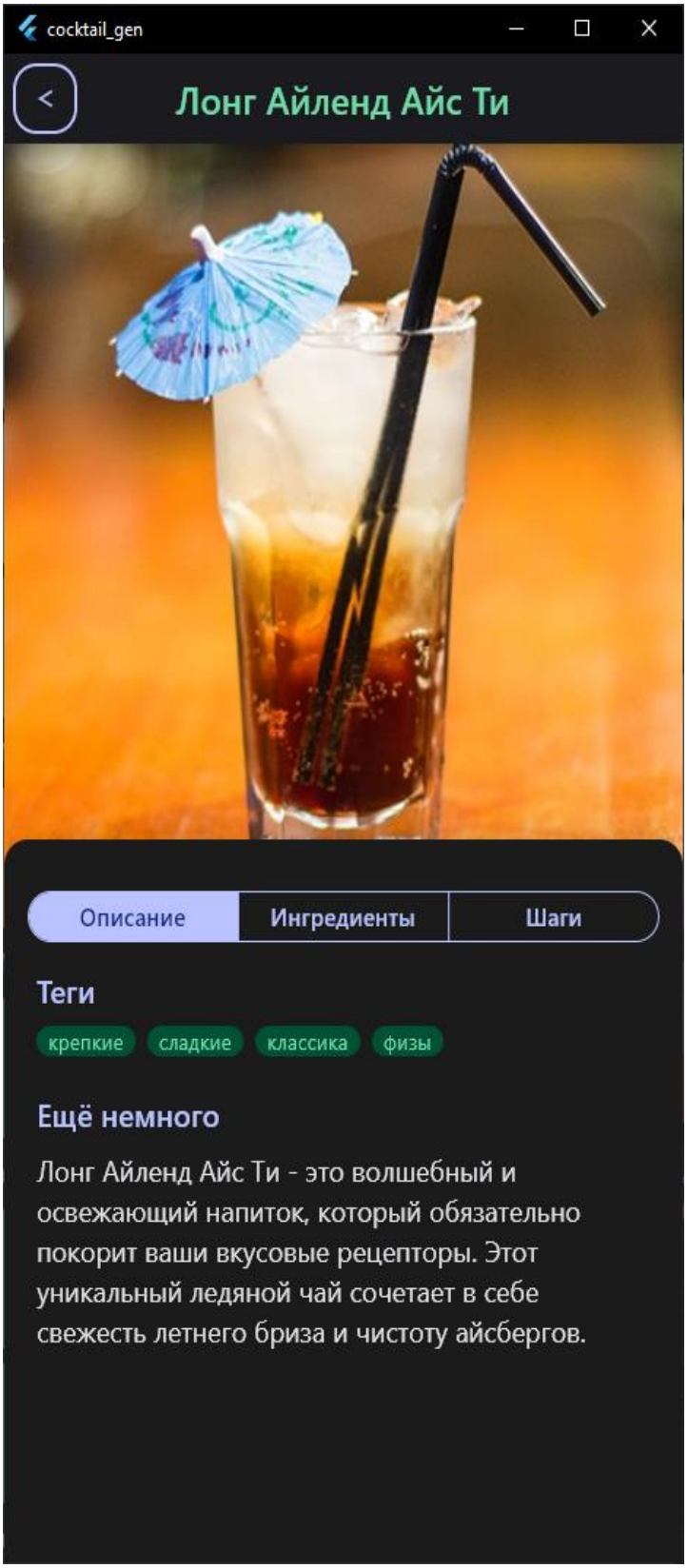

Рисунок 19 – Описание коктейля

Рисунок 20 – Шаги коктейля

Рисунок 21 – Описание ингредиента

В последнюю очередь была произведена интеграции приложения с сервисами OpenAi, а именно с API GPT 3.5. В результате мы получили экраны, отображающие сгенерированные рецепты (см. рис. 22 – 25) [5].

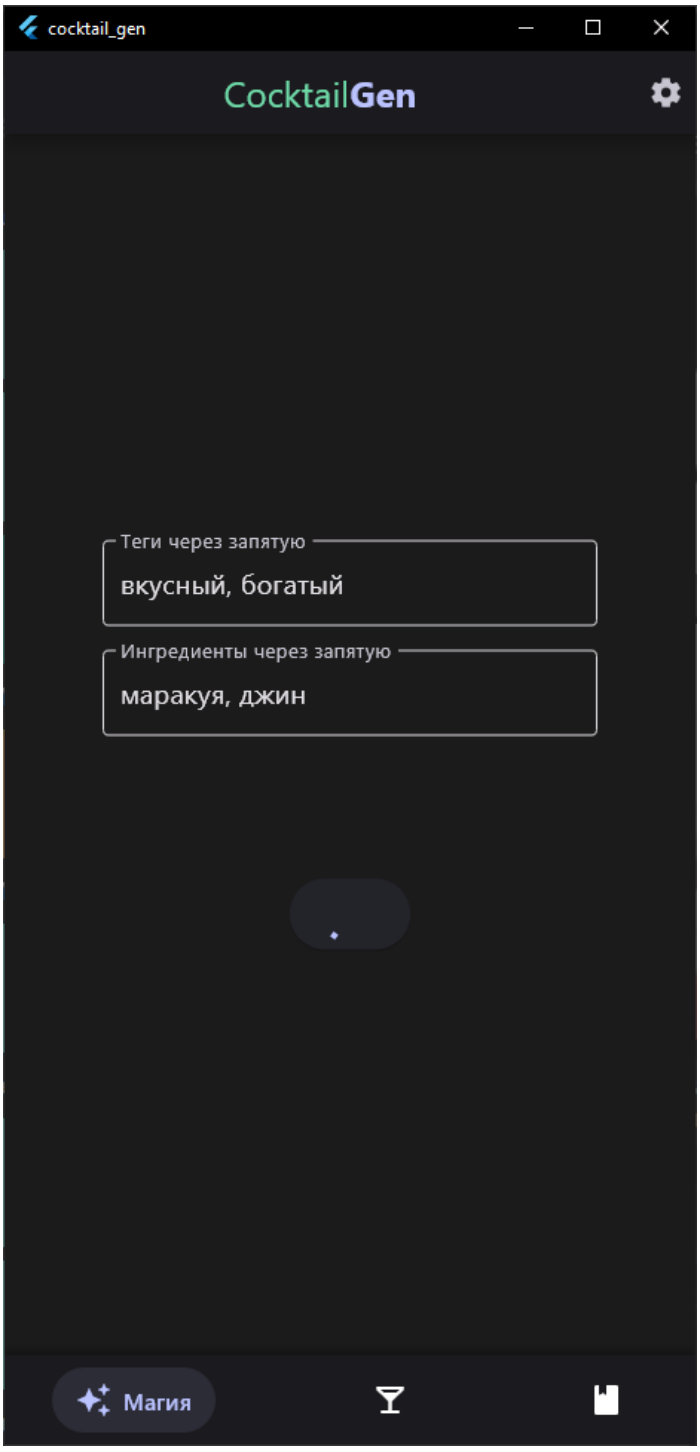

Рисунок 22 – Введенные параметры для генерации

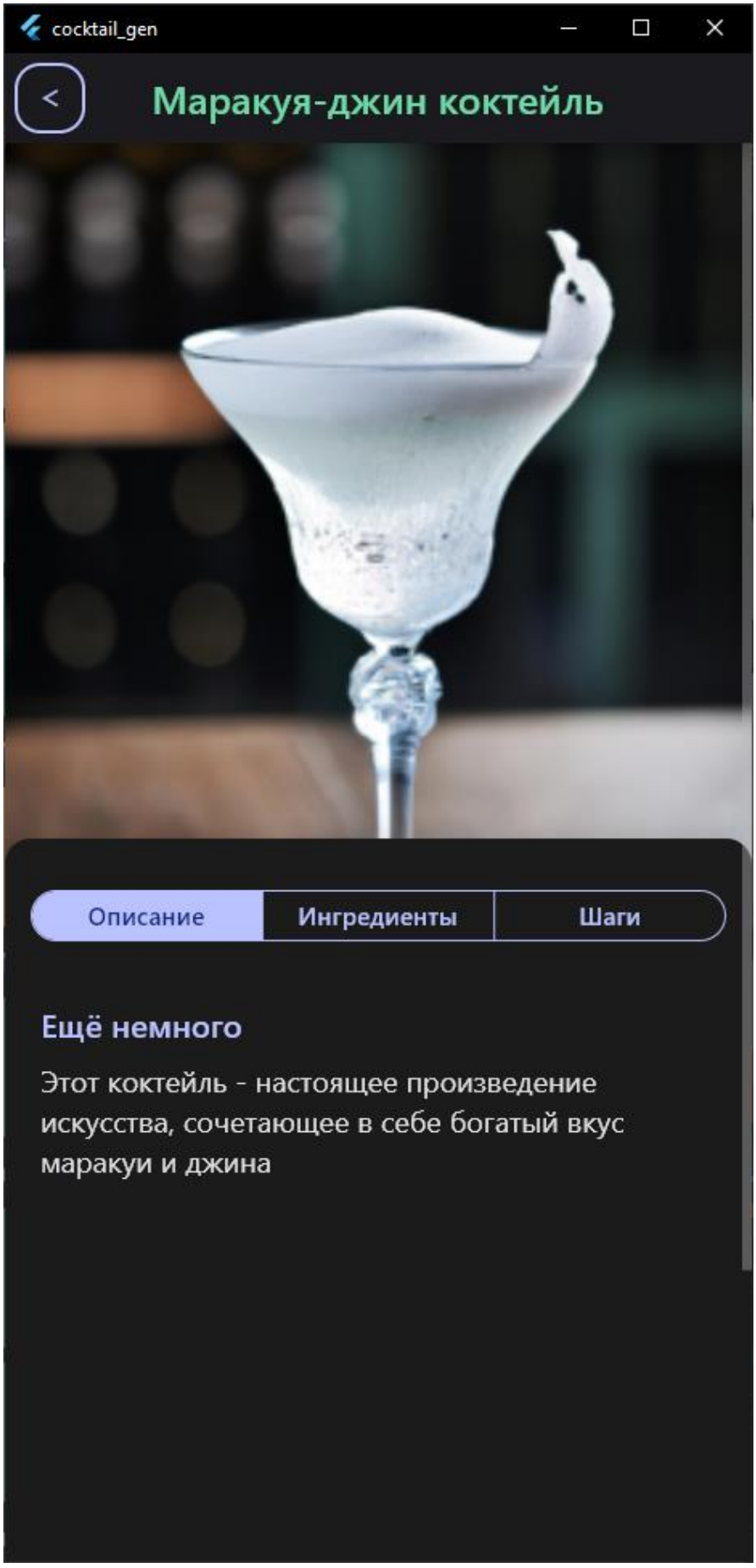

Рисунок 23 – Описание коктейля

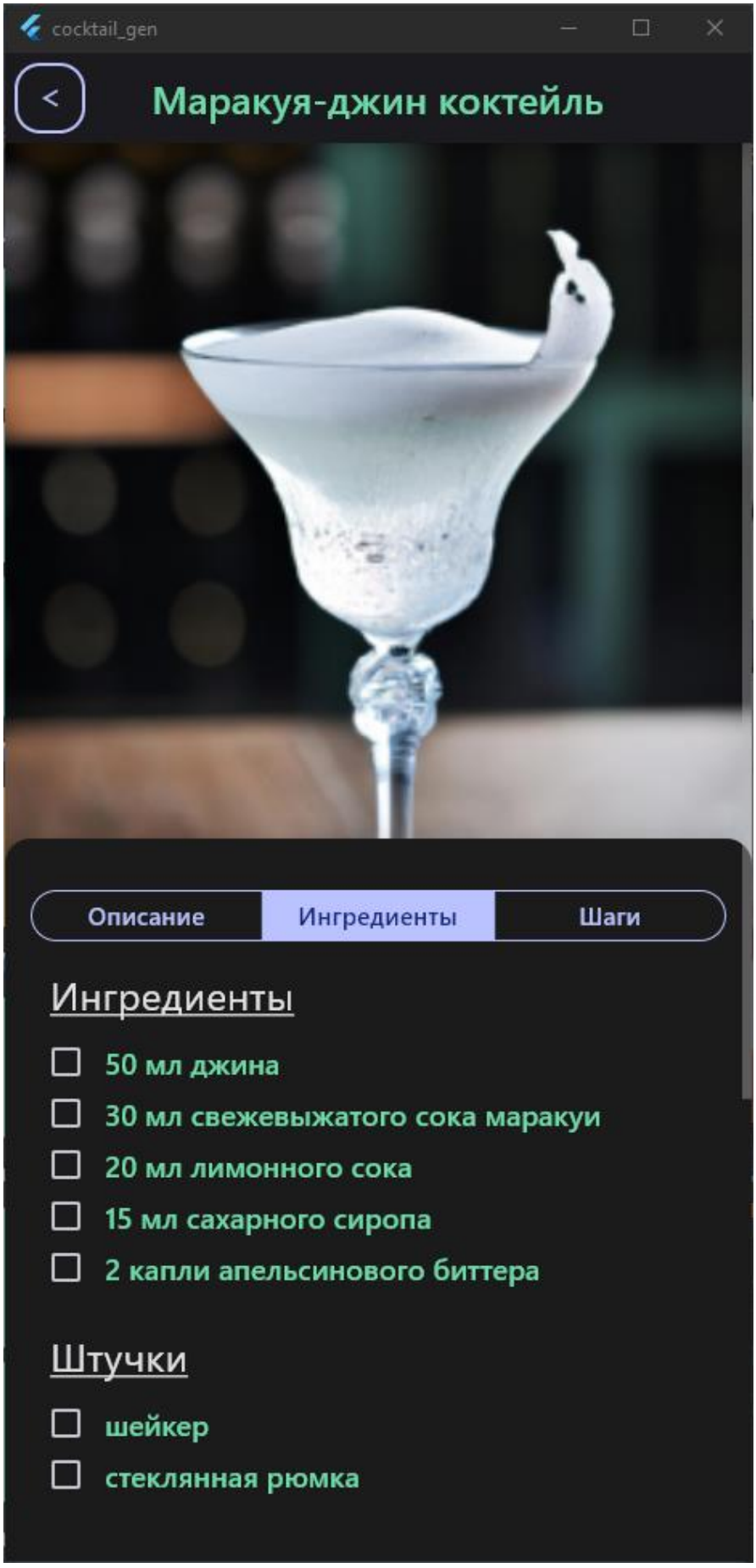

Рисунок 24 – Ингредиенты коктейля

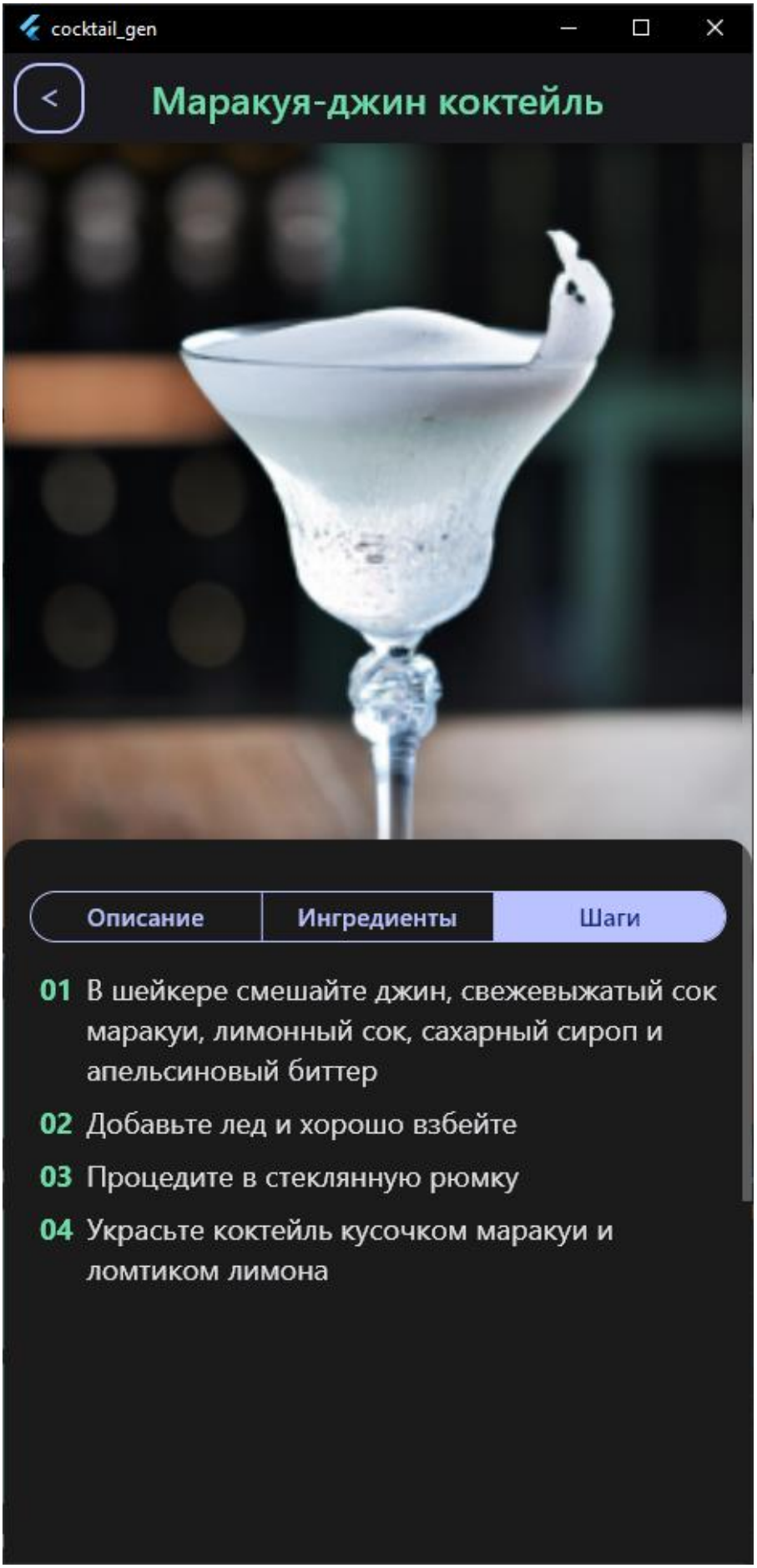

Рисунок 25 – Шаги коктейля

ЗАКЛЮЧЕНИЕ

В результате выполнения производственной практики была разработана информационная система доступа к информации о коктейлях и ингредиентах. Было создано приложение на языке программирования высокого уровня Dart с использованием фреймворка Flutter, а также СУБД Isar. Приложение просто в использовании и позволяет быстро и удобно получить нужную информацию. Использования базы данных для хранения и управления информацией позволило повысить эффективность как хранения информации, так и её поиска, и редактирования. Приложение также открыто для расширения и доработки. Например, можно подключить погрузку рецептов с облачной базы данных. В результате прохождения производственной практики было реализована ИС, исходный код который доступен на GitHub (URL: https://github.com/vladcto/cocktail-gen) или в Приложении А, а также был получен опыт в создании информационных систем, приложений и баз данных.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Flutter.dev: Фреймворк Flutter: сайт. – URL: https://flutter.dev/ (дата обращения: 21.07.2023)

2 Medium: MindMap при моделировании: сайт. – URL: https://medium.com/@jeanclmaia/a-mind-map-to-modelling-time-series-part-1-univariate-928dea36164e (дата обращения: 21.07.2023)

3 Isar DB: Библиотека Isar: сайт. – URL: https://isar.dev/ (дата обращения: 21.07.2023)

4 Riverpod.dev: Документация Riverpod: сайт. – URL: https://riverpod.dev/ (дата обращения: 21.07.2023)

5 OpenAi: Документация GPT 3.5: сайт. – URL: https://platform.openai.com/docs/introduction (дата обращения: 21.07.2023)

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД

main.dart

```
import 'dart:ui';

import 'package:cocktail_gen/app/di.dart';
import 'package:cocktail_gen/app/navigation/router.dart';
import 'package:cocktail_gen/app/provider.dart';
import 'package:cocktail_gen/interfaces/db_repository.dart';
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';

void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  setupDependencies();
  await getIt<RecipeRepository>().init();
  runApp(const MainApp());
}

final appRouter = AppRouter();

class MainApp extends StatelessWidget {
  const MainApp({super.key});

  @override
  Widget build(BuildContext context) {
    return ProviderScope(
      child: Consumer(
        builder: (_, ref, child) {
```

```
        return MaterialApp.router(
          debugShowCheckedModeBanner: false,
          theme: ref.watch(themeProvider),
          scrollBehavior: ScrollConfiguration.of(context).copyWith(
            dragDevices: {
              PointerDeviceKind.touch,
              PointerDeviceKind.mouse,
            },
          ),
          routerConfig: appRouter.config(),
        );
      },
    ),
  );
 }
}
```

di.dart

```
import 'package:cocktail_gen/data/api/gpt/dto/gpt_client.dart';
import 'package:cocktail_gen/data/repos/isar/controllers/isar_db.dart';
import 'package:cocktail_gen/interfaces/db_repository.dart';
import 'package:get_it/get_it.dart';

import 'keys.dart';

final getIt = GetIt.instance;

void setupDependencies() {
```

```
  getIt.registerSingleton<GptClient>(GptClient(token: Keys.gptKey));
  getIt.registerSingleton<RecipeRepository>(CocktailIsarDb());
}
```

provider.dart

```
import 'package:flutter/material.dart';
import 'package:riverpod/riverpod.dart';

import 'constants/app_theme.dart';

final themeProvider = StateProvider<ThemeData>(
  (ref) => AppTheme.dark,
);
```

app_font_size.dart

```
sealed class AppFontSize {
  static const double small = 12;
  static const double medium = 16;
  static const double header = 18;
  static const double title = 22;
}
```

app_paddings.dart

```
sealed class AppPaddings {
  static const double small = 8;
```

```
  static const double medium = 16;
  static const double large = 22;
  static const double veryLarge = 32;
}
```

app_radius.dart

```
import 'package:flutter/material.dart';

sealed class AppRadius {
  static BorderRadius small = BorderRadius.circular(4);
  static BorderRadius standard = BorderRadius.circular(16);
  static BorderRadius large = BorderRadius.circular(24);
}
```

app_shadows.dart

```
import 'package:flutter/material.dart';

sealed class AppShadows {
  static const Color shadowColor = Colors.black26;

  static const BoxShadow bottomTop = BoxShadow(
    color: shadowColor,
    blurRadius: 4,
    offset: Offset(0, -4),
  );
```

```
  static const BoxShadow topBottom = BoxShadow(
    color: shadowColor,
    blurRadius: 4,
    offset: Offset(0, 4),
  );
}
```

app_theme.dart

```
import 'package:flutter/material.dart';

sealed class AppTheme {
  // TODO: Убрать повторения цветов.
  static ThemeData dark = ThemeData(
    useMaterial3: true,
    brightness: Brightness.dark,
    primaryColor: const Color.fromRGBO(186, 195, 255, 1),
    colorScheme: const ColorScheme.dark(
      // Primary.
      primary: Color.fromRGBO(186, 195, 255, 1),
      onPrimary: Color.fromRGBO(22, 39, 122, 1),
      primaryContainer: Color.fromRGBO(48, 63, 145, 1),
      onPrimaryContainer: Color.fromRGBO(222, 224, 255, 1),
      // Secondary
      secondary: Color.fromRGBO(111, 219, 169, 1),
      secondaryContainer: Color.fromRGBO(0, 82, 54, 1),
      // Background
      background: Color.fromRGBO(27, 27, 27, 1),
      onBackground: Color.fromRGBO(228, 225, 230, 1),
```

```
    // Surface
    surface: Color.fromRGBO(27, 27, 31, 1),
    onSurface: Color.fromRGBO(228, 225, 230, 1),
    surfaceVariant: Color.fromRGBO(35, 36, 42, 1),
    onSurfaceVariant: Color.fromRGBO(199, 197, 208, 1),
  ),
  appBarTheme: const AppBarTheme(
    shadowColor: Colors.black,
    elevation: 8,
    surfaceTintColor: Color.fromRGBO(27, 27, 31, 1),
  ),
);

static ThemeData light = ThemeData(
  useMaterial3: true,
  brightness: Brightness.light,
  primaryColor: const Color.fromRGBO(73, 88, 171, 1),
  cardColor: const Color.fromRGBO(236, 235, 239, 1),
  colorScheme: const ColorScheme.light(
    // Primary.
    primary: Color.fromRGBO(73, 88, 171, 1),
    onPrimary: Color.fromRGBO(255, 255, 255, 1),
    primaryContainer: Color.fromRGBO(222, 224, 255, 1),
    onPrimaryContainer: Color.fromRGBO(0, 16, 92, 1),
    // Secondary
    secondary: Color.fromRGBO(0, 108, 73, 1),
    secondaryContainer: Color.fromRGBO(114, 219, 170, 1),
    // Background
    background: Color.fromRGBO(254, 251, 255, 1),
    onBackground: Color.fromRGBO(27, 27, 31, 1),
```

```
      // Surface
      surface: Color.fromRGBO(235, 231, 235, 1),
      onSurface: Color.fromRGBO(27, 27, 31, 1),
      surfaceVariant: Color.fromRGBO(236, 235, 239, 1),
      onSurfaceVariant: Color.fromRGBO(70, 70, 79, 1),
    ),
    appBarTheme: const AppBarTheme(
      shadowColor: Colors.black,
      elevation: 8,
      surfaceTintColor: Color.fromRGBO(236, 235, 239, 1),
    ),
  );
}
```

navigation_url.dart

```
sealed class NavigationUrl {
  static const String cocktail = "cocktail";
  static const String generate = "generate";
  static const String ingredients = "ingredients";
}
```

router.dart

```
import 'package:auto_route/auto_route.dart';
import 'package:cocktail_gen/app/navigation/navigation_url.dart';
import 'router.gr.dart';
```

```
@AutoRouterConfig(replaceInRouteName: 'Page,Route')
class AppRouter extends $AppRouter {
  static const cocktailParam = "cocktailId";
  static const ingredientParam = "ingredientId";

  @override
  List<AutoRoute> get routes => [
        AutoRoute(
          page: IngredientPreviewRoute.page,
          path: "/cocktail/1/*",
        ),
        AutoRoute(
          page: MainRoute.page,
          initial: true,
          path: "/",
          children: [
            AutoRoute(
              page: CocktailsRoute.page,
              path: NavigationUrl.cocktail,
            ),
            AutoRoute(
              page: GeneratorRoute.page,
              path: NavigationUrl.generate,
            ),
            AutoRoute(
              page: IngredientsRoute.page,
              path: NavigationUrl.ingredients,
            ),
          ],
        ),
```

```
        AutoRoute(
          page: CocktailPreviewRoute.page,
          path: "/${NavigationUrl.cocktail}/:$cocktailParam",
        ),
        AutoRoute(
          page: IngredientPreviewRoute.page,
          path: "/${NavigationUrl.ingredients}/:$ingredientParam",
        ),
        RedirectRoute(path: "*", redirectTo: "/${NavigationUrl.cocktail}"),
      ];
}
```

cocktails_page.dart

```
import 'package:auto_route/annotations.dart';
import 'package:cocktail_gen/app/widgets/cocktail_card.dart';
import 'package:cocktail_gen/interfaces/db_repository.dart';
import 'package:flutter/material.dart';
import 'package:get_it/get_it.dart';

/// Страница, отображающая список всех коктейлей.
@RoutePage()
class CocktailsPage extends StatelessWidget {
  const CocktailsPage({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return ListView(
      scrollDirection: Axis.vertical,
```

```
            children: [
              for (final cocktail in  GetIt.I<RecipeRepository>().fetchCocktails())
                SizedBox(
                  height: 168,
                  child: CocktailCard(cocktail: cocktail),
                ),
            ],
          );
        }
      }
```

cocktail_preview_page.dart

```
import 'package:auto_route/auto_route.dart';
import 'package:cocktail_gen/app/di.dart';
import 'package:cocktail_gen/app/navigation/router.dart';
import
'package:cocktail_gen/app/widgets/cocktail_preview/cocktail_preview_layout.dart'
;
import 'package:cocktail_gen/interfaces/db_repository.dart';
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';

@RoutePage()
class CocktailPreviewPage extends ConsumerWidget {
  final int cocktailId;
  const CocktailPreviewPage(
    @PathParam(AppRouter.cocktailParam) this.cocktailId, {
    Key? key,
```

```
  }) : super(key: key);

  @override
  Widget build(BuildContext context, WidgetRef ref) {
    return CocktailPreviewLayout(
      cocktail: getIt<RecipeRepository>().getCocktailById(cocktailId)!,
    );
  }
}
```

generated_cocktail_preview.dart

```
import
'package:cocktail_gen/app/widgets/cocktail_preview/cocktail_preview_layout.dart'
;
import 'package:cocktail_gen/domain/entities/cocktail.dart';
import 'package:flutter/material.dart';

class GeneratedCocktailPreview extends StatelessWidget {
  final Cocktail cocktail;

  const GeneratedCocktailPreview({Key? key, required this.cocktail})
      : super(key: key);

  @override
  Widget build(BuildContext context) {
    return CocktailPreviewLayout(
      cocktail: cocktail,
    );
```

```
  }
}
```

generator_page.dart

```
import 'dart:async';

import 'package:auto_route/auto_route.dart';
import 'package:cocktail_gen/app/constants/app_paddings.dart';
import 'package:cocktail_gen/app/pages/generated_cocktail_preview.dart';
import 'package:cocktail_gen/data/api/gpt/dto/gpt_client.dart';
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';
import 'package:get_it/get_it.dart';

final _tagsProvider = StateProvider<String>((_) => "");
final _ingredientsProvider = StateProvider<String>((_) => "");

final _loadingProvider = StateProvider((ref) => false);

@RoutePage()
class GeneratorPage extends ConsumerWidget {
  const GeneratorPage({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context, WidgetRef ref) {
    final router = context.router;

    return Padding(
```

```
            padding: const EdgeInsets.symmetric(horizontal: AppPaddings.large *
3),
            child: Center(
              child: Column(
                crossAxisAlignment: CrossAxisAlignment.center,
                mainAxisAlignment: MainAxisAlignment.center,
                children: [
                  TextField(
                    decoration: const InputDecoration(
                      label: Text("Теги через запятую"),
                      border: OutlineInputBorder(),
                    ),
                    onChanged: (val) => ref.read(_tagsProvider.notifier).state = val,
                  ),
                  const SizedBox(height: AppPaddings.medium),
                  TextField(
                    decoration: const InputDecoration(
                      label: Text("Ингредиенты через запятую"),
                      border: OutlineInputBorder(),
                    ),
                    onChanged: (val) =>
                        ref.read(_ingredientsProvider.notifier).state = val,
                  ),
                  const SizedBox(height: AppPaddings.veryLarge * 3),
                  SizedBox(
                    height: 48,
                    child: ElevatedButton(
                      onPressed: () {
                        if (ref.read(_loadingProvider)) return;
                        final tags = ref.read(_tagsProvider);
```

```
              final ingredients = ref.read(_ingredientsProvider);
              _generateRecipe(tags, ingredients, router, ref);
            },
            child: ref.watch(_loadingProvider)
                ? const CircularProgressIndicator()
                : const Text("Придумать"),
          ),
        ),
      ],
    ),
  ),
 );
}

Future<void> _generateRecipe(
  String tags,
  String ingredients,
  StackRouter router,
  WidgetRef ref,
) async {
  ref.read(_loadingProvider.notifier).state = true;
  await GetIt.I<GptClient>()
      .generateRecipe(
        tags,
        ingredients,
      )
      .then(
        (e) => router.pushWidget(
          GeneratedCocktailPreview(cocktail: e),
        ),
```

```
        )
        .onError((error, stackTrace) {
          return null;
        });
    ref.read(_loadingProvider.notifier).state = false;
  }
}
```

ingredients_page.dart

```
import 'package:auto_route/annotations.dart';
import 'package:cocktail_gen/app/constants/app_paddings.dart';
import 'package:cocktail_gen/app/widgets/ingredient_tile.dart';
import 'package:cocktail_gen/interfaces/db_repository.dart';
import 'package:flutter/material.dart';
import 'package:get_it/get_it.dart';

/// Страница, отображающая список всех ингредиентов.
@RoutePage()
class IngredientsPage extends StatelessWidget {
  static const double itemsPadding = AppPaddings.medium;

  const IngredientsPage({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return ListView(
      padding: const EdgeInsets.symmetric(
        // Ставим отступ в [large] учитывая отступ вокруг детей.
```

```
        vertical: AppPaddings.large - itemsPadding / 2,
        horizontal: AppPaddings.medium,
      ),
      children: [
        for (final ingredient in GetIt.I<RecipeRepository>().fetchIngredients())
          Padding(
            padding: const EdgeInsets.symmetric(vertical: itemsPadding / 2),
            child: SizedBox(
              height: 96,
              child: IngredientTile(ingredient: ingredient),
            ),
          ),
      ],
    );
  }
}
```

ingredient_preview.dart

```
import 'package:auto_route/auto_route.dart';
import 'package:cocktail_gen/app/constants/app_font_size.dart';
import 'package:cocktail_gen/app/constants/app_paddings.dart';
import 'package:cocktail_gen/app/di.dart';
import 'package:cocktail_gen/app/navigation/router.dart';
import 'package:cocktail_gen/app/widgets/additional_layout_info.dart';
import 'package:cocktail_gen/app/widgets/ingredient_tile.dart';
import 'package:cocktail_gen/interfaces/db_repository.dart';
import 'package:flutter/material.dart';
```

```
/// Страница, отображающая информация о ингредиенте.
@RoutePage()
class IngredientPreviewPage extends StatelessWidget {
  static const double imagePadding = 32;
  static const fontWeight = FontWeight.w500;

  final int ingredientId;

  const IngredientPreviewPage(
    @PathParam(AppRouter.ingredientParam) this.ingredientId, {
    Key? key,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    final ingredient = getIt<RecipeRepository>().getIngredientById(
      ingredientId,
    )!;
    final colorScheme = Theme.of(context).colorScheme;

    return AdditionalLayoutInfo(
      appBarText: ingredient.name,
      imageUrl: ingredient.imageUrl,
      heroTag: "${IngredientTile.heroPrefix}${ingredient.id}",
      child: Padding(
        padding: const EdgeInsets.symmetric(horizontal:
AppPaddings.veryLarge),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
```

```
            Text(
              ingredient.type,
              style: TextStyle(
                fontSize: AppFontSize.title,
                fontWeight: fontWeight,
                color: colorScheme.primary,
              ),
            ),
            const SizedBox(height: AppPaddings.medium),
            Text(
              ingredient.description,
              style: TextStyle(
                fontSize: AppFontSize.title,
                fontWeight: fontWeight,
                color: colorScheme.onSurface,
              ),
            ),
          ],
        ),
      ),
    );
  }
}
```

main_page.dart

```
import 'package:auto_route/auto_route.dart';
import 'package:cocktail_gen/app/constants/app_paddings.dart';
import 'package:cocktail_gen/app/constants/app_shadows.dart';
```

```
import 'package:cocktail_gen/app/navigation/router.gr.dart';
import 'package:cocktail_gen/app/pages/settings_page.dart';
import 'package:flutter/material.dart';
import
'package:material_design_icons_flutter/material_design_icons_flutter.dart';
import 'package:salomon_bottom_bar/salomon_bottom_bar.dart';

@RoutePage()
class MainPage extends StatelessWidget {
  static const generateText = "Магия";
  static const cocktailsText = "Коктейли";
  static const ingredientsText = "Ингредиенты";

  const MainPage({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    var colorScheme = Theme.of(context).colorScheme;

    return AutoTabsRouter.pageView(
      routes: const [
        GeneratorRoute(),
        CocktailsRoute(),
        IngredientsRoute(),
      ],
      builder: (ctx, child, controller) {
        final tabsRouter = AutoTabsRouter.of(ctx);

        return Scaffold(
          bottomNavigationBar: DecoratedBox(
```

```
  decoration: const BoxDecoration(
    boxShadow: [
      AppShadows.bottomTop,
    ],
  ),
  child: SalomonBottomBar(
    margin: const EdgeInsets.symmetric(
      vertical: AppPaddings.small,
      horizontal: AppPaddings.veryLarge,
    ),
    currentIndex: tabsRouter.activeIndex,
    backgroundColor: colorScheme.surface,
    items: [
      SalomonBottomBarItem(
        icon: Icon(MdiIcons.creation),
        title: const Text(generateText),
      ),
      SalomonBottomBarItem(
        icon: Icon(MdiIcons.glassCocktail),
        title: const Text(cocktailsText),
      ),
      SalomonBottomBarItem(
        icon: Icon(MdiIcons.book),
        title: const Text(ingredientsText),
      ),
    ],
    onTap: tabsRouter.setActiveIndex,
  ),
),
appBar: AppBar(
```

```
  actions: [
    IconButton(
      onPressed: () => context.router.pushWidget(
        const SettingsPage(),
      ),
      icon: const Icon(Icons.settings),
    ),
  ],
  title: Center(
    child: Row(
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
        Text(
          "Cocktail",
          style: TextStyle(
            color: colorScheme.secondary,
          ),
        ),
        Text(
          "Gen",
          style: TextStyle(
            color: colorScheme.primary,
            fontWeight: FontWeight.w700,
          ),
        ),
      ],
    ),
  ),
),
body: child,
```

```
        );
      },
    );
  }
}
```

settings_page.dart

```
import 'package:cocktail_gen/app/constants/app_paddings.dart';
import 'package:cocktail_gen/app/constants/app_theme.dart';
import 'package:cocktail_gen/app/provider.dart';
import 'package:cocktail_gen/app/widgets/sub_route_app_bar.dart';
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';

final _darkSelectedProvider = StateProvider((ref) => true);

class SettingsPage extends ConsumerWidget {
  const SettingsPage({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context, WidgetRef ref) {
    ref.listen(
      _darkSelectedProvider,
      (_, darkTheme) => ref.read(themeProvider.notifier).state =
          darkTheme ? AppTheme.dark : AppTheme.light,
    );

    return Scaffold(
```

```
      appBar: SubRouteAppBar(title: "Настройки"),
      body: Center(
        child: Row(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            Checkbox(
              value: ref.watch(_darkSelectedProvider),
              onChanged: (val) =>
                  ref.read(_darkSelectedProvider.notifier).state = val!,
            ),
            const SizedBox(width: AppPaddings.small),
            const Text("Темная тема")
          ],
        ),
      ),
    );
  }
}
```

url_cutter_extension.dart

```
extension UrlCutterExtension on String {
  /// Обрезает URL до последнего "/".
  String get prevUrl => substring(0, lastIndexOf("/"));
}
```

additional_layout_info.dart

```
import 'package:cocktail_gen/app/widgets/sub_route_app_bar.dart';
import 'package:cocktail_gen/app/widgets/theme_shimmer.dart';
import 'package:flutter/material.dart';

import '../constants/app_radius.dart';

/// Разметка для дочернего маршрута.
/// Отображает изображение [imageUrl] и прокручивающийся виджет [child].
class AdditionalLayoutInfo extends StatelessWidget {
  static const double imagePadding = 32;
  final String? heroTag;
  final String appBarText;
  final String imageUrl;
  final Widget child;

  const AdditionalLayoutInfo({
    Key? key,
    required this.appBarText,
    required this.imageUrl,
    required this.child,
    this.heroTag,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    final screenWidth = MediaQuery.of(context).size.width;
    final colorScheme = Theme.of(context).colorScheme;

    final image = Image.network(
```

```
  imageUrl,
  fit: BoxFit.cover,
  loadingBuilder: (_, child, event) {
    if (event == null) return child;
    return const ThemeShimmer();
  },
  errorBuilder: (_, __, ___) => const ThemeShimmer(),
);

return Scaffold(
  appBar: SubRouteAppBar(
    title: appBarText,
  ),
  body: Stack(
    children: [
      // Делаем изображение квадратным.
      Positioned(
        top: 0,
        left: 0,
        right: 0,
        height: screenWidth,
        child: heroTag != null
            ? Hero(
                tag: heroTag!,
                child: image,
              )
            : image,
      ),
      Positioned.fill(
        child: ListView(
```

```
            children: [
              // Чтобы подложка немного заехала на изображение.
              SizedBox(height: screenWidth - imagePadding),
              Container(
                decoration: BoxDecoration(
                    borderRadius: AppRadius.standard,
                    color: colorScheme.background,
                    boxShadow: const [
                      BoxShadow(
                        offset: Offset(0, -4),
                        blurRadius: 12,
                        color: Colors.black12,
                      )
                    ]),
                padding: EdgeInsets.only(
                  // Учитываем шапку контейнера.
                  top: imagePadding,
                  // Делаем список прокручивающимся.
                  bottom: screenWidth,
                ),
                child: child,
              ),
            ],
          ),
        )
      ],
    ),
  );
 }
}
```

cocktail_card.dart

```
import 'package:auto_route/auto_route.dart';
import 'package:cocktail_gen/app/constants/app_font_size.dart';
import 'package:cocktail_gen/app/constants/app_paddings.dart';
import 'package:cocktail_gen/app/constants/app_radius.dart';
import 'package:cocktail_gen/app/widgets/tag_chip.dart';
import 'package:cocktail_gen/app/widgets/theme_shimmer.dart';
import 'package:cocktail_gen/domain/entities/cocktail.dart';
import 'package:cocktail_gen/domain/entities/tag.dart';
import 'package:flutter/material.dart';

/// Карточка, отображающая краткую информацию о [Cocktail].
///
/// Отображает картинку, имя, 4 ингредиента и список тегов.
class CocktailCard extends StatelessWidget {
  static const heroPrefix = "cocktail-";
  final Cocktail cocktail;
  const CocktailCard({Key? key, required this.cocktail}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return GestureDetector(
      onTap: () => context.router.pushNamed(
        "${context.router.currentUrl}/${cocktail.id}",
      ),
      child: Card(
        shape: RoundedRectangleBorder(
```

```
  borderRadius: AppRadius.small,
),
child: Padding(
  padding: const EdgeInsets.symmetric(
    horizontal: AppPaddings.medium,
    vertical: AppPaddings.small,
  ),
  child: Column(
    children: [
      Expanded(
        child: Row(
          children: [
            // Картинка рецепта.
            Hero(
              tag: "$heroPrefix${cocktail.id}",
              child: AspectRatio(
                aspectRatio: 1,
                child: ClipRRect(
                  borderRadius: AppRadius.large,
                  child: Image.network(
                    cocktail.imageUrl,
                    fit: BoxFit.cover,
                    loadingBuilder: (_, child, event) {
                      if (event == null) return child;
                      return const ThemeShimmer();
                    },
                    errorBuilder: (_, __, ___) => const ThemeShimmer(),
                  ),
                ),
              ),
```

```
                ),
                const SizedBox(width: AppPaddings.large),
                Expanded(
                  child: _CocktailShortInfo(cocktail: cocktail),
                ),
              ],
            ),
          ),
          const SizedBox(height: AppPaddings.small),
          SizedBox(
            height: 24,
            child: _TagList(tags: cocktail.tags),
          )
        ],
      ),
    ),
  );
 }
}

/// Отображает имя и список ингредиентов коктейля.
class _CocktailShortInfo extends StatelessWidget {
 final Cocktail cocktail;

 const _CocktailShortInfo({required this.cocktail});

 @override
 Widget build(BuildContext context) {
  final colorScheme = Theme.of(context).colorScheme;
```

```
        final ingredientsText =
            cocktail.ingredients.map((e) => " • ${e.name}").join("\n");

        return Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            // Имя коктейля.
            Text(
              cocktail.name,
              style: TextStyle(
                letterSpacing: 1.2,
                color: colorScheme.secondary,
                fontSize: AppFontSize.header,
                fontWeight: FontWeight.bold,
              ),
            ),
            // Список ингредиентов коктейля.
            Expanded(
              // FIXME: Исправть обрывание текста. [elipsis] не обрезает по
строке.
              child: Text(
                ingredientsText,
                maxLines: 4,
                overflow: TextOverflow.ellipsis,
                style: TextStyle(
                  fontSize: AppFontSize.small,
                  color: colorScheme.onSurfaceVariant,
                ),
              ),
            ),
```

```
      ],
    );
  }
}

/// [ListView] отображающий [Tag] с помощью [TagChip].
class _TagList extends StatelessWidget {
  final List<Tag> tags;

  const _TagList({required this.tags});

  @override
  Widget build(BuildContext context) {
    return ScrollConfiguration(
      behavior: ScrollConfiguration.of(context).copyWith(
        scrollbars: false,
      ),
      child: ListView.separated(
        scrollDirection: Axis.horizontal,
        itemCount: tags.length,
        itemBuilder: (context, index) => Center(
          child: TagChip(
            tag: tags[index],
          ),
        ),
        separatorBuilder: (context, i) =>
            const SizedBox(width: AppPaddings.small),
      ),
    );
  }
```

```
}
```

ingredient_tile.dart

```
import 'package:auto_route/auto_route.dart';
import 'package:cocktail_gen/app/constants/app_font_size.dart';
import 'package:cocktail_gen/app/constants/app_paddings.dart';
import 'package:cocktail_gen/app/constants/app_radius.dart';
import 'package:cocktail_gen/app/widgets/theme_shimmer.dart';
import 'package:cocktail_gen/domain/entities/ingredient.dart';
import 'package:flutter/material.dart';

/// Карточка, которая отображает имя ингредиента и его картинку.
class IngredientTile extends StatelessWidget {
  static const heroPrefix = "ingredient-";
  final Ingredient ingredient;

  const IngredientTile({Key? key, required this.ingredient}) : super(key:
key);

  @override
  Widget build(BuildContext context) {
    final colorScheme = Theme.of(context).colorScheme;

    return Row(
      children: [
        GestureDetector(
          onTap: () {
            context.router.pushNamed(
```

```
      "${context.router.currentUrl}/${ingredient.id}",
    );
  },
  child: Hero(
    tag: "$heroPrefix${ingredient.id}",
    child: AspectRatio(
      aspectRatio: 1,
      child: ClipRRect(
        borderRadius: AppRadius.standard,
        child: Image.network(
          ingredient.imageUrl,
          fit: BoxFit.cover,
          loadingBuilder: (_, child, event) {
            if (event == null) return child;
            return const ThemeShimmer();
          },
          errorBuilder: (_, __, ___) => const ThemeShimmer(),
        ),
      ),
    ),
  ),
),
const SizedBox(width: AppPaddings.small),
Expanded(
  child: Text(
    ingredient.name,
    style: TextStyle(
      fontSize: AppFontSize.title,
      color: colorScheme.onSurface,
    ),
```

```
        ),
      ),
    ],
   );
  }
}
```

simple_checkbox.dart

```
import 'package:flutter/material.dart';

/// [Checkbox] которые можно переключать, но не отслеживать.
class SimpleCheckbox extends StatefulWidget {
  const SimpleCheckbox({Key? key}) : super(key: key);

  @override
  State<SimpleCheckbox> createState() => _SimpleCheckboxState();
}

class _SimpleCheckboxState extends State<SimpleCheckbox> {
  bool checked = false;

  @override
  Widget build(BuildContext context) {
    return Checkbox(
      value: checked,
      onChanged: (_) => setState(() => checked = !checked),
    );
  }
```

```
}
```

step_tile.dart

```
import 'package:cocktail_gen/app/constants/app_font_size.dart';
import 'package:cocktail_gen/app/constants/app_paddings.dart';
import 'package:flutter/material.dart';

class StepTile extends StatelessWidget {
  final int index;
  final String text;

  const StepTile({
    super.key,
    required this.index,
    required this.text,
  });

  @override
  Widget build(BuildContext context) {
    final colorScheme = Theme.of(context).colorScheme;

    return Row(
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
        Text(
          index.toString().padLeft(2, "0"),
          style: TextStyle(
```

```
            color: colorScheme.secondary,
            fontSize: AppFontSize.medium,
            fontWeight: FontWeight.w700,
          ),
        ),
        const SizedBox(width: AppPaddings.small),
        Flexible(
          fit: FlexFit.loose,
          child: Text(
            text,
            style: TextStyle(
              color: colorScheme.onSurface,
              fontWeight: FontWeight.w400,
              fontSize: AppFontSize.medium,
            ),
          ),
        ),
      ],
    );
  }
}
```

sub_route_app_bar.dart

```
import 'package:cocktail_gen/app/constants/app_font_size.dart';
import 'package:cocktail_gen/app/widgets/theme_back_button.dart';
import 'package:flutter/material.dart';

class SubRouteAppBar extends AppBar {
```

```
  SubRouteAppBar({
    Key? key,
    required String title,
  }) : super(
          key: key,
          centerTitle: true,
          leading: const ThemeBackButton(),
          title: Builder(
            builder: (context) {
              return Text(
                title,
                style: TextStyle(
                  fontSize: AppFontSize.title,
                  color: Theme.of(context).colorScheme.secondary,
                  fontWeight: FontWeight.w500,
                ),
              );
            },
          ),
        );
}
```

tag_chip.dart

```
import 'package:cocktail_gen/app/constants/app_font_size.dart';
import 'package:cocktail_gen/domain/entities/tag.dart';
import 'package:flutter/material.dart';

// Chip отображающий имя тега.
```

```
class TagChip extends StatelessWidget {
  final Tag tag;
  const TagChip({Key? key, required this.tag}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    final colorScheme = Theme.of(context).colorScheme;

    return DecoratedBox(
      decoration: ShapeDecoration(
        shape: const StadiumBorder(),
        color: colorScheme.secondaryContainer,
      ),
      child: Padding(
        padding: const EdgeInsets.symmetric(horizontal: 8),
        child: Text(
          tag.name,
          textAlign: TextAlign.center,
          style: TextStyle(
            color: colorScheme.secondary,
            fontSize: AppFontSize.small,
          ),
        ),
      ),
    );
  }
}
```

theme_back_button.dart

```
import 'package:auto_route/auto_route.dart';
import 'package:cocktail_gen/app/constants/app_radius.dart';
import 'package:cocktail_gen/app/utils/url_cutter_extension.dart';
import 'package:flutter/material.dart';
import
'package:material_design_icons_flutter/material_design_icons_flutter.dart';

/// Кнопка, которая возвращает на предыдущий стек навигации.
class ThemeBackButton extends StatelessWidget {
  static const double margin = 6;
  static const double iconSize = 16;
  static const double strokeWidth = 2;

  const ThemeBackButton({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    final colorScheme = Theme.of(context).colorScheme;

    return Padding(
      padding: const EdgeInsets.all(margin),
      child: OutlinedButton(
        style: OutlinedButton.styleFrom(
          padding: const EdgeInsets.all(0),
          shape: RoundedRectangleBorder(
            borderRadius: AppRadius.standard,
          ),
          side: BorderSide(
            width: strokeWidth,
```

```
            color: colorScheme.primary,
          ),
        ),
        onPressed: () {
          final router = context.router;
          if (router.stack.length == 1 && !router.canPop()) {
            // Сохраняем URL *ДО* того, как уберем текущую страницу.
            final prevUrl = router.currentUrl.prevUrl;
            // Чтобы не держать в стеке убранную страницу.
            router.removeLast();
            router.navigateNamed(prevUrl);
          } else {
            router.pop();
          }
        },
        child: Icon(
          MdiIcons.lessThan,
          size: iconSize,
        ),
      ),
    );
  }
}
```

theme_segmented_button.dart

```
import 'package:flutter/material.dart';
```

```
/// Виджет - стилизованный [SegmentedButton] с единственным выбором.
class ThemeSegmentedButton<T> extends StatefulWidget {
  final List<ButtonSegment<T>> segments;
  final T initial;
  final void Function(T) onSelected;

  const ThemeSegmentedButton({
    super.key,
    required this.segments,
    required this.onSelected,
    required this.initial,
  });

  @override
  State<ThemeSegmentedButton<T>> createState() =>
      _ThemeSegmentedButtonState<T>();
}

class _ThemeSegmentedButtonState<T> extends State<ThemeSegmentedButton<T>> {
  late T _selected;

  @override
  void initState() {
    _selected = widget.initial;
    super.initState();
  }

  @override
```

```
Widget build(BuildContext context) {
  var colorScheme = Theme.of(context).colorScheme;

  return SegmentedButton(
    style: ButtonStyle(
      backgroundColor: MaterialStateProperty.resolveWith<Color>(
        (states) => switch (states.firstOrNull) {
          MaterialState.selected => colorScheme.primary,
          _ => Colors.transparent,
        },
      ),
      side: MaterialStateProperty.all(
        BorderSide(color: colorScheme.primary),
      ),
      padding: MaterialStateProperty.all(const EdgeInsets.all(0)),
      foregroundColor: MaterialStateProperty.resolveWith(
        (state) => state.firstOrNull == MaterialState.selected
            ? colorScheme.onPrimary
            : colorScheme.primary,
      ),
    ),
    showSelectedIcon: false,
    segments: widget.segments,
    selected: {_selected},
    onSelectionChanged: (selected) {
      setState(() => _selected = selected.first);
      widget.onSelected(_selected);
    },
  );
}
```

```
}
```

theme_shimmer.dart

```
import 'package:flutter/material.dart';
import 'package:shimmer/shimmer.dart';

class ThemeShimmer extends StatelessWidget {
  const ThemeShimmer({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    final colorScheme = Theme.of(context).colorScheme;

    return Shimmer.fromColors(
      baseColor: colorScheme.background,
      highlightColor: colorScheme.surfaceVariant,
      child: const ColoredBox(color: Colors.black),
    );
  }
}
```

cocktail_preview_layout.dart

```
import 'package:cocktail_gen/app/constants/app_paddings.dart';
import 'package:cocktail_gen/app/widgets/additional_layout_info.dart';
import 'package:cocktail_gen/app/widgets/cocktail_card.dart';
```

```
import
'package:cocktail_gen/app/widgets/cocktail_preview/description_preview.dart';
import
'package:cocktail_gen/app/widgets/cocktail_preview/ingredients_preview.dart';
import
'package:cocktail_gen/app/widgets/cocktail_preview/steps_preview.dart';
import 'package:cocktail_gen/app/widgets/theme_segmented_button.dart';
import 'package:cocktail_gen/domain/entities/cocktail.dart';
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';

enum PreviewType {
  description,
  ingredients,
  steps,
}

final _previewTypeProvider = StateProvider(
  (ref) => PreviewType.description,
);

/// Страница, отображающая информацию о [Cocktail].
class CocktailPreviewLayout extends ConsumerWidget {
  final Cocktail cocktail;

  const CocktailPreviewLayout({
    Key? key,
    required this.cocktail,
  }) : super(key: key);
```

```
        @override
        Widget build(BuildContext context, WidgetRef ref) {
          return AdditionalLayoutInfo(
            heroTag: "${CocktailCard.heroPrefix}${cocktail.id}",
            appBarText: cocktail.name,
            imageUrl: cocktail.imageUrl,
            child: Column(
              crossAxisAlignment: CrossAxisAlignment.stretch,
              children: [
                Padding(
                  padding: const EdgeInsets.symmetric(horizontal:
AppPaddings.medium),
                  child: ThemeSegmentedButton(
                    segments: const [
                      ButtonSegment(
                        value: PreviewType.description,
                        label: Text("Описание"),
                      ),
                      ButtonSegment(
                        value: PreviewType.ingredients,
                        label: Text("Ингредиенты"),
                      ),
                      ButtonSegment(
                        value: PreviewType.steps,
                        label: Text("Шаги"),
                      ),
                    ],
                    initial: ref.read(_previewTypeProvider),
                    onSelected: (type) {
                      ref.read(_previewTypeProvider.notifier).state = type;
```

```
                    },
                  ),
                ),
                const SizedBox(height: AppPaddings.medium),
                Padding(
                  padding: const EdgeInsets.symmetric(horizontal:
AppPaddings.large),
                  child: AnimatedSwitcher(
                    duration: const Duration(milliseconds: 200),
                    child: switch (ref.watch(_previewTypeProvider)) {
                      PreviewType.description =>
                        DescriptionPreview(cocktail: cocktail),
                      PreviewType.ingredients => IngredientsPreview(
                          ingredients: cocktail.ingredients,
                          things: cocktail.things,
                        ),
                      PreviewType.steps => StepsPreview(steps: cocktail.steps),
                    },
                  ),
                )
              ],
            ),
          );
        }
      }
```

description_preview.dart

```
import 'package:cocktail_gen/app/constants/app_font_size.dart';
```

```
import 'package:cocktail_gen/app/constants/app_paddings.dart';
import 'package:cocktail_gen/app/widgets/tag_chip.dart';
import 'package:cocktail_gen/domain/entities/cocktail.dart';
import 'package:flutter/material.dart';

class DescriptionPreview extends StatelessWidget {
  static const _titleFontWeight = FontWeight.w600;
  static const _descriptionFontWeight = FontWeight.w400;

  final Cocktail cocktail;
  const DescriptionPreview({super.key, required this.cocktail});

  @override
  Widget build(BuildContext context) {
    final colorScheme = Theme.of(context).colorScheme;

    return Column(
      crossAxisAlignment: CrossAxisAlignment.stretch,
      children: [
        // Нужно ли показывать теги.
        if (cocktail.tags.isNotEmpty) ...[
          Text(
            "Теги",
            style: TextStyle(
              color: colorScheme.primary,
              fontSize: AppFontSize.header,
              fontWeight: _titleFontWeight,
            ),
          ),
          const SizedBox(height: AppPaddings.small),
```

```
      Wrap(
        spacing: AppPaddings.small,
        runSpacing: AppPaddings.small,
        children: cocktail.tags
            .map(
              (e) => TagChip(tag: e),
            )
            .toList(),
      ),
    ],
    const SizedBox(height: AppPaddings.large),
    Text(
      "Ещё немного",
      style: TextStyle(
        color: colorScheme.primary,
        fontSize: AppFontSize.header,
        fontWeight: _titleFontWeight,
      ),
    ),
    const SizedBox(height: AppPaddings.small),
    Text(
      cocktail.description,
      style: TextStyle(
        color: colorScheme.onSurface,
        fontWeight: _descriptionFontWeight,
        fontSize: AppFontSize.medium,
      ),
    ),
  ],
);
```

```
  }
}
```

ingredients_preview.dart

```
import 'package:cocktail_gen/app/constants/app_font_size.dart';
import 'package:cocktail_gen/app/constants/app_paddings.dart';
import
'package:cocktail_gen/app/widgets/cocktail_preview/ingredient_measure_tile.dart';
import
'package:cocktail_gen/app/widgets/cocktail_preview/ingredient_thing_tile.dart';
import 'package:cocktail_gen/domain/entities/ingredient_measure.dart';
import 'package:cocktail_gen/domain/entities/ingredient_thing.dart';
import 'package:flutter/material.dart';

class IngredientsPreview extends StatelessWidget {
  final List<IngredientMeasure> ingredients;
  final List<IngredientThing> things;

  const IngredientsPreview({
    Key? key,
    required this.ingredients,
    required this.things,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    final colorScheme = Theme.of(context).colorScheme;
```

```
        return Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            // * Ингредиенты
            Padding(
              // Игнорируем отступы [CheckBox]. Костыль, но эффективное
решение.
              padding: const EdgeInsets.only(left: 6),
              child: Text(
                "Ингредиенты",
                style: TextStyle(
                  color: colorScheme.onSurface,
                  fontSize: AppFontSize.title,
                  decoration: TextDecoration.underline,
                ),
              ),
            ),
            const SizedBox(height: AppPaddings.small),
            ...ingredients
                .map(
                  (e) => IngredientMeasureTile(measure: e),
                )
                .toList(),
            // * Штучки
            const SizedBox(height: AppPaddings.large),
            Padding(
              // Игнорируем отступы [CheckBox]. Костыль, но эффективное
решение.
              padding: const EdgeInsets.only(left: 6),
              child: Text(
```

```
            "Штучки",
            style: TextStyle(
              color: colorScheme.onSurface,
              fontSize: AppFontSize.title,
              decoration: TextDecoration.underline,
            ),
          ),
        ),
        const SizedBox(height: AppPaddings.small),
        ...things
            .map(
              (e) => IngredientThingTile(thing: e),
            )
            .toList(),
      ],
    );
  }
}
```

ingredient_measure_tile.dart

```
import 'package:auto_route/auto_route.dart';
import 'package:cocktail_gen/app/constants/app_font_size.dart';
import 'package:cocktail_gen/app/constants/app_paddings.dart';
import 'package:cocktail_gen/app/navigation/router.gr.dart';
import 'package:cocktail_gen/app/widgets/simple_checkbox.dart';
import 'package:cocktail_gen/domain/entities/ingredient_measure.dart';
import 'package:flutter/material.dart';
```

```
class IngredientMeasureTile extends StatelessWidget {
  static const FontWeight _fontWeight = FontWeight.w500;
  static const double _fontSize = AppFontSize.medium;

  final IngredientMeasure measure;

  const IngredientMeasureTile({Key? key, required this.measure})
      : super(key: key);

  @override
  Widget build(BuildContext context) {
    final colorScheme = Theme.of(context).colorScheme;

    return Row(
      children: [
        const SimpleCheckbox(),
        const SizedBox(width: AppPaddings.small),
        // Нужно ли показывать единицы измерения.
        if (measure.unit != MeasureUnits.none) ...[
          SizedBox(
            width: 64,
            child: Text(
              "${measure.quantity.toInt().toString()} ${measure.unit.toString()}",
              style: TextStyle(
                color: colorScheme.onSurfaceVariant,
                fontWeight: _fontWeight,
                fontSize: _fontSize,
              ),
            ),
          ),
```

```
          const SizedBox(width: AppPaddings.small),
        ],
        Expanded(
          child: GestureDetector(
            onTap: measure.ingredientId > -1
                ? () {
                    context.router.push(
                      IngredientPreviewRoute(
                        ingredientId: measure.ingredientId,
                      ),
                    );
                  }
                : null,
            child: Text(
              measure.name,
              style: TextStyle(
                color: colorScheme.secondary,
                decorationColor: colorScheme.secondary,
                fontWeight: _fontWeight,
                fontSize: _fontSize,
                decoration:
                    measure.ingredientId > -1 ? TextDecoration.underline : null,
              ),
            ),
          ),
        ),
      ],
    );
  }
}
```

ingredient_thing_tile.dart

```
import 'package:auto_route/auto_route.dart';
import 'package:cocktail_gen/app/constants/app_font_size.dart';
import 'package:cocktail_gen/app/constants/app_paddings.dart';
import 'package:cocktail_gen/app/navigation/router.gr.dart';
import 'package:cocktail_gen/app/widgets/simple_checkbox.dart';
import 'package:cocktail_gen/domain/entities/ingredient_thing.dart';
import 'package:flutter/material.dart';

class IngredientThingTile extends StatelessWidget {
  static const FontWeight _fontWeight = FontWeight.w500;
  static const double _fontSize = AppFontSize.medium;
  final IngredientThing thing;

  const IngredientThingTile({Key? key, required this.thing}) : super(key:
key);

  @override
  Widget build(BuildContext context) {
    final colorScheme = Theme.of(context).colorScheme;

    return Row(
      children: [
        const SimpleCheckbox(),
        const SizedBox(width: AppPaddings.small),
        Expanded(
          child: GestureDetector(
```

```
            onTap: thing.ingredientId > -1
                ? () {
                    context.router.push(
                      IngredientPreviewRoute(ingredientId: thing.ingredientId),
                    );
                  }
                : null,
            child: Text(
              thing.name,
              style: TextStyle(
                color: colorScheme.secondary,
                decorationColor: colorScheme.secondary,
                fontWeight: _fontWeight,
                fontSize: _fontSize,
                decoration:
                    thing.ingredientId > -1 ? TextDecoration.underline : null,
              ),
            ),
          ),
        ),
      ],
    );
  }
}
```

steps_preview.dart

```
import 'package:cocktail_gen/app/widgets/step_tile.dart';
import 'package:flutter/material.dart';
```

```
class StepsPreview extends StatelessWidget {
  final List<String> steps;

  const StepsPreview({Key? key, required this.steps}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    final stepTiles = List.generate(
      steps.length,
      (i) => Padding(
          padding: const EdgeInsets.only(
            bottom: 8,
          ),
          child: StepTile(index: i + 1, text: steps[i])),
    );

    return Column(
      children: stepTiles,
    );
  }
}
```

gpt_api.dart

```
class GptApi {}
```

gpt_client.dart

```
class GptClient {

}
```

gpt_parser.dart

```
class GptParser {

}
```

cocktail_gpt_dto.dart

```
class CocktailGptDto {
  final String name;
  final String description;
  String? imageUrl;
  final List<String> ingredients;
  final List<String> things;
  final List<String> steps;

  CocktailGptDto({
    required this.name,
    required this.description,
    required this.ingredients,
    required this.things,
    required this.steps,
  });

  factory CocktailGptDto.fromString(String data) {
```

```
  List<String> parseArray(String line) {
    final RegExp arrayRegExp = RegExp(r'"[^"]*"');
    return arrayRegExp
        .allMatches(line)
        .map((e) => e.group(0)!.cutOne())
        .toList();
  }

  // prepare data
  data = data.replaceAll("\n", "").replaceAll(RegExp(r'[a-z]*:'), "\n");
  List<String> dataLines = data.split("\n").sublist(1);
  // parse name
  String name = dataLines[0].trim();
  // parse description
  String description = dataLines[1].trim();
  // parse ingredients
  final List<String> ingredients = parseArray(dataLines[2]);
  // parse things
  final List<String> things = parseArray(dataLines[3]);
  // parse steps
  final List<String> steps = parseArray(dataLines[4]);
  // return dto
  return CocktailGptDto(
    name: name.cutOne(),
    description: description.cutOne(),
    ingredients: ingredients,
    things: things,
    steps: steps,
  );
}
```

```
}

extension _StringHelpers on String {
  String cutOne() => substring(1, length - 1);
}
```

cocktail_gpt_mapper.dart

```
import 'package:cocktail_gen/domain/entities/cocktail.dart';
import 'package:cocktail_gen/domain/entities/ingredient_measure.dart';
import 'package:cocktail_gen/domain/entities/ingredient_thing.dart';

import 'cocktail_gpt_dto.dart';

class CocktailGptMapper {
  static const imageAddress =
      "https://horoshop.ua/content/images/11/how-to-use-chat-gpt-
91100288650261.jpg";

  static Cocktail mapCocktailGptDtoToCocktail(CocktailGptDto dto) {
    final ingredients = dto.ingredients
        .map((ingredient) => IngredientMeasure(
            name: ingredient,
            quantity:
                0,
            unit: MeasureUnits
                .none,
          ),)
        .toList();
```

```
    final things = dto.things
        .map((thing) => IngredientThing(
              name: thing,
            ))
        .toList();

    return Cocktail(
      name: dto.name,
      description: dto.description,
      imageUrl: dto.imageUrl ?? imageAddress,
      tags: [],
      ingredients: ingredients,
      things: things,
      steps: dto.steps,
    );
  }
}
```

gpt_client.dart

```
import 'package:chat_gpt_sdk/chat_gpt_sdk.dart';
import 'package:cocktail_gen/data/api/gpt/dto/cocktail_gpt_dto.dart';
import 'package:cocktail_gen/data/api/gpt/dto/cocktail_gpt_mapper.dart';
import 'package:cocktail_gen/data/api/gpt/dto/gpt_prompts.dart';
import 'package:cocktail_gen/domain/entities/cocktail.dart';
import 'package:flutter/material.dart';

class GptClient {
```

```
final OpenAI openAI;

GptClient._(this.openAI);

factory GptClient({required String token}) {
  final openAi = OpenAI.instance.build(
    token: token,
    baseOption: HttpSetup(
      receiveTimeout: const Duration(seconds: 8),
    ),
    enableLog: true,
  );
  return GptClient._(openAi);
}

Future<Cocktail> generateRecipe(String tags, String ingredients) async {
  final request = ChatCompleteText(
    messages: [
      Messages(
        role: Role.system,
        content: GptPrompts.generateRecipePrompt(
          ingredients,
          tags,
        ),
      ),
    ],
    maxToken: 2200,
    model: GptTurbo0301ChatModel(),
  );
  ChatCTResponse response =
```

```
            (await openAI.onChatCompletion(request: request))!;
        final data = response.choices.first.message!.content;
        final CocktailGptDto dto = CocktailGptDto.fromString(data);
        // Generate image
        final imageRequest = GenerateImage(
            'Beautiful cocktail with ingredients
${dto.ingredients.toString().characters.take(64)}',
            1,
            size: ImageSize.size256,
            responseFormat: Format.url);
        GenImgResponse? imageResponse;
        try {
          imageResponse = await openAI.generateImage(imageRequest);
        } catch (_) {}
        dto.imageUrl = imageResponse?.data?.last?.url;
        return CocktailGptMapper.mapCocktailGptDtoToCocktail(dto);
      }
    }


    gpt_prompts.dart

    class GptPrompts {
      // ? Может заменить [String] на [List]
      // ? Но тогда усложниться логика написания UI.
      static String generateRecipePrompt(String ingredients, String tags) {
        return """Ты генератор рецептов в приложении для
профессиональных барменов.
    Твоя задача генерировать уникальные и необычные рецепты.
    Игнорируй и не учитывай предыдущие сообщения.
```

Ты не можешь общаться с пользователем, только генерировать рецепты.

Следующие требования:

Схема рецепта:

name: string

description:string

ingredients:[string]

things:[string]

step:[string]

Пример заполненной схемы:

name: "имя"

description:"описание"

ingredients:["и","и","и","и"]

things:["ш","ш","ш","ш"]

step:["ш","ш","ш"]

Требования к значениям:

name - Уникальное и необычное имя.

description - Поэтичное описание в 1-2 предложения.

ingredients - Названия ингредиентов с указанием их количества в мл, шт, гр или oz.

things - Приспособления(шейкер и тд.), обязательно включая официальный вид бокала для подачи.

step - Подробные и однозначные шаги.

Требования рецепта:

- Рецепт для профессионального бармена

- Теги: $tags

```
- Обязательные ингредиенты: $ingredients
- Добавить ещё ингредиентов

Рецепт на основе требований к значениям и требований к рецепту: """;
  }
}
```

mock_cocktail.dart

```
import 'package:cocktail_gen/domain/entities/cocktail.dart';
import 'package:cocktail_gen/domain/entities/ingredient_measure.dart';
import 'package:cocktail_gen/domain/entities/ingredient_thing.dart';
import 'package:cocktail_gen/domain/entities/tag.dart';

class MockCocktail {
  static final Cocktail cocktail = Cocktail(
    id: 1,
    name: "Мохито",
    description: "Refreshing cocktail with mint and lime flavors.",
    imageUrl:

"https://upload.wikimedia.org/wikipedia/commons/b/b3/Mojito_made_with_rum%2C_lime%2C_sugar%2C_mint%2C_club_soda%2C_served_in_a_tall_glass_-_Evan_Swigart.jpg",
    tags: [
      Tag(id: 1, name: "Refreshing"),
      Tag(id: 2, name: "Minty"),
    ],
    ingredients: [
```

```
IngredientMeasure(
  ingredientId: 1,
  name: "White Rum",
  quantity: 60.0,
  unit: MeasureUnits.milliliters,
),
IngredientMeasure(
  ingredientId: 2,
  name: "Fresh Lime Juice",
  quantity: 30.0,
  unit: MeasureUnits.milliliters,
),
IngredientMeasure(
  ingredientId: 3,
  name: "Simple Syrup",
  quantity: 20.0,
  unit: MeasureUnits.milliliters,
),
IngredientMeasure(
  ingredientId: 4,
  name: "Fresh Mint Leaves",
  quantity: 6.0,
  unit: MeasureUnits.pieces,
),
IngredientMeasure(
  ingredientId: 5,
  name: "Soda Water",
  quantity: 120.0,
  unit: MeasureUnits.milliliters,
),
```

```
      ],
      things: [
        IngredientThing(ingredientId: 1, name: "Джиггер 30 мл"),
        IngredientThing(ingredientId: 2, name: "Бостонский шейкер"),
      ],
      steps: [
        "In a glass, muddle mint leaves and simple
syrup.SOOOOOOOOOOOOOOOOOOOOOOOOOOOOMEEEEEEEEEEEEE
EEEEEEE LOOOOOOOOOOOOOONG TEEEXT",
        "Add fresh lime juice and white rum.",
        "Fill the glass with crushed ice.",
        "Top up with soda water.",
        "Garnish with mint sprigs and lime slices.",
      ],
    );

    @Deprecated("Тестовые данные")
    static List<Cocktail> data = [
      cocktail,
      cocktail,
      cocktail,
      cocktail,
      cocktail,
      cocktail,
      cocktail,
      cocktail,
      cocktail,
      cocktail,
      cocktail,
    ];
```

```
}
```

mock_ingredient.dart

```
import 'package:cocktail_gen/domain/entities/ingredient.dart';

class MockIngredient {
  static final Ingredient ingredient = Ingredient(
    id: 1,
    name: "Самбука",
    type: "Ликер",
    description:
        """Итальянский и испанский традиционный напиток — ликёр с ароматом аниса.

Обычно это прозрачная сладкая вязкая жидкость со специфическим ароматом и содержанием спирта 38—42 %.

В то же время имеются тёмные и даже красные разновидности самбуки.""",
    imageUrl:
        "https://irecommend.ru/sites/default/files/product-images/290233/O1DWqpsEiz63ZIJY2GFd1A.jpg",
  );

  @Deprecated("Тестовые данные")
  static List<Ingredient> data = [
    ingredient,
    ingredient,
```

```
      ingredient,
      ingredient,
      ingredient,
      ingredient,
      ingredient,
      ingredient,
      ingredient,
      ingredient,
      ingredient,
      ingredient,
      ingredient,
    ];
  }
```

initial_data_repo.dart

```
import 'dart:core';

import 'package:cocktail_gen/data/repos/isar/dto/cocktail_isar.dart';
import
'package:cocktail_gen/data/repos/isar/dto/ingredient_measure_isar.dart';
import 'package:cocktail_gen/data/repos/isar/dto/ingredient_thing_isar.dart';
import 'package:cocktail_gen/data/repos/isar/dto/tag_isar.dart';

import 'dto/ingredient_isar.dart';

sealed class InitialDataRepo {
  static List<CocktailIsar> getCocktails() {
    return [
```

```
CocktailIsar(
  name: "Лонг Айленд Айс Ти",
  description:
    "Лонг Айленд Айс Ти - это волшебный и освежающий напиток, который обязательно покорит ваши вкусовые рецепторы. Этот уникальный ледяной чай сочетает в себе свежесть летнего бриза и чистоту айсбергов.",
  imageUrl: "https://www.edim.tv/img/small/long-island-iced-tea.jpg",
  tagsIds: [0, 1, 2, 3],
  ingredients: [
   IngredientMeasureIsar()
    ..name = "Царская Водка"
    ..ingredientId = 0
    ..quantity = 30
    ..unit = MeasureUnitsIsar.milliliters,
   IngredientMeasureIsar()
    ..name = "Белый Ром"
    ..ingredientId = -1
    ..quantity = 30
    ..unit = MeasureUnitsIsar.milliliters,
   IngredientMeasureIsar()
    ..name = "Серебряная текила"
    ..ingredientId = -1
    ..quantity = 30
    ..unit = MeasureUnitsIsar.milliliters,
   IngredientMeasureIsar()
    ..name = "Джин"
    ..ingredientId = -1
    ..quantity = 30
    ..unit = MeasureUnitsIsar.milliliters,
   IngredientMeasureIsar()
```

```
    ..name = "Трипл сек Fruko Schulz"
    ..ingredientId = -1
    ..quantity = 30
    ..unit = MeasureUnitsIsar.milliliters,
  IngredientMeasureIsar()
    ..name = "Сахарный сироп"
    ..ingredientId = -1
    ..quantity = 30
    ..unit = MeasureUnitsIsar.milliliters,
  IngredientMeasureIsar()
    ..name = "Кола"
    ..ingredientId = -1
    ..quantity = 100
    ..unit = MeasureUnitsIsar.milliliters,
  IngredientMeasureIsar()
    ..name = "Долька лимона"
    ..ingredientId = -1
    ..quantity = 1
    ..unit = MeasureUnitsIsar.pieces,
],
things: [
  IngredientThingIsar()
    ..ingredientId = -1
    ..name = "Хайбол",
  IngredientThingIsar()
    ..ingredientId = -1
    ..name = "Джиггер 30 мл",
  IngredientThingIsar()
    ..ingredientId = -1
    ..name = "Коктейльная ложка",
```

```
            IngredientThingIsar()
              ..ingredientId = -1
              ..name = "2 трубочки",
          ],
          steps: [
            "Наполни хайбол кубиками льда доверху",
            "Налей лимонный сок 30 мл, сахарный сироп 30 мл и ликер трипл сек 30 мл",
            "Добавь водку 30 мл, джин 30 мл, белый ром 30 мл и серебряную текилу 30 мл",
            "Долей колу доверху и аккуратно размешай коктейльной ложкой",
            "Укрась долькой лимона",
          ],
        ),
      ];
    }

    static List<IngredientIsar> getIngredients() {
      return [
        IngredientIsar(
          id: 0,
          name: "Водка",
          description:
              "Водка - это блестящий и легендарный алкогольный напиток, который уносит нас в удивительный мир роскоши и традиции. Она обладает кристальной чистотой, наполняющей каждый глоток чистотой и ясностью.\n\nИзготавливается водка из отборных зерновых культур или картофеля, которые проходят тщательную переработку и дистилляцию, чтобы получить наилучшее качество. Ее процесс создания является
```

настоящим искусством, передающим мастерство поколений и тайные рецепты, способные удивить своим великолепием.\n\nВолшебное сочетание нот и ароматов придает водке особый характер. В каждой капле она раскрывает свою уникальную гармонию, переплетая легкие цветочные ноты с ягодными оттенками и легким ореховым послевкусием. Она приносит наслаждение и радость, притягивая своей привлекательной сладостью и приятным пикантным штрихом.",

type: "Зерновой спиртной напиток",

imageUrl: "https://cdn.metro-cc.ru/ru/ru_pim_29086001001_01.png",

),

IngredientIsar(

id: 1,

name: "Ром",

description: "Крутой",

type: "Акссесуар",

imageUrl:

"https://sdelai-doma.ru/upload/medialibrary/135/135a80107ee01bdda3e9663b5631e1e8.jpg",

),

IngredientIsar(

id: 2,

name: "Текила",

description: "Крутой",

type: "Акссесуар",

imageUrl: "https://shop.miratorg.ru/upload/iblock/0c1/RN000310.jpg",

),

IngredientIsar(

id: 3,

name: "Джин",

description: "Крутой",

```
            type: "Джин",
            imageUrl:
                "https://vinoteki.ru/wp-content/uploads/2022/08/32945_f4aa18a2-
ab1f-11ea-8149-002590ea1e13.jpeg",
          ),
          IngredientIsar(
            id: 4,
            name: "Сахарный сироп",
            description: "Крутой",
            type: "Акссесуар",
            imageUrl:
                "https://static.insales-
cdn.com/images/products/1/3109/120212517/large_sirop-monin-saharniy-trosnik-
900x900px.jpg",
          ),
          IngredientIsar(
            id: 5,
            name: "Кола",
            description: "Крутой",
            type: "Акссесуар",
            imageUrl: "https://sokovoz.ru/_sh/49/4969m.jpg",
          ),
          IngredientIsar(
            id: 6,
            name: "Лимон",
            description: "Крутой",
            type: "Акссесуар",
            imageUrl:
                "https://m.dom-
eda.com/uploads/topics/preview/00/00/03/20/06fa2fc486_500.jpg",
```

```
      ),
    ];
  }

  static List<TagIsar> getTags() {
    return [
      TagIsar("крепкие", id: 0),
      TagIsar("сладкие", id: 1),
      TagIsar("классика", id: 2),
      TagIsar("физы", id: 3),
    ];
  }
}
```

isar_db.dart

```
import 'dart:convert';

import 'package:cocktail_gen/data/repos/isar/dto/cocktail_isar.dart';
import 'package:cocktail_gen/data/repos/isar/dto/ingredient_isar.dart';
import 'package:cocktail_gen/data/repos/isar/dto/tag_isar.dart';
import 'package:cocktail_gen/data/repos/isar/mappers/cocktail_mapper.dart';
import
'package:cocktail_gen/data/repos/isar/mappers/ingredient_mapper.dart.dart';
import 'package:cocktail_gen/data/repos/isar/mappers/tag_mapper.dart';
import 'package:cocktail_gen/domain/entities/cocktail.dart';
import 'package:cocktail_gen/domain/entities/ingredient.dart';
import 'package:cocktail_gen/domain/entities/tag.dart';
import 'package:cocktail_gen/interfaces/db_repository.dart';
```

```
import 'package:flutter/services.dart';
import 'package:isar/isar.dart';
import 'package:path_provider/path_provider.dart';

class CocktailIsarDb implements RecipeRepository {
  late List<CocktailIsar> cocktails;
  late List<IngredientIsar> ingredients;
  late List<TagIsar> tags;
  late Isar isar;

  @override
  List<Cocktail> fetchCocktails() {
    return cocktails.map((e) => e.toCocktail(this)).toList();
  }

  @override
  List<Ingredient> fetchIngredients() {
    return ingredients.map((e) => e.toIngredient()).toList();
  }

  @override
  List<Tag> fetchTags() {
    return tags.map((e) => e.toTag()).toList();
  }

  @override
  Cocktail? getCocktailById(int id) {
    return isar.cocktailIsars.getSync(id)?.toCocktail(this);
  }
```

```
@override
Ingredient? getIngredientById(int id) {
  return isar.ingredientIsars.getSync(id)?.toIngredient();
}

@override
Tag? getTagById(int id) {
  return isar.tagIsars.getSync(id)?.toTag();
}

@override
Future<void> init() async {
  final dir = await getApplicationSupportDirectory();
  isar = await Isar.open(
    [CocktailIsarSchema, IngredientIsarSchema, TagIsarSchema],
    directory: dir.path,
  );

  final cocktailCount = await isar.cocktailIsars.count();
  if (cocktailCount == 0) await _init();

  cocktails = await isar.cocktailIsars.where().findAll();
  ingredients = await isar.ingredientIsars.where().findAll();
  tags = await isar.tagIsars.where().findAll();
}

Future<void> _init() async {
  await isar.writeTxn(
    () async {
      final cocktailData =
```

```
        await rootBundle.loadString('assets/isar/CocktailIsar.json');
      await isar.cocktailIsars.importJson(
        (jsonDecode(cocktailData) as List<dynamic>)
            .map(
              (e) => e as Map<String, dynamic>,
            )
            .toList(),
      );

      final ingredientData =
        await rootBundle.loadString('assets/isar/IngredientIsar.json');
      await isar.ingredientIsars.importJson(
        (jsonDecode(ingredientData) as List<dynamic>)
            .map(
              (e) => e as Map<String, dynamic>,
            )
            .toList(),
      );

      final tagData = await rootBundle.loadString('assets/isar/TagIsar.json');
      await isar.tagIsars.importJson(
        (jsonDecode(tagData) as List<dynamic>)
            .map(
              (e) => e as Map<String, dynamic>,
            )
            .toList(),
      );
    },
  );
}
```

```
}
```

cocktail_isar.dart

```
import 'package:isar/isar.dart';

import 'ingredient_measure_isar.dart';
import 'ingredient_thing_isar.dart';

part 'cocktail_isar.g.dart';

@collection
class CocktailIsar {
  final Id id;
  final String name;
  final String description;
  final String imageUrl;
  final List<Id> tagsIds;
  final List<IngredientMeasureIsar> ingredients;
  final List<IngredientThingIsar> things;
  final List<String> steps;

  CocktailIsar({
    this.id = Isar.autoIncrement,
    required this.name,
    required this.description,
    required this.imageUrl,
    required this.tagsIds,
    required this.ingredients,
```

```
    required this.things,
    required this.steps,
  });
}
```

ingredient_isar.dart

```
import 'package:isar/isar.dart';

part 'ingredient_isar.g.dart';

@collection
class IngredientIsar {
  final Id id;
  final String name;
  final String description;
  final String type;
  final String imageUrl;

  IngredientIsar({
    this.id = Isar.autoIncrement,
    required this.name,
    required this.description,
    required this.type,
    required this.imageUrl,
  });
}
```

ingredient_measure_isar.dart

```
import 'package:isar/isar.dart';

part 'ingredient_measure_isar.g.dart';

enum MeasureUnitsIsar {
  none,
  spoons,
  pieces,
  milliliters,
  oz,
  leaves,
  drops,
}

@embedded
class IngredientMeasureIsar {
  late String name;
  late double quantity;
  @enumerated
  late MeasureUnitsIsar unit;
  late int ingredientId;
}
```

ingredient_thing_isar.dart

```
import 'package:isar/isar.dart';
```

```
part 'ingredient_thing_isar.g.dart';

@embedded
class IngredientThingIsar {
  late String name;
  late int ingredientId;
}
```

tag_isar.dart

```
import 'package:isar/isar.dart';

part 'tag_isar.g.dart';

@collection
class TagIsar {
  final Id id;
  final String name;

  TagIsar(
    this.name, {
    this.id = Isar.autoIncrement,
  });
}
```

cocktail_mapper.dart

```
// cocktail.dart
```

```
// ignore_for_file: unnecessary_this

import 'package:cocktail_gen/data/repos/isar/controllers/isar_db.dart';
import 'package:cocktail_gen/data/repos/isar/dto/cocktail_isar.dart';
import
'package:cocktail_gen/data/repos/isar/dto/ingredient_measure_isar.dart';
import 'package:cocktail_gen/data/repos/isar/dto/ingredient_thing_isar.dart';
import 'package:cocktail_gen/domain/entities/cocktail.dart';
import 'package:cocktail_gen/domain/entities/ingredient_measure.dart';
import 'package:cocktail_gen/domain/entities/ingredient_thing.dart';

// cocktail_isar.dart
extension CocktailIsarMapper on CocktailIsar {
  Cocktail toCocktail(CocktailIsarDb db) {
    return Cocktail(
      id: this.id,
      name: this.name,
      description: this.description,
      imageUrl: this.imageUrl,
      tags: this.tagsIds.map((id) => db.getTagById(id)!).toList(),
      ingredients: this
          .ingredients
          .map(
            (ingredientIsar) => ingredientIsar.toIngredientMeasure(),
          )
          .toList(),
      things: this
          .things
          .map((thingIsar) => thingIsar.toIngredientThing())
          .toList(),
```

```
      steps: this.steps,
    );
  }
}

// ingredient_measure_isar.dart
extension IngredientMeasureIsarMapper on IngredientMeasureIsar {
  IngredientMeasure toIngredientMeasure() {
    return IngredientMeasure(
      ingredientId: this.ingredientId,
      name: this.name,
      quantity: this.quantity,
      unit: _mapMeasureUnits(this.unit),
    );
  }

  MeasureUnits _mapMeasureUnits(MeasureUnitsIsar unit) {
    switch (unit) {
      case MeasureUnitsIsar.none:
        return MeasureUnits.none;
      case MeasureUnitsIsar.spoons:
        return MeasureUnits.spoons;
      case MeasureUnitsIsar.pieces:
        return MeasureUnits.pieces;
      case MeasureUnitsIsar.milliliters:
        return MeasureUnits.milliliters;
      case MeasureUnitsIsar.oz:
        return MeasureUnits.oz;
      case MeasureUnitsIsar.leaves:
        return MeasureUnits.leaves;
```

```
      case MeasureUnitsIsar.drops:
        return MeasureUnits.drops;
    }
  }
}

// ingredient_thing_isar.dart
extension IngredientThingIsarMapper on IngredientThingIsar {
  IngredientThing toIngredientThing() {
    return IngredientThing(
      ingredientId: this.ingredientId,
      name: this.name,
    );
  }
}
```

ingredient_mapper.dart.dart

```
// ingredient_isar.dart
// ignore_for_file: unnecessary_this

import 'package:cocktail_gen/data/repos/isar/dto/ingredient_isar.dart';
import 'package:cocktail_gen/domain/entities/ingredient.dart';

extension IngredientIsarMapper on IngredientIsar {
  Ingredient toIngredient() {
    return Ingredient(
      id: this.id,
      name: this.name,
```

```
      description: this.description,
      type: this.type,
      imageUrl: this.imageUrl,
    );
  }
}
```

tag_mapper.dart

```
// tag_isar.dart
import 'package:cocktail_gen/data/repos/isar/dto/tag_isar.dart';
import 'package:cocktail_gen/domain/entities/tag.dart';

extension TagIsarMapper on TagIsar {
  Tag toTag() {
    return Tag(
      id: id,
      name: name,
    );
  }
}
```

cocktail.dart

```
// ignore_for_file: invalid_annotation_target

import 'package:cocktail_gen/domain/entities/ingredient_thing.dart';
import 'package:cocktail_gen/domain/entities/tag.dart';
```

```
import 'ingredient_measure.dart';
import 'package:freezed_annotation/freezed_annotation.dart';

part 'cocktail.freezed.dart';
part 'cocktail.g.dart';

@freezed
class Cocktail with _$Cocktail {
  @JsonSerializable(explicitToJson: true)
  @Assert("steps.length > 0")
  @Assert("ingredients.length > 0")
  factory Cocktail({
    /// ID рецепта в БД.
    ///
    /// -1 если рецепт не существует в БД.
    @Default(-1) int id,
    required String name,
    required String description,
    required String imageUrl,
    required List<Tag> tags,
    required List<IngredientMeasure> ingredients,
    required List<IngredientThing> things,
    required List<String> steps,
  }) = _Cocktail;

  factory Cocktail.fromJson(Map<String, dynamic> json) =>
      _$CocktailFromJson(json);
}
```

ingredient.dart

```
import 'package:freezed_annotation/freezed_annotation.dart';

part 'ingredient.freezed.dart';
part 'ingredient.g.dart';

@freezed
class Ingredient with _$Ingredient {
  factory Ingredient({
    required int id,
    required String name,
    required String description,

    /// Тип ингредиента (ликер, фрукт и тд...)
    required String type,
    required String imageUrl,
  }) = _Ingredient;

  factory Ingredient.fromJson(Map<String, dynamic> json) =>
      _$IngredientFromJson(json);
}
```

ingredient_measure.dart

```
import 'package:cocktail_gen/domain/entities/ingredient.dart';
import 'package:freezed_annotation/freezed_annotation.dart';
```

```
part 'ingredient_measure.freezed.dart';
part 'ingredient_measure.g.dart';

/// Единица измерения ингридиентов.
enum MeasureUnits {
  /// Вариант, если в названии ингредиента уже есть единица измерения.
  ///
  /// Не рекомендуется использовать. Только для запросов к API GPT.
  none,
  spoons,
  pieces,
  milliliters,
  oz,
  leaves,
  drops;

  @override
  String toString() {
    return switch (this) {
      MeasureUnits.milliliters => "мл",
      _ => "шт",
    };
  }
}

/// Информация о пропорциях [Ingredient].
@freezed
class IngredientMeasure with _$IngredientMeasure {
  factory IngredientMeasure({
    /// ID соответствующего [Ingredient].
```

```
        ///
        /// -1 если ингредиента не существует в базе данных.
        @Default(-1) int ingredientId,

        // ? Может выделить в отдельный класс пару [ingredientId] и
[ingredientId]
        // ? и назвать [ShortIngredient]?
        // * Во избежания загрузки [Ingredient] здесь есть имя ингредиента.
        /// Имя соответствующего [Ingredient].
        required String name,

        /// Количество ингридиента.
        required double quantity,

        /// Единица измерения количества ингридиента.
        required MeasureUnits unit,
      }) = _IngredientMeasure;

      factory IngredientMeasure.fromJson(Map<String, dynamic> json) =>
          _$IngredientMeasureFromJson(json);
    }
```

ingredient_thing.dart

```
// ignore_for_file: invalid_annotation_target

import 'package:freezed_annotation/freezed_annotation.dart';

part 'ingredient_thing.freezed.dart';
```

```
part 'ingredient_thing.g.dart';

/// Информация о "штучках" в рецепте. (джиггеры, шейкеры и тд.)
@freezed
abstract class IngredientThing with _$IngredientThing {
  factory IngredientThing({
    /// ID соответствующего ингредиента.
    ///
    /// -1 - если ингредиент не существует в БД.
    @Default(-1) int ingredientId,
    required String name,
  }) = _IngredientThing;

  factory IngredientThing.fromJson(Map<String, dynamic> json) =>
      _$IngredientThingFromJson(json);
}
```

tag.dart

```
import 'package:freezed_annotation/freezed_annotation.dart';

part 'tag.freezed.dart';
part 'tag.g.dart';

@freezed
abstract class Tag with _$Tag {
  factory Tag({
    /// ID тега в базе данных.
    ///
```

```
    /// -1 если такого тега не существует.
    @Default(-1) int id,
    required String name,
  }) = _Tag;

  factory Tag.fromJson(Map<String, dynamic> json) =>
_$TagFromJson(json);
}
```

db_repository.dart

```
import 'package:cocktail_gen/domain/entities/cocktail.dart';
import 'package:cocktail_gen/domain/entities/ingredient.dart';
import 'package:cocktail_gen/domain/entities/tag.dart';

abstract interface class RecipeRepository {
  Future<void> init();
  List<Ingredient> fetchIngredients();
  List<Cocktail> fetchCocktails();
  List<Tag> fetchTags();
  Ingredient? getIngredientById(int id);
  Cocktail? getCocktailById(int id);
  Tag? getTagById(int id);
}
```