МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА №42

КУРСОВАЯ РАБОТА (ПРОЕКТ)
ЗАЩИЩЕНА С ОЦЕНКОЙ

РУКОВОДИТЕЛЬ

Старший преподаватель		С. Ю. Гуков
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К КУРСОВОЙ РАБОТЕ (ПРОЕКТУ)

по дисциплине:

ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ гр. № 4128		Воробьев В.А.
	подпись, дата	инициалы, фамилия

Санкт-Петербург 2023

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Современное программирование требует от разработчиков не только умения создавать работающие программы, но и обладать глубоким пониманием принципов разработки, применения архитектурных шаблонов и использования паттернов проектирования. В рамках данной курсовой работы предлагается разработать программный продукт среднего уровня сложности, который имитирует процесс покупки товаров или услуг в магазине. Основная цель работы заключается в разработке качественной архитектуры проекта и закреплении полученных знаний по различным технологиям программирования.

Задачей проекта является создание программы, которая предоставит возможность покупателю, обладающему персональными данными, бонусной картой, кошельком и списком покупок, осуществить покупку в магазине. Пользователь сможет выбрать необходимые товары или услуги, добавить их в корзину, взвесить товары перед покупкой и оплатить покупку с использованием наличных средств или бонусов. Кроме того, необходимо предусмотреть ситуации, когда у покупателя недостаточно средств для совершения покупки или некоторые товары не взвешены. В таких случаях покупатель должен иметь возможность удалить товары из корзины до тех пор, пока сумма не станет доступной для оплаты. Проект может быть выполнен в виде консольного приложения с командно-текстовым интерфейсом или с использованием графического пользовательского интерфейса (UI). Также разработка может быть выполнена на любом выбранном языке программирования.

Структура проекта должна соответствовать принципам объектно-ориентированного программирования и SOLID. Программа должна обладать дружелюбным командно-текстовым или графическим пользовательским интерфейсом, использовать шаблоны архитектуры системы и паттерны проектирования.

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

Описание: необходимо разработать программу, которая имитирует покупку товаров или услуг в магазине. Покупатель, у которого есть персональные данные, бонусная карта, кошелек и список покупок, приходит в магазин, в котором он может приобрести какие-то товары или услуги. Покупатель может положить товар в корзину, выложить товар, оплатить все. Также покупатель перед тем, как положить в корзину некоторые товары, должен их взвесить. Покупатель может полностью оплатить покупку наличными средствами либо бонусами или часть чека он может оплатить баллами, а остальное наличными. Необходимо также предусмотреть ситуации, при которой у покупателя не хватает средств для совершения покупки либо некоторые товары не взвешены. Если средств не хватает для совершения покупки, то покупатель может выкладывать товары из корзины до тех пор, пока не хватит средств для совершения покупки.

Проект может быть выполнен либо в качестве консольного приложения (тогда обязателен командно-текстовый интерфейс), либо иметь графический пользовательский интерфейс (User Interface, UI), а также может быть написан на любом языке программирования.

Требования к структуре проекта:

1) Применение принципов ООП (наследования, инкапсуляции, полиморфизма, абстракции) и SOLID;
2) Дружелюбный командно-текстовый либо графический пользовательский интерфейс;
3) Использование шаблонов архитектуры системы;
4) Использование паттернов проектирования.

КРАТКОЕ ОПИСАНИЕ ХОДА РАЗРАБОТКИ И НАЗНАЧЕНИЯ ИСПОЛЬЗУЕМЫХ ТЕХНОЛОГИЙ

Программа была реализована на языке Dart и кроссплатформенном фреймворке Flutter. Для разработки был выбран графический интерфейс, за удобство работы с пользователем.

При разработке я следовал принципам KISS, SOLID, DRY, YAGNI. Для разработки интерфейса – сначала набросал интерфейс в Figma (URL - https://www.figma.com/proto/tmUiXnFIr2Vvp52QYEcX2T/BlagoDat?type=design&node-id=8-3&scaling=scale-down&page-id=5%3A2&starting-point-node-id=8%3A3), а затем реализовал мокапы в приложении. Для системы контроля версии использовал Git и хостинг GitHub (URL - https://github.com/vladcto/blagodat).

1. Dart: Язык программирования, на котором разработано приложение. Dart предлагает эффективный и простой в использовании синтаксис, а также поддерживает множество платформ, включая Flutter, который используется для создания кросс-платформенных мобильных приложений.

2. Flutter: Фреймворк для разработки кросс-платформенных мобильных приложений. Он предоставляет набор инструментов и виджетов для создания пользовательского интерфейса и управления логикой приложения. Flutter позволяет создавать высокопроизводительные и эффективные приложения, которые могут работать на различных платформах.

3. Riverpod: Пакет управления состоянием для Flutter, основанный на концепции Provider. Riverpod облегчает управление состоянием приложения, предоставляя инъекцию зависимостей и возможность

обновления состояния в реактивном стиле. Он используется для организации зависимостей и управления состоянием в приложении.

4. Provider: Библиотека для управления состоянием в Flutter, которая предлагает простой и эффективный способ обеспечить доступ к данным и уведомлять виджеты об изменениях состояния. Provider используется вместе с Riverpod для предоставления и управления зависимостями в приложении.

5. MockAssortment: Класс, который предоставляет фиктивные данные о товарах. Он используется в приложении для имитации реальных данных и упрощения разработки и тестирования функциональности.

6. Transaction: Класс, представляющий транзакцию покупки. Он содержит информацию о стоимости покупки и времени совершения. Объекты этого класса используются для записи и передачи информации о покупках в приложении.

7. BonusProgram и BonusProvider: Классы и провайдер, связанные с бонусной программой. Они предоставляют функциональность для работы с бонусами, включая выполнение транзакций, списание бонусов и управление бонусными баллами пользователя.

8. DiscountProvider: Провайдер, отвечающий за предоставление скидок на покупки. Он используется вместе с PurchaseManager для определения доступных скидок при оформлении покупок.

9. CartStateNotifier: Класс, отвечающий за управление состоянием корзины покупок.

Он предоставляет методы для добавления, удаления и очистки товаров в корзине, а также уведомляет связанные виджеты об изменениях состояния корзины.

10. PurchaseManager: Класс, осуществляющий оформление покупок. Он связывает различные компоненты приложения, такие как корзина, скидки и бонусная программа, для обработки и регистрации покупок. PurchaseManager также предоставляет поток для отслеживания успешных покупок.

Ход разработки включал создание классов и провайдеров, установку необходимых зависимостей, написание логики оформления покупок и управления состоянием, а также тестирование функциональности приложения. Технологии Dart, Flutter, Riverpod и Provider были использованы для создания эффективного и гибкого приложения с хорошо управляемым состоянием и инъекцией зависимостей. 1. Dart: Язык программирования, на котором разработано приложение. Dart предлагает эффективный и простой в использовании синтаксис, а также поддерживает множество платформ, включая Flutter, который используется для создания кросс-платформенных мобильных приложений.

ПОЛЬЗОВАТЕЛЬСКАЯ ДОКУМЕНТАЦИЯ

При запуске приложения вас графический интерфейс, пройдемся по примерному flow приложения.

Рисунок 1 – Домашняя страница

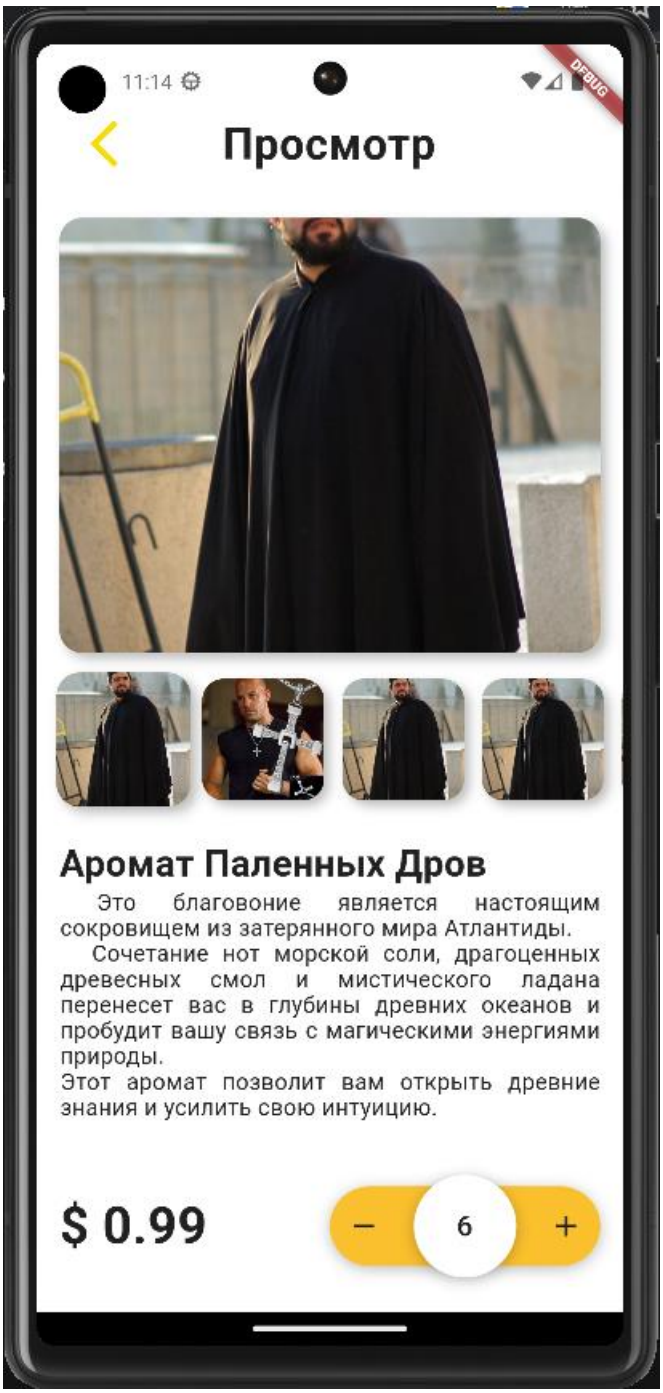

Рисунок 2 – Экран товаров

Рисунок 3 – Смена картинок

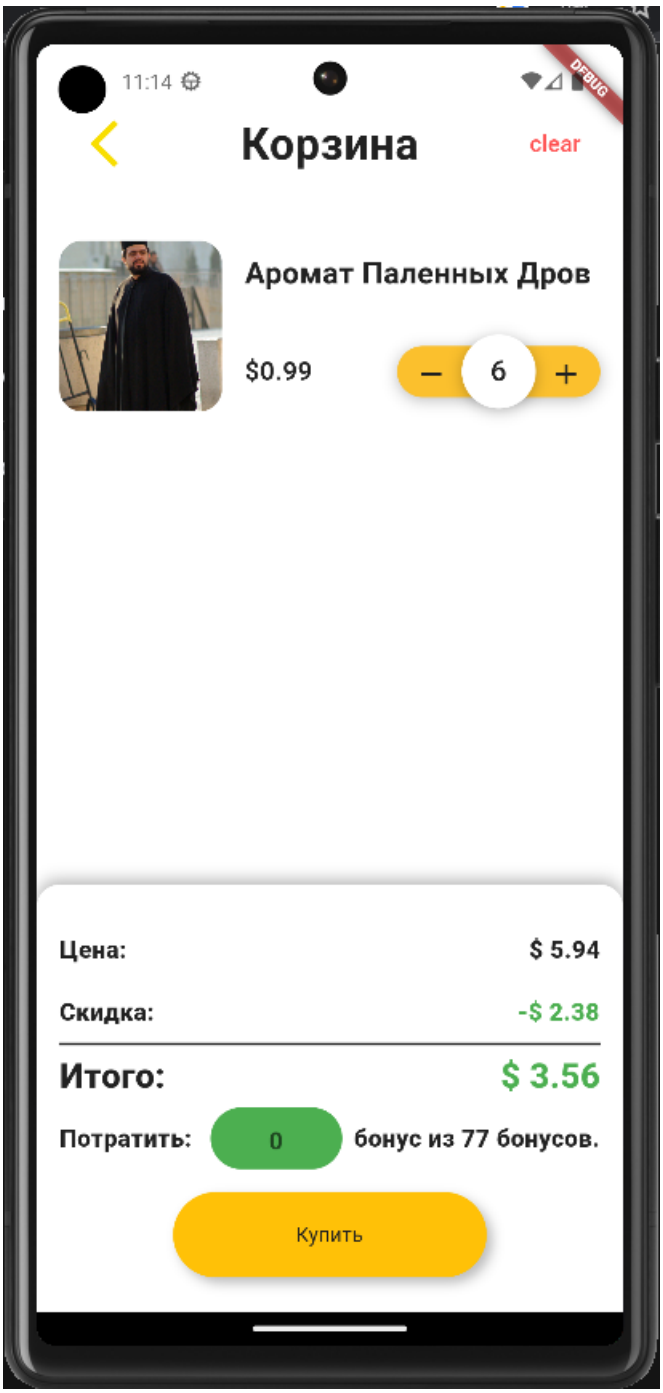

Рисунок 4 – Просмотр корзины

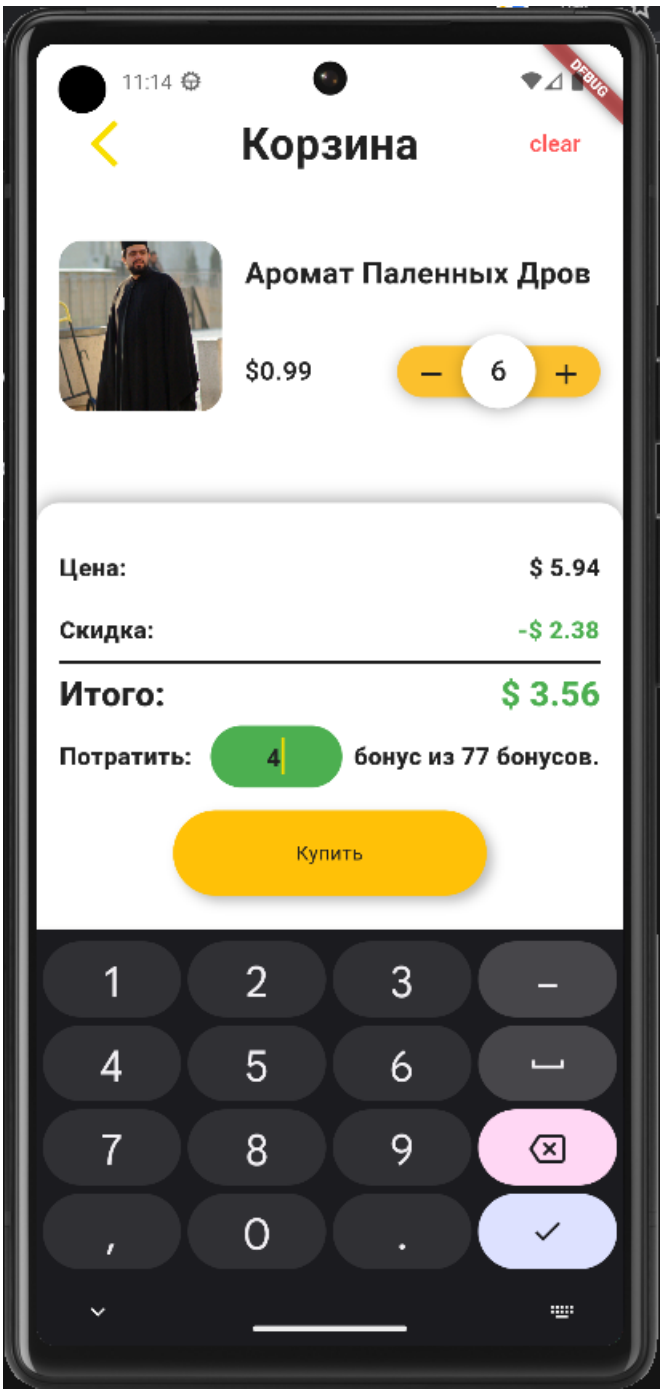

Рисунок 5 – Изменение количества бонусов

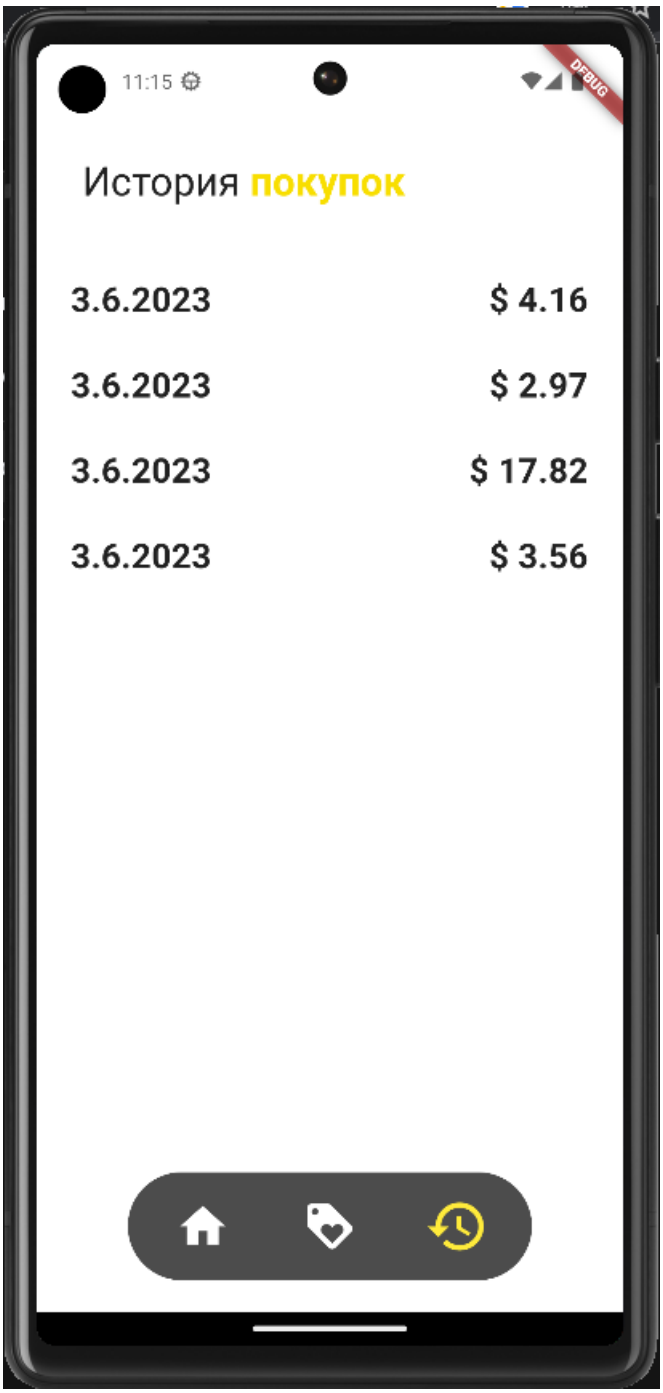

Рисунок 6 – История покупок

ТЕХНИЧЕСКАЯ ДОКУМЕНТАЦИЯ

1. Описание архитектуры:
   - Архитектура проекта выполнена в соотв. с соглашениями Flutter. Архитектура выполнена в реактивном стиле посредством подписок на провайдеров.
2. Используемые компоненты:
   - Язык программирования: Dart, Flutter
   - Среда разработки: VS Code
   - State-management и DI – Riverpod
   - Библиотеки: Flutter Material, Freezed.
3. Шаги установки:
   - Для установки и запуска проекта, следуйте этим шагам:
     - Убедитесь, что установлен Flutter SDK и Dart.
     - Находясь в корне проекте – напишите dart pub get.
     - Затем введите выбери устройство, на котором хотите запустить.
     - Напишите flutter run.
4. Документация API:
   - Для документации я использовал средства автодокуменетирования Dart.
   - Документация доступна на GitHub (URL - https://github.com/vladcto/blagodat/commit/56e96a5e0ce56a198e43919a16f93d82547f84ca) .

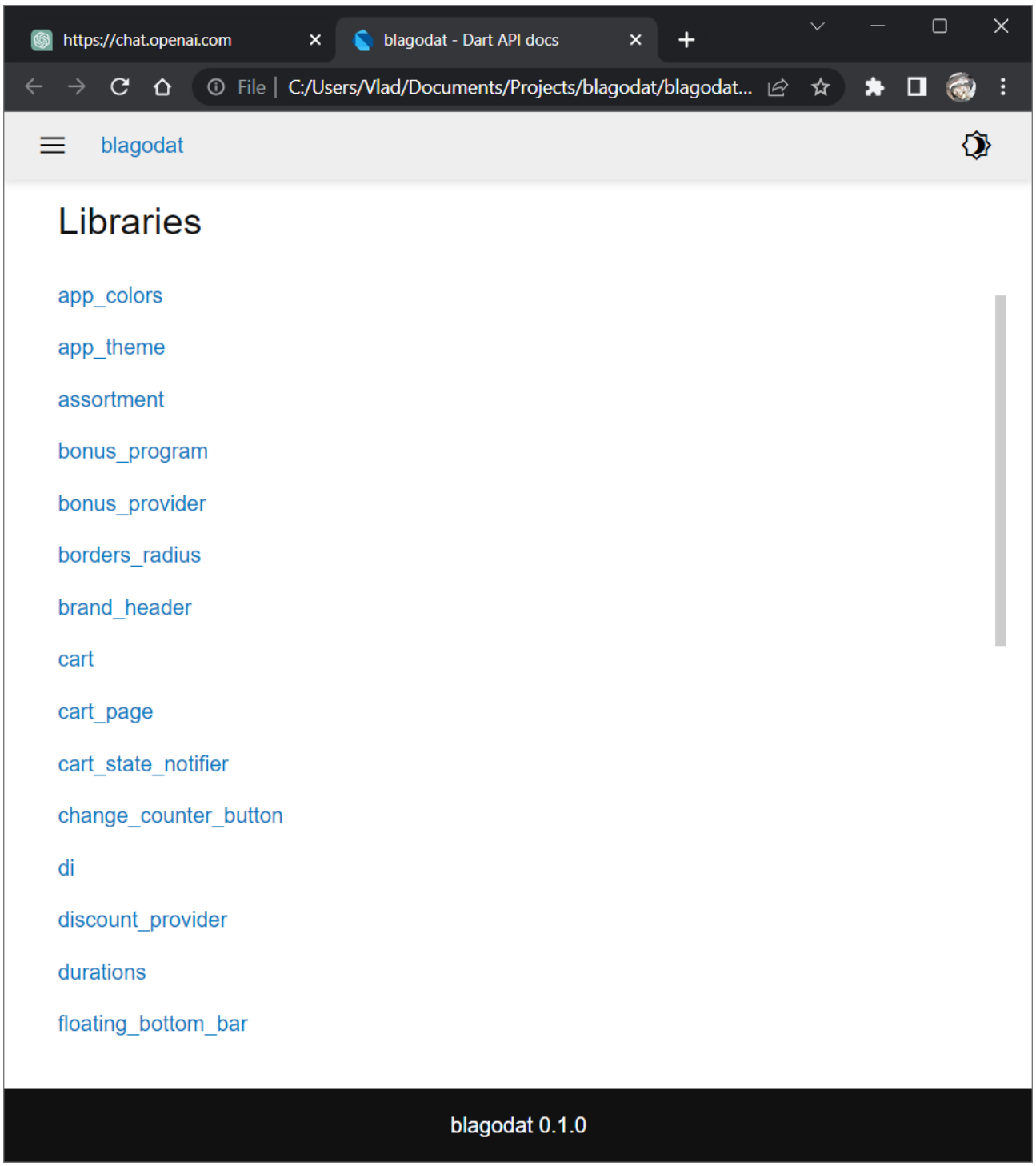

Рисунок 7 – Главная страница документации

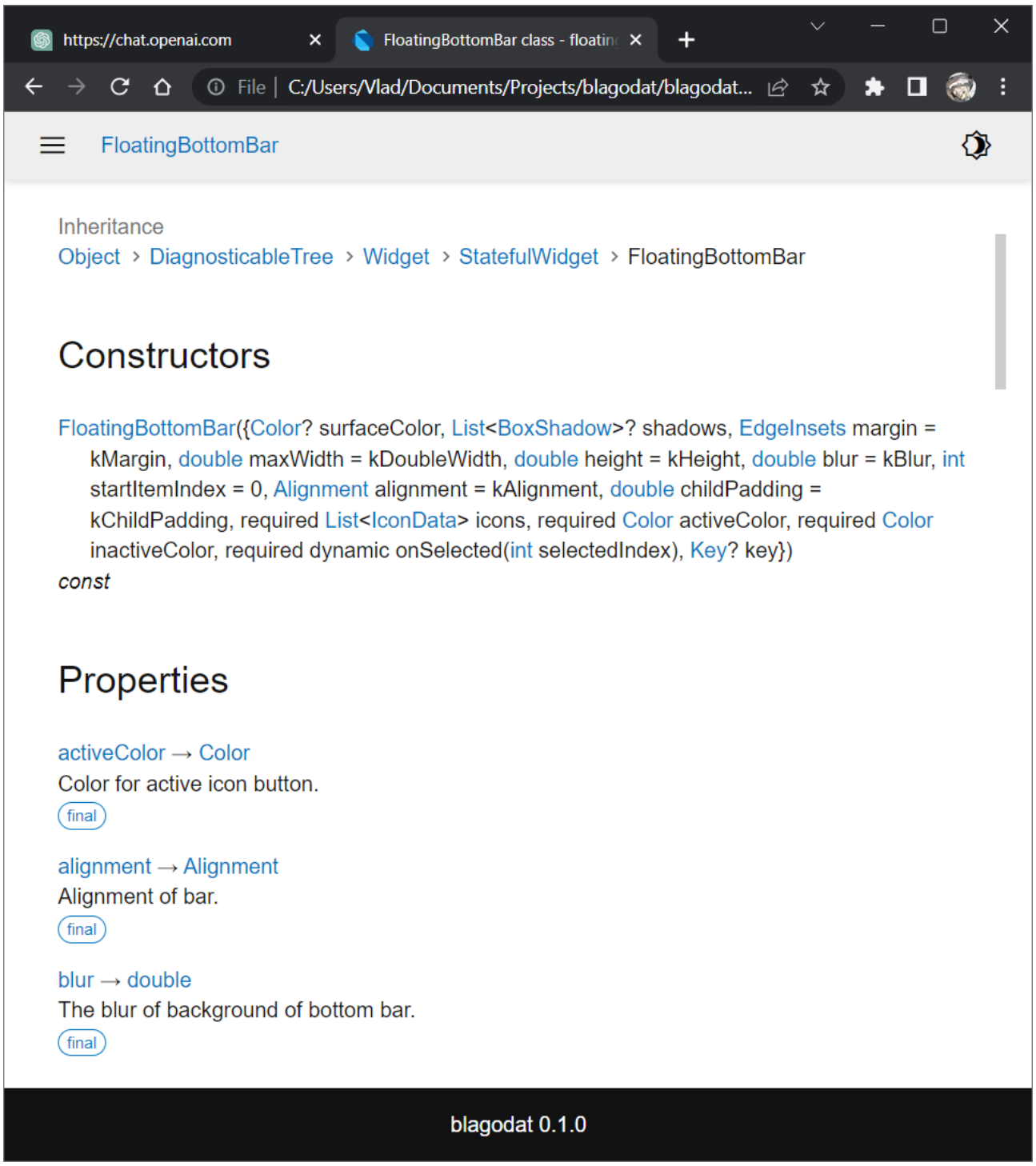

Рисунок 8 – Пример документации

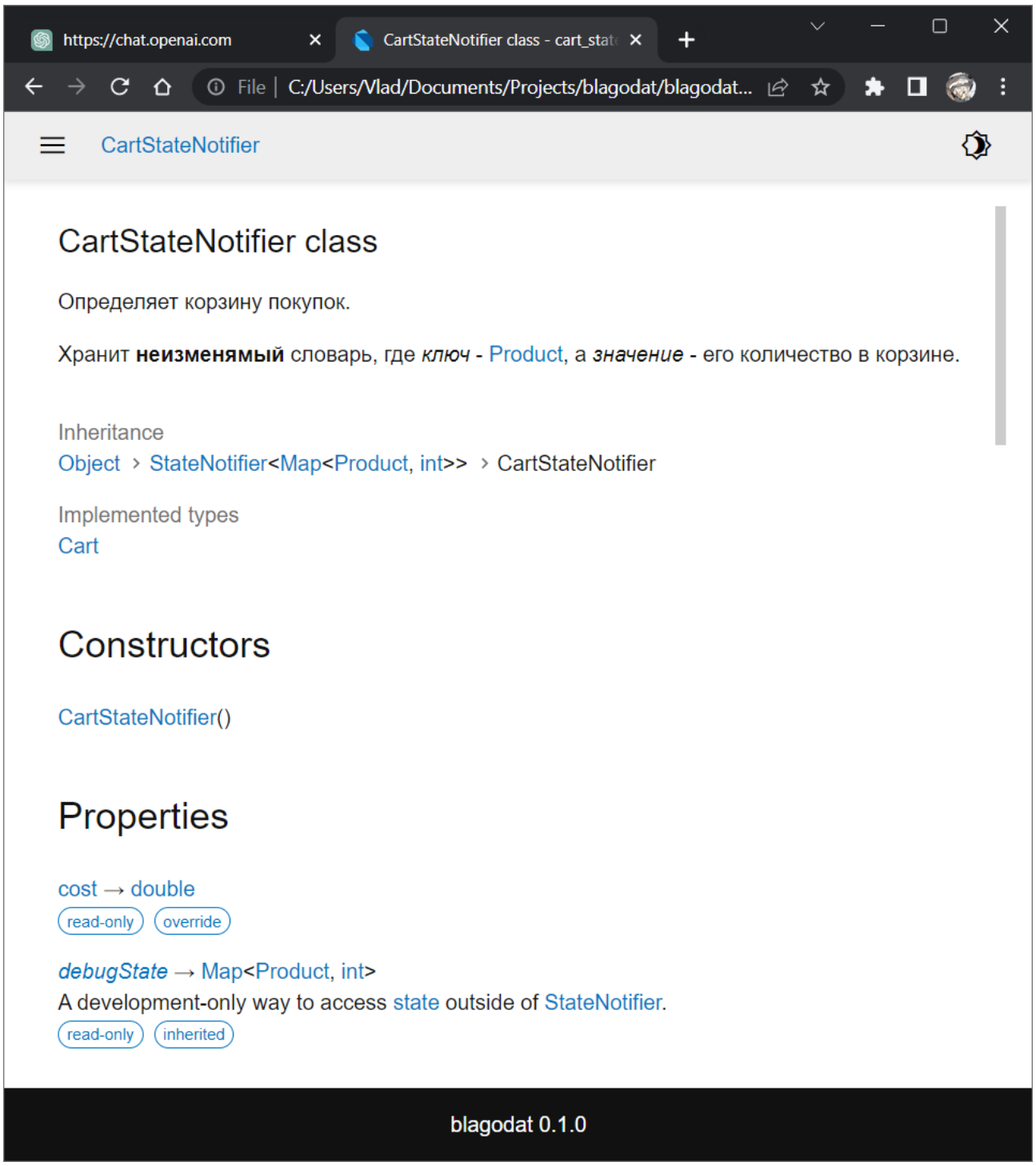

Рисунок 9 – Пример документации

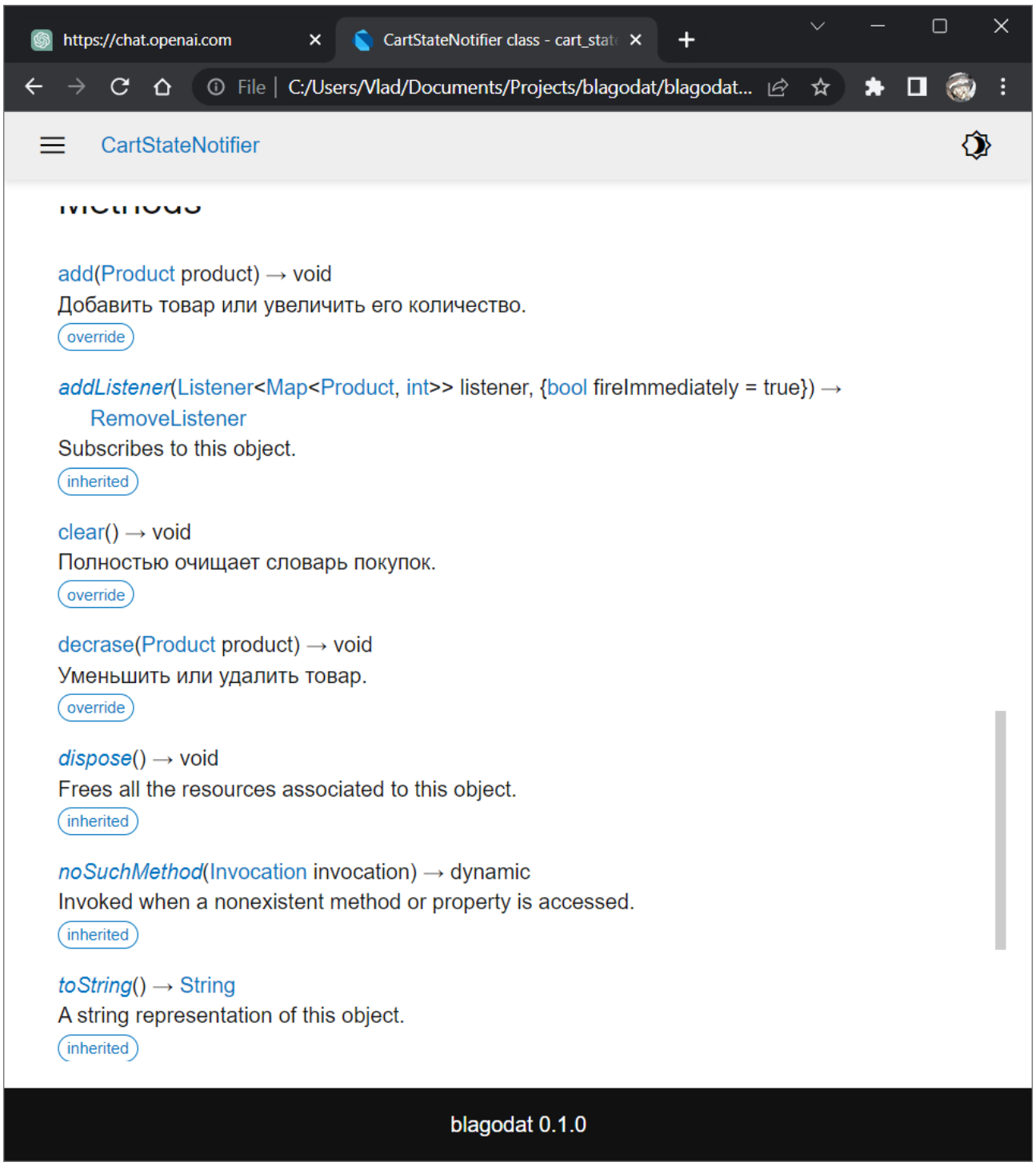

Рисунок 10 – Пример документации

РЕЗУЛЬТАТЫ РАБОТЫ ПРОГРАММЫ С ПРИМЕРАМИ

Проверим работоспособность каждой функции и приложим скриншоты.

Рисунок 11 – Главная страница

Рисунок 12 – Изменение количества

Рисунок 13 – Покупка дополнительных единиц товара.

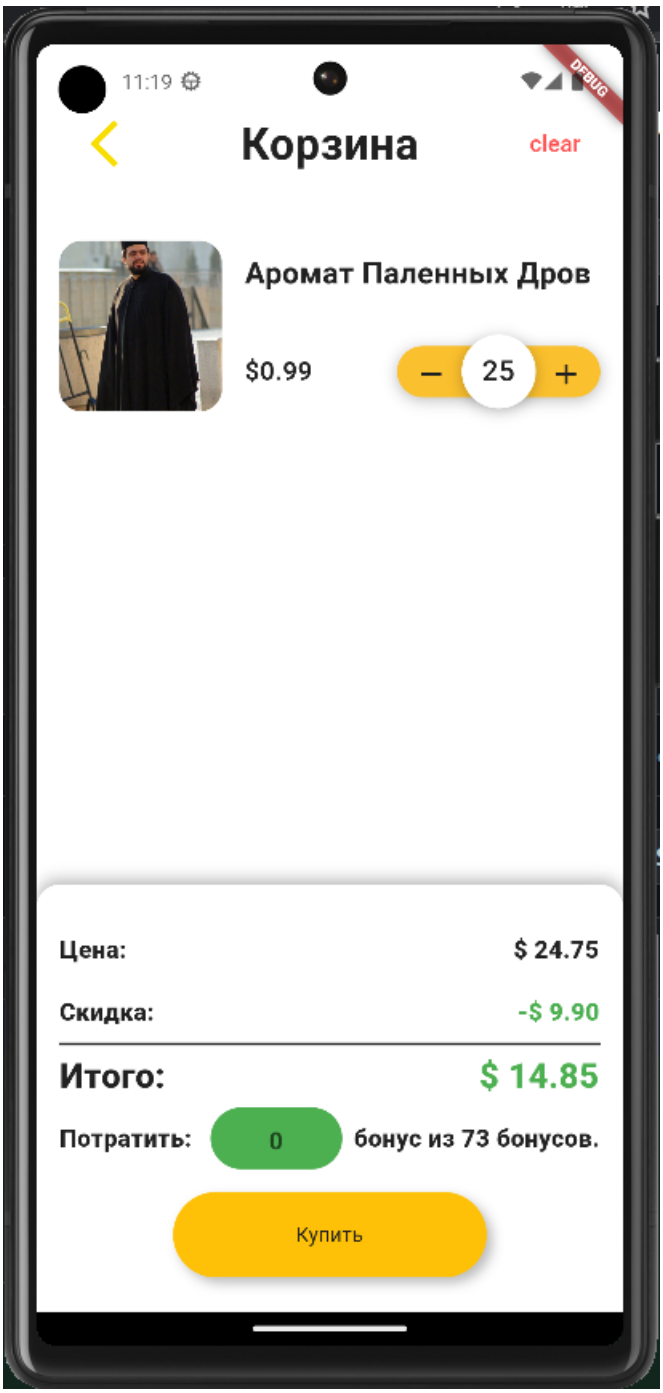

Рисунок 14 – Экран корзины

ЗАКЛЮЧЕНИЕ

В ходе выполнения данной курсовой работы была разработана программа, представляющая собой магазин химических ингредиентов. Программа позволяет пользователям приобретать необходимое количество определенных товаров, взвешивать их и производить оплату. В рамках разработки также была учтена балльная система, которая позволяет пользователям оплачивать покупки частично или полностью баллами, а также предоставляет доступ к истории покупок конкретного клиента.

В процессе работы были представлены технологии и этапы разработки программы. Для обеспечения более понятного использования функционала программы была создана пользовательская документация, которая пошагово объясняет методы работы с программой.

Также была составлена техническая документация, в которой указаны минимальные системные требования для работы с программой. Для более подробного описания системы была представлена её функциональность, а также создана UML-диаграмма, отображающая взаимосвязи основных компонентов кода. В документации были перечислены и описаны элементы алгоритма программы.

Разработанный исходный код был протестирован на наличие возможных ошибок, и результаты тестирования были зафиксированы на снимках экрана.

При написании кода были учтены принципы объектно-ориентированного программирования (ООП) и принципы SOLID. Был также уделен особый внимание качественной архитектуре проекта и использованию различных паттернов программирования.

В итоге выполнения данной курсовой работы были закреплены навыки и знания, полученные в процессе изучения различных технологий программирования.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Habr, Паттерны для новичков: MVC vs MVP vs MVVM, URL: https://habr.com/ru/articles/215605/ (дата обращения 26.05.2023)

2 Software Ideas Modeler, C# Class, Interface, Enum and Other Concepts in UML, URL: https://www.softwareideas.net/csharp-uml-class (дата обращения 26.05.2023)

3 Refactoring guru, Strategy pattern, URL: https://refactoring.guru/design-patterns/ (дата обращения 26.05.2023)

4 Microsoft Learn, Требования к системе для .NET Framework, URL: https://learn.microsoft.com/ru-ru/dotnet/framework/get-started/system-requirements (дата обращения 26.05.2023)

ПРИЛОЖЕНИЕ

main.dart

```
import 'package:blagodat/presentation/pages/main_page.dart';
import 'package:blagodat/presentation/theme/app_theme.dart';
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';

void main() {
  runApp(const MainApp());
}

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

  @override
  Widget build(BuildContext context) {
    return ProviderScope(
      child: MaterialApp(
        theme: AppTheme.lightTheme,
        home: const MainPage(),
      ),
    );
  }
}
```

transaction.dart

```
class Transaction {
  final double totalCost;
  final DateTime purchaseTime;

  Transaction({
    required this.totalCost,
    required this.purchaseTime,
  });
}
```

assortment.dart

```
import 'package:blagodat/data/shop/info/product.dart';

/// Интерфейс, предоставляющий получение списка товаров.
abstract interface class Assortment {
  /// Список товаров доступный в магазине.
  List<Product> get products;
}
```

mock_assortment.dart

```
import 'package:blagodat/data/shop/assortment/assortment.dart';
import 'package:blagodat/data/shop/info/product.dart';

/// Моковый класс для тестовой выборки продуктов.
class MockAssortment implements Assortment {
  static final _products = [
    Product(
      uid: _AutoId.id,
      name: "Аромат Паленных Дров",
      description:
          """ Это благовоние является настоящим сокровищем из затерянного мира Атлантиды.
    Сочетание нот морской соли, драгоценных древесных смол и мистического ладана перенесет вас в глубины древних океанов и пробудит вашу связь с магическими энергиями природы.
Этот аромат позволит вам открыть древние знания и усилить свою интуицию.""",
      imageURL: [

"https://upload.wikimedia.org/wikipedia/commons/6/6b/Eastern_Orthodox_man_in_Jerusalem_by_David_Shankbone.jpg",
        "https://ir.ozone.ru/s3/multimedia-i/wc500/6647844690.jpg",

"https://upload.wikimedia.org/wikipedia/commons/6/6b/Eastern_Orthodox_man_in_Jerusalem_by_David_Shankbone.jpg",

"https://upload.wikimedia.org/wikipedia/commons/6/6b/Eastern_Orthodox_man_in_Jerusalem_by_David_Shankbone.jpg",

"https://upload.wikimedia.org/wikipedia/commons/6/6b/Eastern_Orthodox_man_in_Jerusalem_by_David_Shankbone.jpg",

"https://upload.wikimedia.org/wikipedia/commons/6/6b/Eastern_Orthodox_man_in_Jerusalem_by_David_Shankbone.jpg",

"https://upload.wikimedia.org/wikipedia/commons/6/6b/Eastern_Orthodox_man_in_Jerusalem_by_David_Shankbone.jpg",

"https://upload.wikimedia.org/wikipedia/commons/6/6b/Eastern_Orthodox_man_in_Jerusalem_by_David_Shankbone.jpg",

"https://upload.wikimedia.org/wikipedia/commons/6/6b/Eastern_Orthodox_man_in_Jerusalem_by_David_Shankbone.jpg",

"https://upload.wikimedia.org/wikipedia/commons/6/6b/Eastern_Orthodox_man_in_Jerusalem_by_David_Shankbone.jpg",
      ],
      cost: 0.99,
    ),
    Product(
      uid: _AutoId.id,
```

```
      name: "Аромат Паленных Дров",
      description:
          """ Это благовоние является настоящим сокровищем из затерянного мира Атлантиды.
    Сочетание нот морской соли, драгоценных древесных смол и мистического ладана
перенесет вас в глубины древних океанов и пробудит вашу связь с магическими энергиями
природы.
Этот аромат позволит вам открыть древние знания и усилить свою интуицию.""",
      imageURL: [

"https://upload.wikimedia.org/wikipedia/commons/6/6b/Eastern_Orthodox_man_in_Jerusalem_by_
David_Shankbone.jpg",
        "https://ir.ozone.ru/s3/multimedia-i/wc500/6647844690.jpg",

"https://upload.wikimedia.org/wikipedia/commons/6/6b/Eastern_Orthodox_man_in_Jerusalem_by_
David_Shankbone.jpg",

"https://upload.wikimedia.org/wikipedia/commons/6/6b/Eastern_Orthodox_man_in_Jerusalem_by_
David_Shankbone.jpg",

"https://upload.wikimedia.org/wikipedia/commons/6/6b/Eastern_Orthodox_man_in_Jerusalem_by_
David_Shankbone.jpg",

"https://upload.wikimedia.org/wikipedia/commons/6/6b/Eastern_Orthodox_man_in_Jerusalem_by_
David_Shankbone.jpg",

"https://upload.wikimedia.org/wikipedia/commons/6/6b/Eastern_Orthodox_man_in_Jerusalem_by_
David_Shankbone.jpg",

"https://upload.wikimedia.org/wikipedia/commons/6/6b/Eastern_Orthodox_man_in_Jerusalem_by_
David_Shankbone.jpg",

"https://upload.wikimedia.org/wikipedia/commons/6/6b/Eastern_Orthodox_man_in_Jerusalem_by_
David_Shankbone.jpg",

"https://upload.wikimedia.org/wikipedia/commons/6/6b/Eastern_Orthodox_man_in_Jerusalem_by_
David_Shankbone.jpg",
      ],
      cost: 0.99,
    ),
    Product(
      uid: _AutoId.id,
      name: "Аромат Паленных Дров",
      description:
          """ Это благовоние является настоящим сокровищем из затерянного мира Атлантиды.
    Сочетание нот морской соли, драгоценных древесных смол и мистического ладана
перенесет вас в глубины древних океанов и пробудит вашу связь с магическими энергиями
природы.
Этот аромат позволит вам открыть древние знания и усилить свою интуицию.""",
      imageURL: [

"https://upload.wikimedia.org/wikipedia/commons/6/6b/Eastern_Orthodox_man_in_Jerusalem_by_
David_Shankbone.jpg",
```

```
        "https://ir.ozone.ru/s3/multimedia-i/wc500/6647844690.jpg",

"https://upload.wikimedia.org/wikipedia/commons/6/6b/Eastern_Orthodox_man_in_Jerusalem_by_David_Shankbone.jpg",

"https://upload.wikimedia.org/wikipedia/commons/6/6b/Eastern_Orthodox_man_in_Jerusalem_by_David_Shankbone.jpg",

"https://upload.wikimedia.org/wikipedia/commons/6/6b/Eastern_Orthodox_man_in_Jerusalem_by_David_Shankbone.jpg",

"https://upload.wikimedia.org/wikipedia/commons/6/6b/Eastern_Orthodox_man_in_Jerusalem_by_David_Shankbone.jpg",

"https://upload.wikimedia.org/wikipedia/commons/6/6b/Eastern_Orthodox_man_in_Jerusalem_by_David_Shankbone.jpg",

"https://upload.wikimedia.org/wikipedia/commons/6/6b/Eastern_Orthodox_man_in_Jerusalem_by_David_Shankbone.jpg",

"https://upload.wikimedia.org/wikipedia/commons/6/6b/Eastern_Orthodox_man_in_Jerusalem_by_David_Shankbone.jpg",

"https://upload.wikimedia.org/wikipedia/commons/6/6b/Eastern_Orthodox_man_in_Jerusalem_by_David_Shankbone.jpg",
      ],
      cost: 0.99,
    ),
    Product(
      uid: _AutoId.id,
      name: "Аромат Паленных Дров",
      description:
          """ Это благовоние является настоящим сокровищем из затерянного мира Атлантиды.
    Сочетание нот морской соли, драгоценных древесных смол и мистического ладана перенесет вас в глубины древних океанов и пробудит вашу связь с магическими энергиями природы.
Этот аромат позволит вам открыть древние знания и усилить свою интуицию.""",
      imageURL: [

"https://upload.wikimedia.org/wikipedia/commons/6/6b/Eastern_Orthodox_man_in_Jerusalem_by_David_Shankbone.jpg",
        "https://ir.ozone.ru/s3/multimedia-i/wc500/6647844690.jpg",

"https://upload.wikimedia.org/wikipedia/commons/6/6b/Eastern_Orthodox_man_in_Jerusalem_by_David_Shankbone.jpg",

"https://upload.wikimedia.org/wikipedia/commons/6/6b/Eastern_Orthodox_man_in_Jerusalem_by_David_Shankbone.jpg",

"https://upload.wikimedia.org/wikipedia/commons/6/6b/Eastern_Orthodox_man_in_Jerusalem_by_David_Shankbone.jpg",
```

```
"https://upload.wikimedia.org/wikipedia/commons/6/6b/Eastern_Orthodox_man_in_Jerusalem_by_David_Shankbone.jpg",

"https://upload.wikimedia.org/wikipedia/commons/6/6b/Eastern_Orthodox_man_in_Jerusalem_by_David_Shankbone.jpg",

"https://upload.wikimedia.org/wikipedia/commons/6/6b/Eastern_Orthodox_man_in_Jerusalem_by_David_Shankbone.jpg",

"https://upload.wikimedia.org/wikipedia/commons/6/6b/Eastern_Orthodox_man_in_Jerusalem_by_David_Shankbone.jpg",

"https://upload.wikimedia.org/wikipedia/commons/6/6b/Eastern_Orthodox_man_in_Jerusalem_by_David_Shankbone.jpg",
      ],
      cost: 0.99,
    ),
    Product(
      uid: _AutoId.id,
      name: "Аромат Паленных Дров",
      description:
          """ Это благовоние является настоящим сокровищем из затерянного мира Атлантиды.
    Сочетание нот морской соли, драгоценных древесных смол и мистического ладана перенесет вас в глубины древних океанов и пробудит вашу связь с магическими энергиями природы.
Этот аромат позволит вам открыть древние знания и усилить свою интуицию.""",
      imageURL: [

"https://upload.wikimedia.org/wikipedia/commons/6/6b/Eastern_Orthodox_man_in_Jerusalem_by_David_Shankbone.jpg",
        "https://ir.ozone.ru/s3/multimedia-i/wc500/6647844690.jpg",

"https://upload.wikimedia.org/wikipedia/commons/6/6b/Eastern_Orthodox_man_in_Jerusalem_by_David_Shankbone.jpg",

"https://upload.wikimedia.org/wikipedia/commons/6/6b/Eastern_Orthodox_man_in_Jerusalem_by_David_Shankbone.jpg",

"https://upload.wikimedia.org/wikipedia/commons/6/6b/Eastern_Orthodox_man_in_Jerusalem_by_David_Shankbone.jpg",

"https://upload.wikimedia.org/wikipedia/commons/6/6b/Eastern_Orthodox_man_in_Jerusalem_by_David_Shankbone.jpg",

"https://upload.wikimedia.org/wikipedia/commons/6/6b/Eastern_Orthodox_man_in_Jerusalem_by_David_Shankbone.jpg",

"https://upload.wikimedia.org/wikipedia/commons/6/6b/Eastern_Orthodox_man_in_Jerusalem_by_David_Shankbone.jpg",
```

```
"https://upload.wikimedia.org/wikipedia/commons/6/6b/Eastern_Orthodox_man_in_Jerusalem_by_David_Shankbone.jpg",

"https://upload.wikimedia.org/wikipedia/commons/6/6b/Eastern_Orthodox_man_in_Jerusalem_by_David_Shankbone.jpg",
      ],
      cost: 0.99,
    ),
  ];

  @override
  List<Product> get products => _products;
}

/// Класс предоставляющий сервис для получения UID.
class _AutoId {
  static int _i = 0;

  static String get id {
    return (_i++).toString();
  }
}
```

product.dart

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

part 'product.freezed.dart';

/// Вся информация о товаре в магазине.
@freezed
class Product with _$Product {
  const Product._();

  @Assert('cost > 0')
  @Assert('name.isNotEmpty')
  @Assert('imageURL.length > 0')
  factory Product({
    /// Уникальный ID продукта.
    required String uid,

    /// Название продукта.
    required String name,

    /// Описание продукта.
    required String description,

    /// Веб-ссылки на изображение продукта.
    ///
    /// Должно быть хотя-бы одно изображение, которое будет являться основным.
```

```
    required List<String> imageURL,

    /// Цена продукта в долларах.
    required double cost,
  }) = _Product;

  @override
  int get hashCode => uid.hashCode;

  @override
  bool operator ==(covariant Product other) => other.uid == uid;
}
```

product.freezed.dart

```
// coverage:ignore-file
// GENERATED CODE - DO NOT MODIFY BY HAND
// ignore_for_file: type=lint
// ignore_for_file: unused_element, deprecated_member_use,
deprecated_member_use_from_same_package, use_function_type_syntax_for_parameters,
unnecessary_const, avoid_init_to_null, invalid_override_different_default_values_named,
prefer_expression_function_bodies, annotate_overrides, invalid_annotation_target,
unnecessary_question_mark

part of 'product.dart';

// **************************************************************************
// FreezedGenerator
// **************************************************************************

T _$identity<T>(T value) => value;

final _privateConstructorUsedError = UnsupportedError(
    'It seems like you constructed your class using `MyClass._()`. This constructor is only meant to
be used by freezed and you are not supposed to need it nor use it.\nPlease check the documentation
here for more information: https://github.com/rrousselGit/freezed#custom-getters-and-methods');

/// @nodoc
mixin _$Product {
  /// Уникальный ID продукта.
  String get uid => throw _privateConstructorUsedError;

  /// Название продукта.
  String get name => throw _privateConstructorUsedError;

  /// Описание продукта.
  String get description => throw _privateConstructorUsedError;

  /// Веб-ссылки на изображение продукта.
  ///
  /// Должно быть хотя-бы одно изображение, которое будет являться основным.
```

```
  List<String> get imageURL => throw _privateConstructorUsedError;

  /// Цена продукта в долларах.
  double get cost => throw _privateConstructorUsedError;

  @JsonKey(ignore: true)
  $ProductCopyWith<Product> get copyWith => throw _privateConstructorUsedError;
}

/// @nodoc
abstract class $ProductCopyWith<$Res> {
  factory $ProductCopyWith(Product value, $Res Function(Product) then) =
      _$ProductCopyWithImpl<$Res, Product>;
  @useResult
  $Res call(
      {String uid,
      String name,
      String description,
      List<String> imageURL,
      double cost});
}

/// @nodoc
class _$ProductCopyWithImpl<$Res, $Val extends Product>
    implements $ProductCopyWith<$Res> {
  _$ProductCopyWithImpl(this._value, this._then);

  // ignore: unused_field
  final $Val _value;
  // ignore: unused_field
  final $Res Function($Val) _then;

  @pragma('vm:prefer-inline')
  @override
  $Res call({
    Object? uid = null,
    Object? name = null,
    Object? description = null,
    Object? imageURL = null,
    Object? cost = null,
  }) {
    return _then(_value.copyWith(
      uid: null == uid
          ? _value.uid
          : uid // ignore: cast_nullable_to_non_nullable
              as String,
      name: null == name
          ? _value.name
          : name // ignore: cast_nullable_to_non_nullable
              as String,
      description: null == description
          ? _value.description
```

```
        : description // ignore: cast_nullable_to_non_nullable
            as String,
      imageURL: null == imageURL
          ? _value.imageURL
          : imageURL // ignore: cast_nullable_to_non_nullable
              as List<String>,
      cost: null == cost
          ? _value.cost
          : cost // ignore: cast_nullable_to_non_nullable
              as double,
    ) as $Val);
  }
}

/// @nodoc
abstract class _$$_ProductCopyWith<$Res> implements $ProductCopyWith<$Res> {
  factory _$$_ProductCopyWith(
          _$_Product value, $Res Function(_$_Product) then) =
      __$$_ProductCopyWithImpl<$Res>;
  @override
  @useResult
  $Res call(
      {String uid,
      String name,
      String description,
      List<String> imageURL,
      double cost});
}

/// @nodoc
class __$$_ProductCopyWithImpl<$Res>
    extends _$ProductCopyWithImpl<$Res, _$_Product>
    implements _$$_ProductCopyWith<$Res> {
  __$$_ProductCopyWithImpl(_$_Product _value, $Res Function(_$_Product) _then)
      : super(_value, _then);

  @pragma('vm:prefer-inline')
  @override
  $Res call({
    Object? uid = null,
    Object? name = null,
    Object? description = null,
    Object? imageURL = null,
    Object? cost = null,
  }) {
    return _then(_$_Product(
      uid: null == uid
          ? _value.uid
          : uid // ignore: cast_nullable_to_non_nullable
              as String,
      name: null == name
          ? _value.name
```

```
        : name // ignore: cast_nullable_to_non_nullable
            as String,
      description: null == description
          ? _value.description
          : description // ignore: cast_nullable_to_non_nullable
            as String,
      imageURL: null == imageURL
          ? _value._imageURL
          : imageURL // ignore: cast_nullable_to_non_nullable
            as List<String>,
      cost: null == cost
          ? _value.cost
          : cost // ignore: cast_nullable_to_non_nullable
            as double,
    ));
  }
}

/// @nodoc

class _$_Product extends _Product {
  _$_Product(
      {required this.uid,
      required this.name,
      required this.description,
      required final List<String> imageURL,
      required this.cost})
      : assert(cost > 0),
        assert(name.isNotEmpty),
        assert(imageURL.length > 0),
        _imageURL = imageURL,
        super._();

  /// Уникальный ID продукта.
  @override
  final String uid;

  /// Название продукта.
  @override
  final String name;

  /// Описание продукта.
  @override
  final String description;

  /// Веб-ссылки на изображение продукта.
  ///
  /// Должно быть хотя-бы одно изображение, которое будет являться основным.
  final List<String> _imageURL;

  /// Веб-ссылки на изображение продукта.
  ///
```

```
  /// Должно быть хотя-бы одно изображение, которое будет являться основным.
  @override
  List<String> get imageURL {
    if (_imageURL is EqualUnmodifiableListView) return _imageURL;
    // ignore: implicit_dynamic_type
    return EqualUnmodifiableListView(_imageURL);
  }

  /// Цена продукта в долларах.
  @override
  final double cost;

  @override
  String toString() {
    return 'Product(uid: $uid, name: $name, description: $description, imageURL: $imageURL, cost:
$cost)';
  }

  @JsonKey(ignore: true)
  @override
  @pragma('vm:prefer-inline')
  _$$_ProductCopyWith<_$_Product> get copyWith =>
      __$$_ProductCopyWithImpl<_$_Product>(this, _$identity);
}

abstract class _Product extends Product {
  factory _Product(
      {required final String uid,
      required final String name,
      required final String description,
      required final List<String> imageURL,
      required final double cost}) = _$_Product;
  _Product._() : super._();

  @override

  /// Уникальный ID продукта.
  String get uid;
  @override

  /// Название продукта.
  String get name;
  @override

  /// Описание продукта.
  String get description;
  @override

  /// Веб-ссылки на изображение продукта.
  ///
  /// Должно быть хотя-бы одно изображение, которое будет являться основным.
  List<String> get imageURL;
```

```
  @override

  /// Цена продукта в долларах.
  double get cost;
  @override
  @JsonKey(ignore: true)
  _$$_ProductCopyWith<_$_Product> get copyWith =>
      throw _privateConstructorUsedError;
}
```

di.dart

```
import 'package:blagodat/data/shop/assortment/assortment.dart';
import 'package:blagodat/data/shop/assortment/mock_assortment.dart';
import 'package:blagodat/data/shop/info/product.dart';
import 'package:blagodat/domain/discount/bonus_program.dart';
import 'package:blagodat/domain/discount/bonus_provider.dart';
import 'package:blagodat/domain/discount/discount_provider.dart';
import 'package:blagodat/domain/cart/cart_state_notifier.dart';
import 'package:blagodat/domain/shop/purchase_manager.dart';
import 'package:blagodat/data/shop/transaction.dart';
import 'package:blagodat/domain/shop/transaction_history.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';

final assortmentProvider = Provider<Assortment>((ref) => MockAssortment());

final cartProvider =
    StateNotifierProvider<CartStateNotifier, Map<Product, int>>(
  (ref) => CartStateNotifier(),
);

final purchaseManager = Provider<PurchaseManager>(
  (ref) {
    final cart = ref.watch(cartProvider.notifier);
    final bonusProgram = ref.watch(bonusProvider);
    final discount = ref.watch(discountProvider);
    final manager = PurchaseManager(cart, discount, bonusProgram);
    return manager;
  },
);

final streamPurchaseProvider = Provider<Stream<Transaction>>(
  (ref) {
    return ref.watch(purchaseManager).transactionStream;
  },
);

final _bonusProgramProvider = Provider<BonusProgram>(
  (ref) => BonusProgram(),
);
```

```
final historyProvider =
    StateNotifierProvider<TransactionHistoryNotfier, List<Transaction>>(
  (ref) {
    final transactionHistory = TransactionHistoryNotfier();
    ref.read(streamPurchaseProvider).listen(
      (value) {
        transactionHistory.add(value);
      },
    );
    return transactionHistory;
  },
);

final Provider<DiscountProvider> discountProvider = _bonusProgramProvider;

final Provider<BonusProvider> bonusProvider = _bonusProgramProvider;
```

cart.dart

```
import 'package:blagodat/data/shop/info/product.dart';

abstract interface class Cart {
  void decrase(Product product);
  void add(Product product);
  void clear();
  double get cost;
  bool get isNotEmpty;
}
```

cart_state_notifier.dart

```
import 'dart:collection';

import 'package:blagodat/data/shop/info/product.dart';
import 'package:blagodat/domain/cart/cart.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';

/// Определяет корзину покупок.
///
/// Хранит **неизменямый** словарь,
/// где *ключ* - [Product], а *значение* - его количество в корзине.
class CartStateNotifier extends StateNotifier<Map<Product, int>>
    implements Cart {
  final Map<Product, int> _cartMap;

  CartStateNotifier()
      : _cartMap = {},
        super(UnmodifiableMapView({}));

  /// Добавить товар или увеличить его количество.
```

```
  ///
  /// Если товар **отсуствует** - делает его количество = 0.
  /// Если товар **присуствует** - прибавляет к его значению +1.
  @override
  void add(Product product) {
    _cartMap[product] = (_cartMap[product] ?? 0) + 1;
    state = UnmodifiableMapView(_cartMap);
  }

  /// Уменьшить или удалить товар.
  ///
  /// Если товара **больше 1** - уменьшает его количество на 1.
  /// Если товара **одна штука** - удаляет его из словаря.
  ///
  /// Если товар **отсутствует** - ничего не делает.
  @override
  void decrase(Product product) {
    // Cant decrase so skip.
    if (_cartMap[product] == null) return;

    final newValue = _cartMap[product]! - 1;
    // if need remove
    if (newValue == 0) {
      _cartMap.remove(product);
    } else {
      _cartMap[product] = newValue;
    }
    state = UnmodifiableMapView(_cartMap);
  }

  /// Полностью очищает словарь покупок.
  @override
  void clear() {
    _cartMap.clear();
    state = UnmodifiableMapView(_cartMap);
  }

  @override
  double get cost {
    return _cartMap.entries.fold(
      0,
      (previousValue, element) =>
          previousValue + element.key.cost * element.value,
    );
  }

  @override
  bool get isNotEmpty => _cartMap.isNotEmpty;
}
```

bonus_program.dart

```
import 'package:blagodat/domain/discount/bonus_provider.dart';
import 'package:blagodat/domain/discount/discount_provider.dart';
import 'package:blagodat/data/shop/transaction.dart';

class BonusProgram implements DiscountProvider, BonusProvider {
  int _bonus = 100;

  @override
  double get discount => 0.6;

  @override
  int get bonus => _bonus;

  /// Зарегестрировать транзакцию в бонусной программе.
  @override
  void performTransaction(Transaction transaction) {
    _bonus += (transaction.totalCost * 0.1).toInt();
  }

  /// Вычитает из текущего пула бонусов [spendBonus].
  ///
  /// Если количество бонусов для траты больше,
  @override
  void spendBonus(int spendBonus) {
    print(_bonus);
    if (_bonus < spendBonus) {
      throw Exception(
          "Количество бонусов для траты больше, чем бонусов в пуле.");
    }
    _bonus -= spendBonus;
  }
}
```

bonus_provider.dart

```
import 'package:blagodat/data/shop/transaction.dart';

abstract interface class BonusProvider {
  int get bonus;
  void spendBonus(int spendBonus);
  void performTransaction(Transaction transaction);
}
```

discount_provider.dart

```
abstract interface class DiscountProvider {
  double get discount;
}
```

purchase_manager.dart

```
import 'dart:async';
import 'dart:math';
import 'package:blagodat/domain/cart/cart.dart';
import 'package:blagodat/domain/discount/bonus_provider.dart';
import 'package:blagodat/data/shop/transaction.dart';
import 'package:blagodat/domain/discount/discount_provider.dart';

/// Менеджер оформления покупок.
class PurchaseManager {
  final Cart _cart;
  final DiscountProvider _discountProvider;
  final BonusProvider bonusProvider;

  final StreamController<Transaction> _transactionStreamController =
      StreamController<Transaction>();

  PurchaseManager(this._cart, this._discountProvider, this.bonusProvider);

  /// Поток успешных покупок.
  Stream<Transaction> get transactionStream =>
      _transactionStreamController.stream;

  /// Покупает корзину пользователя, с помощью наличных [cash] и бонусами [bonus].
  ///
  /// Возвращает true если покупка успешная.
  bool purchase(double cash, int bonus) {
    if (!_cart.isNotEmpty) return false;
    // Может ли пользоваетель позволить себе покупку.
    double totalCost = _cart.cost * _discountProvider.discount;
    bonus = min(totalCost.ceil().toInt(), bonus);
    if (cash + bonus < totalCost) return false;
    final transaction = Transaction(
      totalCost: totalCost,
      purchaseTime: DateTime.now(),
    );
    _transactionStreamController.add(transaction);
    bonusProvider.performTransaction(transaction);
    bonusProvider.spendBonus(bonus);
    _cart.clear();
    return true;
  }

  void dispose() {
    _transactionStreamController.close();
  }
}
```

transaction_history.dart

```
import 'package:blagodat/data/shop/transaction.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';

class TransactionHistoryNotfier extends StateNotifier<List<Transaction>> {
  TransactionHistoryNotfier() : super([]);

  void add(Transaction transaction) {
    state = [...state, transaction];
  }
}
```

borders_radius.dart

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

sealed class BordersRadius {
  static final standart = BorderRadius.circular(16);
}
```

durations.dart

```
sealed class Durations {
  static const standart = Duration(milliseconds: 100);
}
```

font_sizes.dart

```
sealed class FontSize {
  static const double header = 28;
  static const double large = 26;
  static const double medium = 20;
  static const double small = 16;
}
```

paddings.dart

```
sealed class Paddings {
  static const double maximum = 32;
  static const double large = 24;
  static const double medium = 16;
  static const double small = 8;
}
```

shadows.dart

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

```
sealed class Shadows {
  static const standart = BoxShadow(
    offset: Offset(4, 4),
    blurRadius: 6,
    color: Colors.black26,
  );

  static const topDown = BoxShadow(
    offset: Offset(0, 2),
    blurRadius: 8,
    color: Colors.black26,
  );
}
```

main_page.dart

```
import 'package:blagodat/presentation/constants/paddings.dart';
import 'package:blagodat/presentation/pages/fragments/history_fragment.dart';
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';

import '../widgets/floating_bottom_bar.dart';
import 'home/home_fragment.dart';
import 'fragments/loyalty_fragment.dart';

final _pageIndexProvider = StateProvider<int>((ref) => 0);

/// The page that is displayed when the application starts.
class MainPage extends ConsumerWidget {
  /// Bottom padding included floating bar.
  static const double pageBarPadding =
      barHeight + barElevation + Paddings.medium;
  static const double barHeight = 76;
  static const double barElevation = 22;
  static const double barSideMargin = 64;
  const MainPage({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context, WidgetRef ref) {
    return Scaffold(
      extendBody: true,
      bottomNavigationBar: FloatingBottomBar(
        activeColor: Colors.yellow,
        inactiveColor: Colors.white,
        icons: const [
          Icons.home,
          Icons.loyalty,
          Icons.history,
        ],
        margin: const EdgeInsets.only(
```

```
          left: barSideMargin,
          right: barSideMargin,
          bottom: barElevation,
        ),
        height: barHeight,
        maxWidth: 500,
        onSelected: (i) => ref.read(_pageIndexProvider.notifier).state = i,
        surfaceColor: Colors.black.withOpacity(0.7),
      ),
      body: IndexedStack(
        index: ref.watch(_pageIndexProvider),
        children: const [
          HomeFragment(),
          LoyaltyFragment(),
          HistoryFragment(),
        ],
      ),
    );
  }
}
```

product_preview_page.dart

```
import 'package:blagodat/data/shop/info/product.dart';
import 'package:blagodat/domain/di.dart';
import 'package:blagodat/presentation/constants/borders_radius.dart';
import 'package:blagodat/presentation/constants/durations.dart';
import 'package:blagodat/presentation/constants/font_sizes.dart';
import 'package:blagodat/presentation/constants/paddings.dart';
import 'package:blagodat/presentation/constants/shadows.dart';
import 'package:blagodat/presentation/widgets/change_counter_button.dart';
import 'package:blagodat/presentation/widgets/sub_page.dart';
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';

/// Инднекс выбранного [product.imageUrl] для показа.
final _selectedImageIndex = StateProvider((ref) => 0);

/// Странца, отображающая информацию о товаре.
class ProductPreviewPage extends ConsumerWidget {
  static const String title = "Просмотр";
  static const mainImageRatio = 360 / 290;
  static const smallCardSize = 90.0;

  final Product product;
  const ProductPreviewPage({Key? key, required this.product}) : super(key: key);

  @override
  Widget build(BuildContext context, WidgetRef ref) {
    final cartCount = ref.watch(cartProvider)[product] ?? 0;
```

```
return SubPage(
  title: title,
  body: Column(
    children: [
      // Main image
      Padding(
        padding: const EdgeInsets.symmetric(horizontal: Paddings.medium),
        child: AspectRatio(
          aspectRatio: mainImageRatio,
          child: DecoratedBox(
            decoration: BoxDecoration(
              borderRadius: BordersRadius.standart,
              boxShadow: const [Shadows.standart],
            ),
            child: ClipRRect(
              borderRadius: BordersRadius.standart,
              child: Consumer(
                builder: (_, ref, child) {
                  return Image.network(
                    product.imageURL[ref.watch(_selectedImageIndex)],
                    fit: BoxFit.cover,
                  );
                },
              ),
            ),
          ),
        ),
      ),
      const SizedBox(height: Paddings.medium),
      // Additional images
      SizedBox(
        height: smallCardSize,
        child: _ProductImageListView(product.imageURL),
      ),
      const SizedBox(height: Paddings.large),
      // Product name.
      Padding(
        padding: const EdgeInsets.symmetric(horizontal: Paddings.medium),
        child: Align(
          alignment: Alignment.centerLeft,
          child: Text(
            product.name,
            maxLines: 1,
            style: const TextStyle(
              fontWeight: FontWeight.w700,
              fontSize: FontSize.large,
            ),
          ),
        ),
      ),
      // Product description.
      Expanded(
```

```
            child: Padding(
              padding: const EdgeInsets.symmetric(horizontal: Paddings.medium),
              child: SingleChildScrollView(
                child: Text(
                  product.description,
                  textAlign: TextAlign.justify,
                  style: const TextStyle(
                    fontSize: FontSize.small,
                    height: 1.1,
                  ),
                ),
              ),
            ),
          ),
          SizedBox(
            height: 100,
            child: _ControlsRow(product),
          ),
        ],
      ),
    );
  }
}

class _SmallImageCard extends StatelessWidget {
  static const double minScale = 0.95;
  static const double maxScale = 1.05;
  final String imageUrl;
  final bool selected;

  const _SmallImageCard(this.imageUrl, this.selected);

  @override
  Widget build(BuildContext context) {
    return AnimatedScale(
      scale: selected ? maxScale : minScale,
      duration: Durations.standart,
      child: AspectRatio(
        aspectRatio: 1.0,
        child: DecoratedBox(
          decoration: BoxDecoration(
            borderRadius: BordersRadius.standart,
            boxShadow: const [Shadows.standart],
          ),
          child: ClipRRect(
            borderRadius: BordersRadius.standart,
            child: Image.network(
              imageUrl,
              fit: BoxFit.cover,
            ),
          ),
        ),
```

```
      ),
    );
  }
}

class _ProductImageListView extends ConsumerWidget {
  final List<String> imageUrl;
  const _ProductImageListView(this.imageUrl);

  @override
  Widget build(BuildContext context, WidgetRef ref) {
    final int selectedUrlIndex = ref.watch(_selectedImageIndex);
    final productChilds = [
      for (int i = 0; i < imageUrl.length; i++)
        GestureDetector(
          onTap: () => ref.read(_selectedImageIndex.notifier).state = i,
          child: _SmallImageCard(
            imageUrl[i],
            selectedUrlIndex == i,
          ),
        ),
    ];

    return ListView.builder(
      // Images and space btw them.
      itemCount: imageUrl.length * 2 - 1,
      scrollDirection: Axis.horizontal,
      clipBehavior: Clip.none,
      padding: const EdgeInsets.symmetric(horizontal: Paddings.medium),
      // Create images card and space.
      itemBuilder: (_, i) => i % 2 == 0
          ? productChilds[i ~/ 2]
          : const SizedBox(width: Paddings.small),
    );
  }
}

/// Виджет, который отображает цену и кнопку добавления в корзину.
class _ControlsRow extends ConsumerWidget {
  final Product product;
  const _ControlsRow(this.product);

  @override
  Widget build(BuildContext context, WidgetRef ref) {
    return Padding(
      padding: const EdgeInsets.only(
        left: Paddings.medium,
        right: Paddings.medium,
        bottom: Paddings.large,
        top: Paddings.small / 2,
      ),
      child: Row(
```

```
        children: [
          Expanded(
            flex: 3,
            child: Padding(
              // TODO: Вынести в константы и соотнести отсутп с [ChangeCounterButton].
              padding: const EdgeInsets.symmetric(vertical: 10),
              child: Align(
                alignment: Alignment.topLeft,
                // TODO: Не хардкодить размер текста, а расширять его по мере.
                // TODO: Вынести жирность шрифта в константы.
                child: Text(
                  "\$ ${product.cost.toStringAsFixed(2)}",
                  textAlign: TextAlign.left,
                  style: const TextStyle(
                    fontWeight: FontWeight.w700,
                    fontSize: 36,
                  ),
                ),
              ),
            ),
          ),
          const Expanded(
            flex: 1,
            child: SizedBox.shrink(),
          ),
          Expanded(
            flex: 4,
            child: Align(
              alignment: Alignment.centerRight,
              child: ChangeCounterButton(
                zeroTitle: "Добавить",
                counter: ref.watch(cartProvider)[product] ?? 0,
                increase: () => ref.watch(cartProvider.notifier).add(product),
                decrase: () =>
                    ref.watch(cartProvider.notifier).decrase(product),
              ),
            ),
          ),
        ],
      ),
    );
  }
}
```

cart_page.dart

```
import 'package:blagodat/domain/di.dart';
import 'package:blagodat/domain/shop/purchase_manager.dart';
import 'package:blagodat/presentation/constants/borders_radius.dart';
import 'package:blagodat/presentation/constants/font_sizes.dart';
import 'package:blagodat/presentation/constants/paddings.dart';
```

```
import 'package:blagodat/presentation/constants/shadows.dart';
import 'package:blagodat/presentation/pages/cart/mock_card_payment.dart';
import 'package:blagodat/presentation/theme/app_colors.dart';
import 'package:blagodat/presentation/widgets/product_in_cart_card.dart';
import 'package:blagodat/presentation/widgets/row_end_start.dart';
import 'package:blagodat/presentation/widgets/sub_page.dart';
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';

class CartPage extends ConsumerWidget {
  static const String title = "Корзина";

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

  @override
  Widget build(BuildContext context, WidgetRef ref) {
    final products = ref.watch(cartProvider).keys;

    final cartChilds = products
        .map(
          (e) => SizedBox(
            height: 120,
            child: ProductInCartCard(product: e),
          ),
        )
        .toList();

    return SubPage(
      title: title,
      trailing: TextButton(
        onPressed: () => ref.read(cartProvider.notifier).clear(),
        child: const Text(
          "clear",
          style: TextStyle(
            color: Colors.redAccent,
            fontSize: FontSize.small,
            fontWeight: FontWeight.w500,
          ),
        ),
      ),
      body: Stack(
        children: [
          ListView(
            padding: const EdgeInsets.only(
              left: Paddings.medium,
              right: Paddings.medium,
              top: Paddings.medium,
              bottom: 320,
            ),
            children: cartChilds,
          ),
          Positioned(
```

```
            left: 0,
            right: 0,
            bottom: 0,
            height: 300,
            child: Container(
              padding: const EdgeInsets.symmetric(horizontal: Paddings.medium),
              decoration: const BoxDecoration(
                color: AppColors.white,
                borderRadius: BorderRadius.vertical(
                  top: Radius.circular(16),
                ),
                boxShadow: [
                  BoxShadow(
                    color: Colors.black26,
                    offset: Offset(0, -4),
                    blurRadius: 8,
                  ),
                ],
              ),
              child: const _CartCostContainer(),
            ),
          ),
        ],
      ),
    );
  }
}

final _chooseBonus = StateProvider<int>((ref) => 0);

class _CartCostContainer extends ConsumerWidget {
  const _CartCostContainer();

  @override
  Widget build(BuildContext context, WidgetRef ref) {
    ref.watch(cartProvider);
    final cost = ref.watch(cartProvider.notifier).cost;
    final discountCost = (1 - ref.watch(discountProvider).discount) * cost;
    final total = ref.watch(discountProvider).discount * cost;
    final purchase = ref.watch(purchaseManager);
    final bonusSpend = ref.watch(_chooseBonus);

    return Column(
      children: [
        const SizedBox(height: Paddings.large),
        Expanded(
          child: RowEndStart(
            start: const _BoldText("Цена:"),
            end: _BoldText("\$ ${cost.toStringAsFixed(2)}"),
          ),
        ),
        Expanded(
```

```
  child: RowEndStart(
    start: const _BoldText("Скидка:"),
    end: _BoldText(
      "-\$ ${discountCost.toStringAsFixed(2)}",
      color: Colors.green,
    ),
  ),
),
const SizedBox(
  height: 2,
  width: double.infinity,
  child: ColoredBox(color: AppColors.softDark),
),
Expanded(
  child: RowEndStart(
    start: const _BoldText(
      "Итого:",
      fontSize: 24,
    ),
    end: _BoldText(
      "\$ ${total.toStringAsFixed(2)}",
      fontSize: 24,
      color: Colors.green,
    ),
  ),
),
// Трата бонусов.
Expanded(
  child: Row(
    children: [
      _BoldText("Потратить: "),
      Expanded(
        child: _BonusField(),
      ),
      _BoldText("бонус из ${ref.watch(bonusProvider).bonus} бонусов.")
    ],
  ),
),
const SizedBox(height: Paddings.medium),
Center(
  child: SizedBox(
    height: 60,
    width: 220,
    child: GestureDetector(
      onTap: () => showCartPayment(
        context,
        purchase,
        bonusSpend,
      ),
      child: Container(
        alignment: Alignment.center,
        decoration: BoxDecoration(
```

```
                borderRadius: BorderRadius.circular(10000),
                color: AppColors.mainOrange,
                boxShadow: const [Shadows.standart],
              ),
              child: Text("Купить"),
            ),
          ),
        ),
      ),
      const SizedBox(height: Paddings.large),
    ],
  );
 }

 void showCartPayment(
    BuildContext context, PurchaseManager manager, int bonusSpend) {
  showDialog(
    context: context,
    builder: (_) => MockCardPayment(
      onPayment: (cash) {
        bool result = manager.purchase(cash, bonusSpend);
        if (result) {
          ScaffoldMessenger.of(context).showSnackBar(
            const SnackBar(
              content: Text("Оплата успешна."),
            ),
          );
        } else {
          ScaffoldMessenger.of(context).showSnackBar(
            const SnackBar(
              duration: Duration(milliseconds: 500),
              content: Text("Недостаточно средств на лицевом счете."),
            ),
          );
        }
      },
    ),
  );
 }
}

class _BonusField extends ConsumerStatefulWidget {
 const _BonusField({super.key});

 @override
 ConsumerState<_BonusField> createState() => __BonusFieldState();
}

class __BonusFieldState extends ConsumerState<_BonusField> {
 final bonusFieldController = TextEditingController();

 @override
```

```
  Widget build(BuildContext context) {
    return Container(
      margin: EdgeInsets.symmetric(horizontal: 8),
      padding: EdgeInsets.symmetric(vertical: 8),
      decoration: BoxDecoration(
        color: Colors.green,
        borderRadius: BorderRadius.circular(32),
      ),
      alignment: Alignment.center,
      child: TextField(
        controller: bonusFieldController,
        keyboardType: TextInputType.number,
        textAlign: TextAlign.center,
        decoration: const InputDecoration.collapsed(hintText: "0"),
        style: const TextStyle(
          fontSize: 17,
          fontWeight: FontWeight.w700,
        ),
        onSubmitted: (text) {
          int bonuses = ref.read(bonusProvider).bonus;
          int value = int.parse(text);
          if (value > bonuses) value = bonuses;
          bonusFieldController.text = value.toString();
          ref.read(_chooseBonus.notifier).state = value;
        },
      ),
    );
  }

  @override
  void dispose() {
    bonusFieldController.dispose();
    super.dispose();
  }
}

class _BoldText extends StatelessWidget {
  final String text;
  final Color? color;
  final double fontSize;

  const _BoldText(this.text, {this.fontSize = 17, this.color});

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

    return Text(
      text,
      style: TextStyle(
        color: color ?? colorScheme.onPrimary,
        fontSize: fontSize,
```

```
        fontWeight: FontWeight.w800,
      ),
    );
  }
}
```

mock_card_payment.dart

```
import 'package:blagodat/presentation/theme/app_colors.dart';
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';

final _numProvider = StateProvider<double>((ref) => 0);

class MockCardPayment extends ConsumerWidget {
  final void Function(double) onPayment;

  const MockCardPayment({required this.onPayment, Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context, WidgetRef ref) {
    return AlertDialog(
      backgroundColor: AppColors.white,
      content: TextField(
        keyboardType: TextInputType.number,
        onChanged: (text) =>
            ref.read(_numProvider.notifier).state = double.parse(text),
      ),
      actions: [
        TextButton(
          onPressed: () {
            onPayment(
              ref.read(_numProvider),
            );
            Navigator.of(context).pop();
          },
          child: const Text("Купить"),
        ),
      ],
    );
  }
}
```

history_fragment.dart

```
import 'package:blagodat/data/shop/transaction.dart';
import 'package:blagodat/domain/di.dart';
import 'package:blagodat/presentation/constants/paddings.dart';
import 'package:blagodat/presentation/widgets/brand_header.dart';
import 'package:blagodat/presentation/widgets/row_end_start.dart';
```

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

class HistoryFragment extends ConsumerWidget {
  static const headerText = BrandedTextWithColor("История", " покупок");

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

  @override
  Widget build(BuildContext context, WidgetRef ref) {
    final transactionChilds = ref
        .watch(historyProvider)
        .map(
          (e) => SizedBox(
            height: 60,
            child: _TransactionCard(e),
          ),
        )
        .toList();

    return SafeArea(
      bottom: false,
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.stretch,
        children: [
          const SizedBox(
            child: BrandHeader(
              brandedText: headerText,
            ),
          ),
          Expanded(
            child: ListView(
              padding: const EdgeInsets.only(
                left: Paddings.large,
                right: Paddings.large,
                top: Paddings.small,
              ),
              children: transactionChilds,
            ),
          ),
        ],
      ),
    );
  }
}

class _TransactionCard extends StatelessWidget {
  static const textStyle = TextStyle(
    fontSize: 24,
    fontWeight: FontWeight.w700,
  );
  final Transaction transaction;
```

```
  const _TransactionCard(this.transaction);

  @override
  Widget build(BuildContext context) {
    String shortDate = "${transaction.purchaseTime.day}." +
        "${transaction.purchaseTime.month}." +
        "${transaction.purchaseTime.year}";
    return RowEndStart(
      start: Text("$shortDate", style: textStyle),
      end: Text(
        "\$ ${transaction.totalCost.toStringAsFixed(2)}",
        style: textStyle,
      ),
    );
  }
}
```

loyalty_fragment.dart

```
import 'package:blagodat/domain/di.dart';
import 'package:blagodat/presentation/constants/borders_radius.dart';
import 'package:blagodat/presentation/constants/paddings.dart';
import 'package:blagodat/presentation/constants/shadows.dart';
import 'package:blagodat/presentation/pages/main_page.dart';
import 'package:blagodat/presentation/theme/app_colors.dart';
import 'package:blagodat/presentation/widgets/brand_header.dart';
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';

/// Page fragment with the description of the loyalty program.
class LoyaltyFragment extends StatelessWidget {
  static const headerText = BrandedTextWithColor(
    "Loyalty ",
    "Program",
  );

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

  @override
  Widget build(BuildContext context) {
    return const SafeArea(
      bottom: false,
      child: Column(
        mainAxisSize: MainAxisSize.max,
        crossAxisAlignment: CrossAxisAlignment.stretch,
        children: [
          SizedBox(
            width: double.infinity,
            child: BrandHeader(
              brandedText: headerText,
```

```
            ),
          ),
          // TODO: Info card.
          Flexible(
            flex: 2,
            fit: FlexFit.tight,
            child: Placeholder(),
          ),
          // TODO: Levels info.
          Flexible(
            flex: 3,
            fit: FlexFit.tight,
            child: Placeholder(),
          ),
          // TODO: Bonus count info.
          SizedBox(
            height: 120,
            child: _BonusCard(),
          ),
          // Bottom bar space
          SizedBox(height: MainPage.pageBarPadding),
        ],
      ),
    );
  }
}

class _BonusCard extends ConsumerWidget {
  const _BonusCard();

  @override
  Widget build(BuildContext context, WidgetRef ref) {
    ref.watch(streamPurchaseProvider);

    return Container(
      padding: const EdgeInsets.all(Paddings.small),
      decoration: BoxDecoration(
        color: AppColors.white,
        borderRadius: BordersRadius.standart,
        boxShadow: const [Shadows.topDown],
      ),
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        crossAxisAlignment: CrossAxisAlignment.center,
        children: [
          Expanded(
            child: Padding(
              padding: const EdgeInsets.symmetric(vertical: 0.0),
              child: Row(
                mainAxisAlignment: MainAxisAlignment.center,
                children: [
                  Icon(Icons.breakfast_dining_sharp),
```

```
                Container(
                  decoration: ShapeDecoration(
                    shape: StadiumBorder(),
                    color: AppColors.gold,
                  ),
                  padding: EdgeInsets.symmetric(horizontal: 8),
                  child: Text("132/250"),
                ),
              ],
            ),
          ),
        ),
        Expanded(
          child: Text(ref.watch(bonusProvider).bonus.toString()),
        ),
      ],
    ),
  );
 }
}

class _FactRow extends StatelessWidget {
 final String text;

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

 @override
 Widget build(BuildContext context) {
  return Row(
    children: [
      const CircleAvatar(
        foregroundColor: AppColors.mainOrange,
      ),
      Text(text),
    ],
  );
 }
}
```

home_fragment.dart

```
import 'package:blagodat/domain/di.dart';
import 'package:blagodat/presentation/constants/borders_radius.dart';
import 'package:blagodat/presentation/constants/font_sizes.dart';
import 'package:blagodat/presentation/constants/paddings.dart';
import 'package:blagodat/presentation/pages/cart/cart_page.dart';
import 'package:blagodat/presentation/pages/home/product_home_preview_card.dart';
import 'package:blagodat/presentation/theme/app_colors.dart';
import 'package:blagodat/presentation/widgets/brand_header.dart';
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';
```

```
class HomeFragment extends StatelessWidget {
  static const promoUrl =
      "https://www.crushpixel.com/big-static18/preview4/sale-banner-template-design-limited-
2766676.jpg";
  static const headerText = BrandedTextWithColor("Благо", "Дать");
  static const double categoryHeaderSize = 38;

  /// Height that AppBar can expand.
  static const double expandedHeight = BrandHeader.kHeaderHeight + 25;

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

  @override
  Widget build(BuildContext context) {
    return CustomScrollView(
      slivers: [
        SliverAppBar(
          expandedHeight: expandedHeight,
          collapsedHeight: BrandHeader.kHeaderHeight,
          pinned: true,
          backgroundColor: AppColors.white,
          surfaceTintColor: Colors.white,
          flexibleSpace: FlexibleSpaceBar(
            expandedTitleScale: 1,
            titlePadding: EdgeInsets.zero,
            title: BrandHeader(
              brandedText: headerText,
              trailing: GestureDetector(
                onTap: () => Navigator.of(context).push(
                  MaterialPageRoute(
                    builder: (_) => const CartPage(),
                  ),
                ),
                child: const Icon(
                  Icons.shopping_cart_outlined,
                  size: 32,
                ),
              ),
            ),
          ),
        ),
        // TODO: Promo
        SliverToBoxAdapter(
          child: Padding(
            padding: EdgeInsets.symmetric(horizontal: Paddings.medium),
            child: SizedBox(
              height: 240,
              child: ClipRRect(
                borderRadius: BordersRadius.standart,
                child: Image.network(
                  promoUrl,
```

```
                fit: BoxFit.cover,
              ),
            ),
          ),
        ),
      ),
      // TODO: Categories
      const SliverAppBar(
        pinned: true,
        // Disable expanding
        toolbarHeight: 1,
        collapsedHeight: categoryHeaderSize,
        surfaceTintColor: Colors.white,
        backgroundColor: Colors.white,
        flexibleSpace: FlexibleSpaceBar(
          titlePadding: EdgeInsets.zero,
          title: Center(
            child: Text(
              "Продукты",
              style: TextStyle(
                fontSize: FontSize.large,
                fontWeight: FontWeight.w700,
              ),
            ),
          )),
      ),
      const SliverPadding(
        padding: EdgeInsets.symmetric(horizontal: Paddings.medium),
        sliver: _ProductSliverList(),
      ),
    ],
  );
 }
}

/// Список товавров в магазине.
class _ProductSliverList extends ConsumerWidget {
 static const cardRatio = 156 / 225;

 const _ProductSliverList();

 @override
 Widget build(BuildContext context, WidgetRef ref) {
  final products = ref.watch(assortmentProvider).products;
  final productCards = [
   for (final product in products)
    // Центруем, чтобы поместить в ячейку [SliverGrid].
    Center(
     child: Padding(
      padding: const EdgeInsets.all(Paddings.small),
      child: ProductHomePreviewCard(product: product),
     ),
```

```
      ),
    ];

    // ? Может быть заменить на [SliverGrid.builder]
    return SliverGrid.count(
      crossAxisCount: 2,
      childAspectRatio: cardRatio,
      mainAxisSpacing: Paddings.small,
      children: productCards,
    );
  }
}
```

product_home_preview_card.dart

```
import 'package:blagodat/data/shop/info/product.dart';
import 'package:blagodat/presentation/constants/font_sizes.dart';
import 'package:blagodat/presentation/constants/paddings.dart';
import 'package:blagodat/presentation/pages/product_preview_page.dart';
import 'package:flutter/material.dart';

/// Виджет, отображающий краткую информацию о продукте.
class ProductHomePreviewCard extends StatelessWidget {
  static const double imageBorderRadius = 16;

  final Product product;

  const ProductHomePreviewCard({
    Key? key,
    required this.product,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Column(
      crossAxisAlignment: CrossAxisAlignment.stretch,
      children: [
        Expanded(
          child: GestureDetector(
            onTap: () => Navigator.of(context).push(
              MaterialPageRoute(
                builder: (_) => ProductPreviewPage(product: product),
              ),
            ),
            child: ClipRRect(
              borderRadius: BorderRadius.circular(imageBorderRadius),
              child: Image.network(
                product.imageURL[0],
                fit: BoxFit.cover,
              ),
            ),
```

```
        ),
      ),
      const SizedBox(height: Paddings.small / 2),
      Text(
        product.name,
        style: const TextStyle(
          fontSize: FontSize.small,
          overflow: TextOverflow.ellipsis,
          fontWeight: FontWeight.w400,
          height: 1.2,
        ),
        maxLines: 2,
      ),
      Text(
        "${product.cost.toStringAsFixed(2)}\$",
        style: const TextStyle(
          fontSize: FontSize.small,
          fontWeight: FontWeight.w900,
        ),
      ),
    ],
  );
 }
}
```

app_colors.dart

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

sealed class AppColors {
 /// Main color for app.
 static const mainTheme = gold;
 static const gold = Color.fromRGBO(248, 222, 0, 1);
 static const mainOrange = Color.fromRGBO(255, 193, 7, 1);
 static const darkOrange = Color.fromRGBO(251, 192, 45, 1);
 static const softDark = Color.fromRGBO(33, 33, 33, 1);
 static const grey = Color.fromRGBO(117, 117, 117, 1);
 static const lightGrey = Color.fromRGBO(189, 189, 189, 1);
 static const softWhite = Color.fromRGBO(248, 248, 248, 1);
 static const whiteYellow = Color.fromRGBO(254, 254, 244, 1);
 static const white = Colors.white;
}
```

app_theme.dart

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

import 'app_colors.dart';

sealed class AppTheme {
```

```
  static final lightTheme = ThemeData(
    colorScheme: const ColorScheme.light(
      primary: AppColors.gold,
      onPrimary: AppColors.softDark,
      secondary: AppColors.darkOrange,
      onSecondary: AppColors.softDark,
      surface: AppColors.softWhite,
      surfaceVariant: AppColors.white,
      onSurface: AppColors.softDark,
      tertiary: AppColors.grey,
    ),
    brightness: Brightness.light,
    useMaterial3: true,
  );
}
```

brand_header.dart

```
import 'package:blagodat/presentation/constants/font_sizes.dart';
import 'package:blagodat/presentation/constants/paddings.dart';
import 'package:flutter/material.dart';

import '../theme/app_colors.dart';

/// Widget that displays the page name as a branded header.
class BrandHeader extends StatelessWidget {
  /// Height of header container.
  static const double kHeaderHeight = 90;

  /// Header text with several color.
  final BrandedTextWithColor brandedText;

  /// Widget that shown in end of header.
  final Widget? trailing;

  const BrandHeader({Key? key, required this.brandedText, this.trailing})
      : super(key: key);

  @override
  Widget build(BuildContext context) {
    return SizedBox(
      height: kHeaderHeight,
      width: double.infinity,
      child: Padding(
        padding: const EdgeInsets.symmetric(
          horizontal: Paddings.maximum,
          vertical: Paddings.small,
        ),
        child: Row(
          crossAxisAlignment: CrossAxisAlignment.center,
          children: [
```

```
          Expanded(
            child: _BrandHeaderText(brandedText),
          ),
          Align(
            alignment: Alignment.centerRight,
            child: trailing ?? const SizedBox.shrink(),
          ),
        ],
      ),
    ),
  );
 }
}

class _BrandHeaderText extends StatelessWidget {
 static const FontWeight prefixWeight = FontWeight.w400;
 static const FontWeight suffixWeight = FontWeight.w700;
 final BrandedTextWithColor brandedTextItemColor;

 const _BrandHeaderText(this.brandedTextItemColor);

 @override
 Widget build(BuildContext context) {
  return RichText(
    text: TextSpan(
      style: const TextStyle(
        fontSize: FontSize.header,
      ),
      children: [
        TextSpan(
          text: brandedTextItemColor.prefix,
          style: TextStyle(
            color: Theme.of(context).colorScheme.onPrimary,
            fontWeight: prefixWeight,
          ),
        ),
        TextSpan(
          text: brandedTextItemColor.suffix,
          style: const TextStyle(
            color: AppColors.mainTheme,
            fontWeight: suffixWeight,
          ),
        ),
      ],
    ),
  );
 }
}

/// Defines the text content with color scheme.
///
/// See also:
```

```
/// - [BrandHeader], a widget that creates a brand header app bar.
final class BrandedTextWithColor {
  /// Word that would be colored in black.
  final String prefix;

  /// Word that would be colored in yellow.
  final String suffix;

  const BrandedTextWithColor(this.prefix, this.suffix);
}
```

change_counter_button.dart

```
import 'package:blagodat/presentation/constants/durations.dart';
import 'package:blagodat/presentation/constants/shadows.dart';
import 'package:flutter/material.dart';

/// Виджет, который может отображать счетчик, увеличивать и уменьшать его.
///
/// Если счетчик < 1, то он не отображается и виджет превращается в обычную
/// кнопку с надписью [zeroTitle].
class ChangeCounterButton extends StatelessWidget {
  static const double zeroTitleFontSize = 20;
  static const double counterFontSize = 20;
  static const fontWeight = FontWeight.w500;
  static const double counterYAnimation = -0.3;

  /// Заголовок, который отображается на кнопке, если [counter] == 0.
  final String zeroTitle;

  /// Значение, отображаемое на счетчике.
  final int counter;

  /// Вызывается при нажатие на кнопку или при нажатие на инкремент.
  final void Function() increase;

  /// Отступ кнопки при ненулевом значении счетчика.
  final double paddingNonZero;

  /// Вызывается при нажатии на кнопку декремента.
  final void Function() decrase;
  const ChangeCounterButton({
    Key? key,
    required this.zeroTitle,
    required this.counter,
    required this.increase,
    required this.decrase,
    this.paddingNonZero = 8,
  }) : super(key: key);

  @override
```

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

 // Виджет, для отображения счетчтика.
 final counterChild = counter > 0
    // Круглый счетчик.
    ? Container(
        decoration: ShapeDecoration(
          shape: const CircleBorder(),
          color: colorScheme.surfaceVariant,
          shadows: const [Shadows.topDown],
        ),
        alignment: Alignment.center,
        child: Text(
          "$counter",
          style: TextStyle(
            color: colorScheme.onPrimary,
            fontWeight: fontWeight,
            fontSize: counterFontSize,
          ),
        ),
      )
    // Иначе ничего не отображаем.
    : null;

 // Виджет для отображения элементов контроля на кнопке.
 final buttonChild = counter > 0
    ? _IcnrementButtons(
        increase: increase,
        decrase: decrase,
      )
    : Center(
        child: Text(
          zeroTitle,
          style: TextStyle(
            fontSize: zeroTitleFontSize,
            fontWeight: fontWeight,
            color: colorScheme.onPrimary,
          ),
        ),
      );

 return Stack(
  alignment: Alignment.center,
  fit: StackFit.expand,
  children: [
   // Отображаем кнопку и элементы на ней.
   AnimatedPositioned(
    top: _buttonPadding,
    bottom: _buttonPadding,
    left: 0,
    right: 0,
```

```
        duration: Durations.standart,
        // Отображаем кнопку, и если [counter] = 0, то включаем её.
        child: GestureDetector(
          onTap: () => counter > 0 ? null : increase(),
          child: Container(
            decoration: ShapeDecoration(
              shape: const StadiumBorder(),
              color: colorScheme.secondary,
              shadows: const [Shadows.topDown],
            ),
            child: AnimatedSwitcher(
              duration: Durations.standart,
              child: buttonChild,
            ),
          ),
        ),
      ),
      // Отображаем счетчик
      Positioned(
        child: AnimatedSwitcher(
          transitionBuilder: (Widget child, Animation<double> animation) {
            // Появление сверху с постепенным появлением.
            return FadeTransition(
              opacity: animation,
              child: SlideTransition(
                position: Tween(
                  begin: const Offset(0, counterYAnimation),
                  end: const Offset(0, 0),
                ).animate(animation),
                child: child,
              ),
            );
          },
          duration: Durations.standart,
          child: counterChild,
        ),
      ),
    ],
  );
 }

 double get _buttonPadding => counter > 0 ? paddingNonZero : 0;
}

/// Ряд кнопок для уменьшения, а затем увелечения чего либо.
class _IcnrementButtons extends StatelessWidget {
 final void Function() increase;
 final void Function() decrase;

 const _IcnrementButtons({
  required this.increase,
  required this.decrase,
```

```
  });

  @override
  Widget build(BuildContext context) {
    return Row(
      children: [
        Expanded(
          child: Align(
            alignment: Alignment.centerLeft,
            child: IconButton(
              onPressed: decrase,
              icon: const Icon(Icons.remove),
            ),
          ),
        ),
        Expanded(
          child: Align(
            alignment: Alignment.centerRight,
            child: IconButton(
              onPressed: increase,
              icon: const Icon(Icons.add),
            ),
          ),
        )
      ],
    );
  }
}
```

floating_bottom_bar.dart

```
import 'dart:ui';

import 'package:flutter/material.dart';

/// Navigation bar with stadium borders.
class FloatingBottomBar extends StatefulWidget {
  static const kMargin = EdgeInsets.only(left: 64, right: 64, bottom: 32);
  static const kHeight = 96.0;
  static const kBlur = 4.0;
  static const kChildPadding = 8.0;
  static const kDoubleWidth = double.maxFinite;
  static const kAlignment = Alignment.bottomCenter;

  /// Background color of bottom bar.
  final Color? surfaceColor;

  /// Color for active icon button.
  final Color activeColor;

  /// Color for inactive icon button.
```

```
final Color inactiveColor;

/// Shadows from background of bottom bar.
final List<BoxShadow>? shadows;

/// The margin of floating bar sides.
///
/// By default is set for [kMargin].
final EdgeInsets margin;

/// Maximum width of floating bar container.
///
/// By default is set for [double.maxFinite].
final double maxWidth;

/// Height of floating bar container.
///
/// By default is set for [kHeight].
final double height;

/// List of icons that displayed in bottom bar.
final List<IconData> icons;

/// Index of start selected icon from [icons].
final int startItemIndex;

/// The blur of background of bottom bar.
///
/// By default is set for [kBlur].
final double blur;

/// Alignment of bar.
///
/// By defaul is set for [kAlignment]
final Alignment alignment;

/// Padding of items in floating bottom bar.
///
/// By default is set for [kChildPadding].
final double childPadding;

/// Callback about switching the selected bar element to [selectedIndex].
final Function(int selectedIndex) onSelected;

const FloatingBottomBar({
  this.surfaceColor,
  this.shadows,
  this.margin = kMargin,
  this.maxWidth = kDoubleWidth,
  this.height = kHeight,
  this.blur = kBlur,
  this.startItemIndex = 0,
```

```
    this.alignment = kAlignment,
    this.childPadding = kChildPadding,
    required this.icons,
    required this.activeColor,
    required this.inactiveColor,
    required this.onSelected,
    Key? key,
  })  : assert(
          startItemIndex < icons.length,
          "Start item index shoud be contained in icons.",
        ),
        super(key: key);

  @override
  State<FloatingBottomBar> createState() => _FloatingBottomBarState();
}

class _FloatingBottomBarState extends State<FloatingBottomBar> {
  /// Selected index in [widget.icons].
  late int _nowSelectedIndex;

  @override
  void initState() {
    super.initState();
    _nowSelectedIndex = widget.startItemIndex;
  }

  @override
  Widget build(BuildContext context) {
    final iconButtons = [
      // Build active or inactive icon button.
      for (int i = 0; i < widget.icons.length; i++)
        Expanded(
          child: _BottomBarIconButton(
            onTap: () => _updateSelectedIndex(i),
            iconData: widget.icons[i],
            color: _nowSelectedIndex == i
                ? widget.activeColor
                : widget.inactiveColor,
            isEnabled: _nowSelectedIndex == i,
          ),
        )
    ];

    return Align(
      alignment: widget.alignment,
      child: Padding(
        padding: widget.margin,
        // Create blur effect.
        child: ClipRRect(
          borderRadius: BorderRadius.circular(1000),
          child: BackdropFilter(
```

```
          filter: ImageFilter.blur(sigmaX: widget.blur, sigmaY: widget.blur),
          child: Material(
            color: widget.surfaceColor,
            // Content
            child: Container(
              constraints: BoxConstraints(maxWidth: widget.maxWidth),
              clipBehavior: Clip.hardEdge,
              height: widget.height,
              decoration: ShapeDecoration(
                shape: const StadiumBorder(),
                shadows: widget.shadows,
              ),
              padding: EdgeInsets.all(widget.childPadding),
              child: Row(
                mainAxisSize: MainAxisSize.max,
                mainAxisAlignment: MainAxisAlignment.spaceEvenly,
                crossAxisAlignment: CrossAxisAlignment.stretch,
                children: iconButtons,
              ),
            ),
          ),
        ),
      ),
    ),
  );
 }

 /// Updates selected item.
 void _updateSelectedIndex(int selectedIndex) {
  if (_nowSelectedIndex == selectedIndex) return;
  setState(() {
   _nowSelectedIndex = selectedIndex;
  });
  widget.onSelected(_nowSelectedIndex);
 }
}

class _BottomBarIconButton extends StatelessWidget {
 static const double inkWellPadding = 4;
 static const scaleDuration = Duration(milliseconds: 100);
 static const maxScale = 1.0;
 static const minScale = 0.85;

 final Function() onTap;
 final IconData iconData;
 final Color color;

 /// Button is active or not.
 final bool isEnabled;

 const _BottomBarIconButton({
  required this.onTap,
```

```
    required this.iconData,
    required this.color,
    required this.isEnabled,
  });

  @override
  Widget build(BuildContext context) {
    return AnimatedScale(
      duration: scaleDuration,
      scale: isEnabled ? maxScale : minScale,
      child: FittedBox(
        child: InkWell(
          onTap: onTap,
          customBorder: const StadiumBorder(),
          child: Padding(
            padding: const EdgeInsets.all(inkWellPadding),
            child: Icon(
              iconData,
              color: color,
            ),
          ),
        ),
      ),
    );
  }
}
```

product_in_cart_card.dart

```
import 'package:blagodat/data/shop/info/product.dart';
import 'package:blagodat/domain/di.dart';
import 'package:blagodat/presentation/constants/borders_radius.dart';
import 'package:blagodat/presentation/constants/paddings.dart';
import 'package:blagodat/presentation/pages/product_preview_page.dart';
import 'package:blagodat/presentation/widgets/change_counter_button.dart';
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';

class ProductInCartCard extends StatelessWidget {
  final Product product;

  const ProductInCartCard({Key? key, required this.product}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return GestureDetector(
      onTap: () => Navigator.of(context).push(
        MaterialPageRoute(
          builder: (_) => ProductPreviewPage(product: product),
        ),
      ),
```

```
      child: Row(
        children: [
          AspectRatio(
            aspectRatio: 0.95,
            child: ClipRRect(
              borderRadius: BordersRadius.standart,
              child: Image.network(
                product.imageURL[0],
                fit: BoxFit.cover,
              ),
            ),
          ),
          const SizedBox(width: 16),
          Expanded(
            child: Padding(
              padding: const EdgeInsets.only(top: Paddings.small),
              child: _InCartSideInfo(product),
            ),
          )
        ],
      ),
    );
  }
}

/// Информ
class _InCartSideInfo extends ConsumerWidget {
  final Product product;

  const _InCartSideInfo(this.product);

  @override
  Widget build(BuildContext context, WidgetRef ref) {
    return Column(
      children: [
        // Product name.
        Expanded(
          child: Align(
            alignment: Alignment.topLeft,
            child: Text(
              product.name,
              maxLines: 2,
              style: const TextStyle(
                overflow: TextOverflow.visible,
                fontSize: 21,
                fontWeight: FontWeight.w700,
              ),
            ),
          ),
        ),
        // Product control.
        Expanded(
```

```
        child: Row(
          children: [
            Expanded(
              flex: 3,
              child: Align(
                alignment: Alignment.centerLeft,
                child: Text(
                  "\$${product.cost}",
                  style: const TextStyle(
                    fontWeight: FontWeight.w600,
                    fontSize: 18,
                  ),
                ),
              ),
            ),
            Expanded(
              flex: 4,
              child: Align(
                alignment: Alignment.centerRight,
                child: Padding(
                  padding: EdgeInsets.symmetric(vertical: 2),
                  child: ChangeCounterButton(
                    zeroTitle: "Добавить",
                    counter: ref.watch(cartProvider)[product]!,
                    increase: () =>
                        ref.read(cartProvider.notifier).add(product),
                    decrase: () =>
                        ref.read(cartProvider.notifier).decrase(product),
                  ),
                ),
              ),
            ),
          ],
        ),
      ),
    ],
  );
 }
}
```

row_end_start.dart

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

class RowEndStart extends StatelessWidget {
  final Widget start;
  final Widget end;

  const RowEndStart({
    Key? key,
    required this.start,
```

```
    required this.end,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Row(
      crossAxisAlignment: CrossAxisAlignment.stretch,
      children: [
        Expanded(
          child: Align(
            alignment: Alignment.centerLeft,
            child: start,
          ),
        ),
        Expanded(
          child: Align(
            alignment: Alignment.centerRight,
            child: end,
          ),
        ),
      ],
    );
  }
}
```

sub_page.dart

```
import 'package:blagodat/presentation/constants/paddings.dart';
import 'package:blagodat/presentation/theme/app_colors.dart';
import 'package:flutter/material.dart';

/// Виджет, представляющий дополнительную страницу в стеке навигации.
///
/// Используется для страницы навигации и страницы корзины.
class SubPage extends StatelessWidget {
  static const double appBarHeight = 70;

  /// Название страницы, отображаемое по центру сверху.
  final String title;

  /// Виджет, отображаемый справа в заголовке.
  final Widget? trailing;

  /// Тело страницы.
  final Widget body;

  const SubPage(
      {Key? key, required this.title, this.trailing, required this.body})
      : super(key: key);

  @override
```

```
  Widget build(BuildContext context) {
    final statusBarHeight = MediaQuery.of(context).padding.top;

    return Scaffold(
      appBar: PreferredSize(
        preferredSize: const Size.fromHeight(appBarHeight),
        child: Padding(
          padding: EdgeInsets.only(
            left: Paddings.small,
            top: statusBarHeight / 2,
            right: Paddings.small,
            bottom: Paddings.small,
          ),
          child: _SubPageAppBar(
            title: title,
            trailing: trailing,
          ),
        ),
      ),
      body: SafeArea(
        child: body,
      ),
    );
  }
}

/// Заголовок [SubPage].
class _SubPageAppBar extends StatelessWidget {
  static const titleWeight = FontWeight.w700;

  /// Заглавие AppBar, отображаемое по центру.
  final String title;

  /// Виджет, находящийся полсе заглавия в AppBar.
  final Widget? trailing;

  // ignore: unused_element
  const _SubPageAppBar({required this.title, this.trailing});

  @override
  Widget build(BuildContext context) {
    return Row(
      children: [
        Expanded(
          child: FittedBox(
            fit: BoxFit.fitHeight,
            child: IconButton(
              icon: const Icon(
                Icons.arrow_back_ios_new,
                color: AppColors.mainTheme,
              ),
              // Переходим к предыдущему экрану.
```

```
            onPressed: () {
              Navigator.of(context).pop();
            },
          ),
        ),
      ),
      Expanded(
        flex: 3,
        child: Padding(
          padding: const EdgeInsets.all(Paddings.medium + 6),
          child: FittedBox(
            fit: BoxFit.fitHeight,
            child: Text(
              title,
              style: const TextStyle(
                fontWeight: titleWeight,
              ),
            ),
          ),
        ),
      ),
      Expanded(
        child: trailing ?? const SizedBox.shrink(),
      ),
    ],
  );
 }
}
```