БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ

И ИНФОРМАТИКИ

Кафедра технологий программирования

ЛАБОРАТОРНАЯ РАБОТА 10

ПО ДИСЦИПЛИНЕ «ПРОГРАММИРОВАНИЕ МОБИЛЬНЫХ И

ВСТРАИВАЕМЫХ СИСТЕМ»

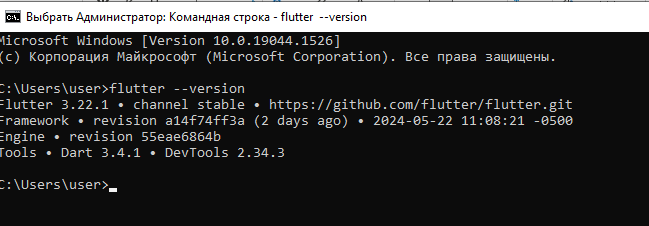
Разработка мобильного приложения для ОС Автора с использованием

Flutter SDK

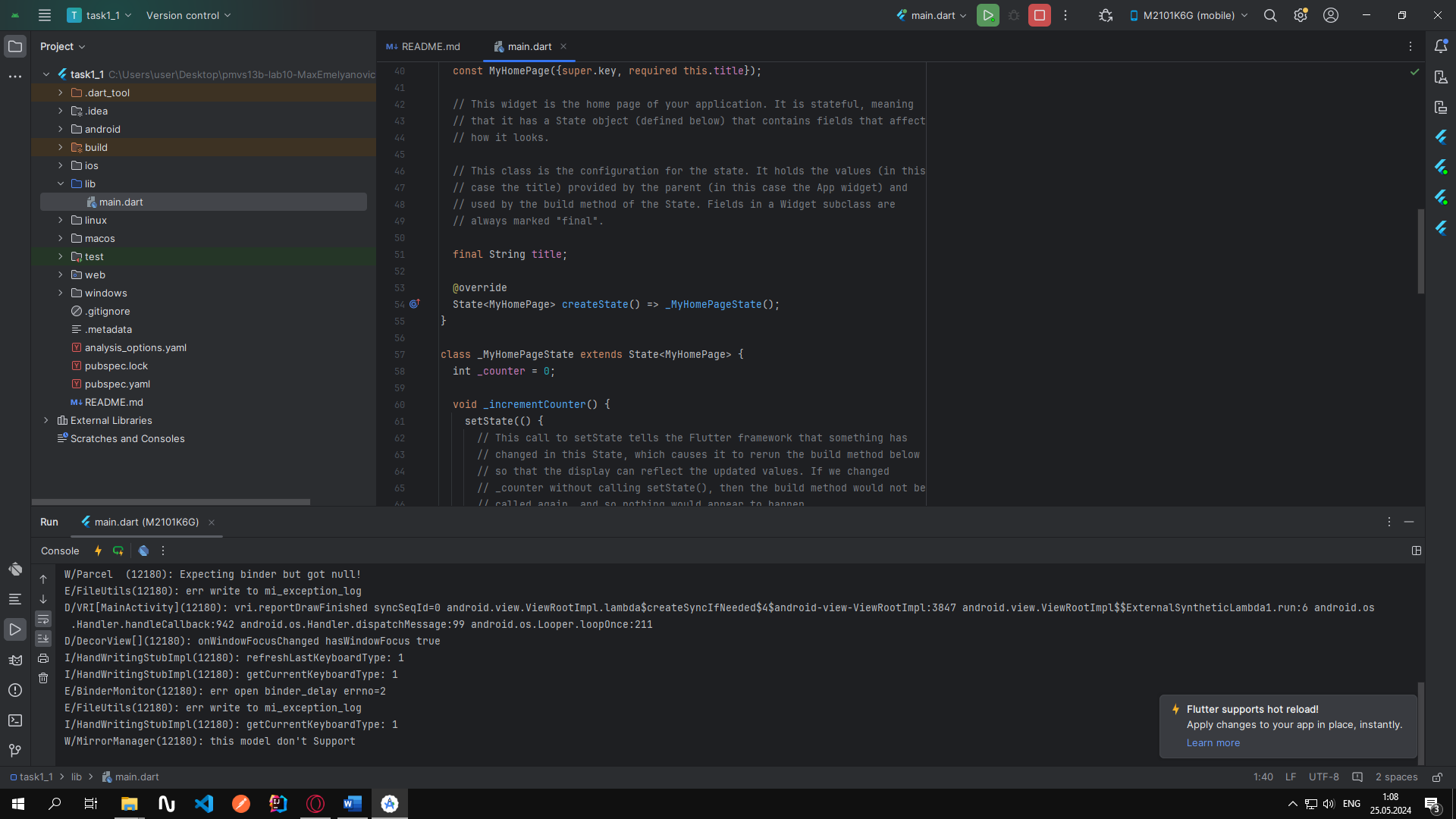
Лебедевич Артём

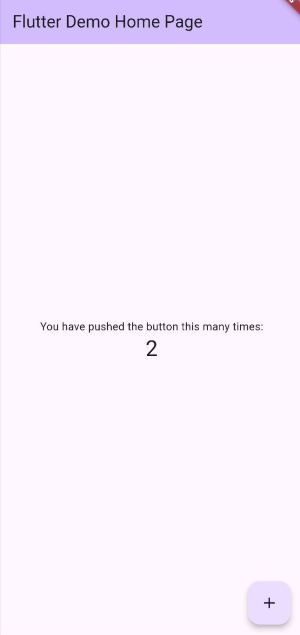
Задание 1

Установлен Flutter SDK

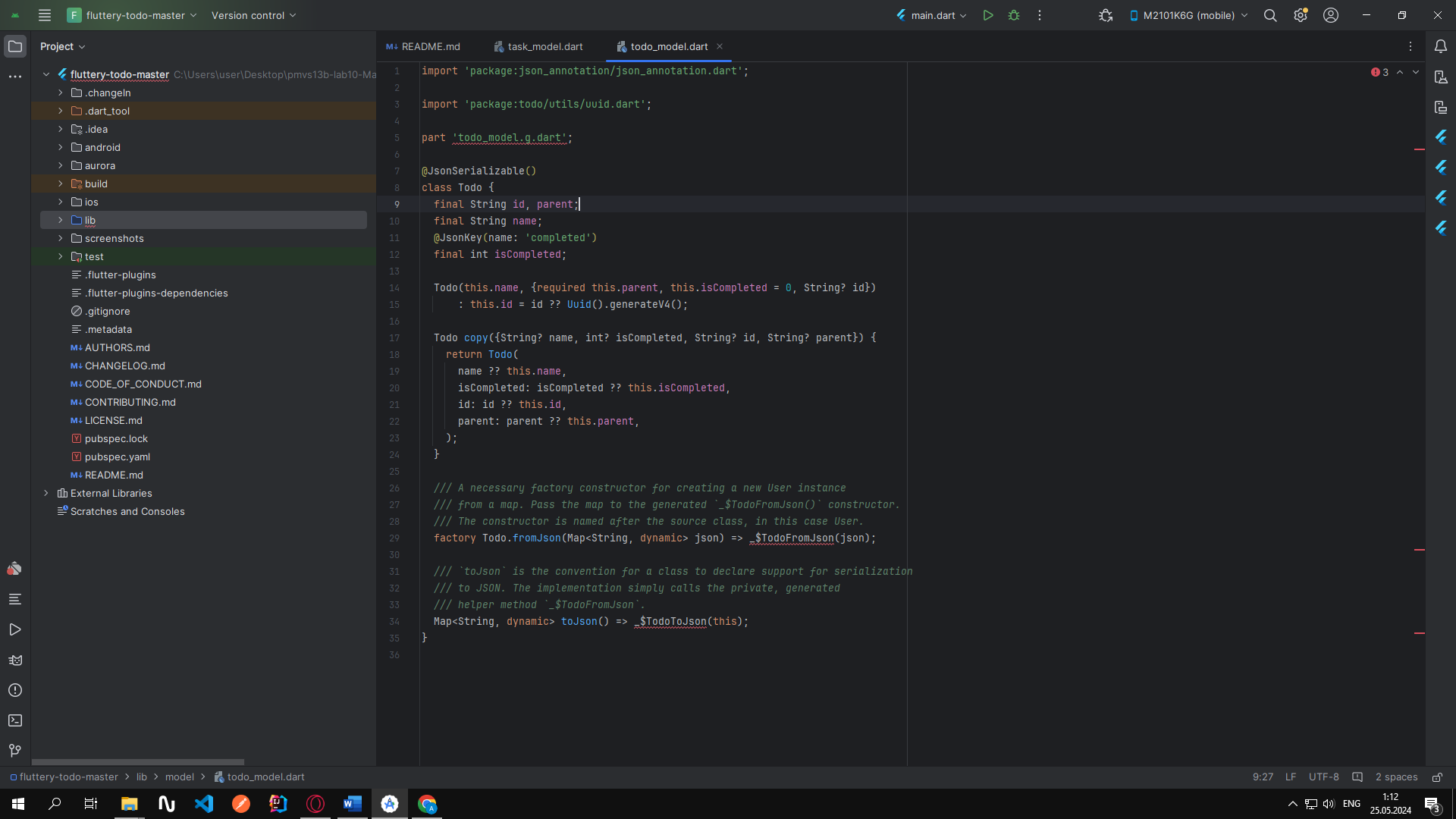


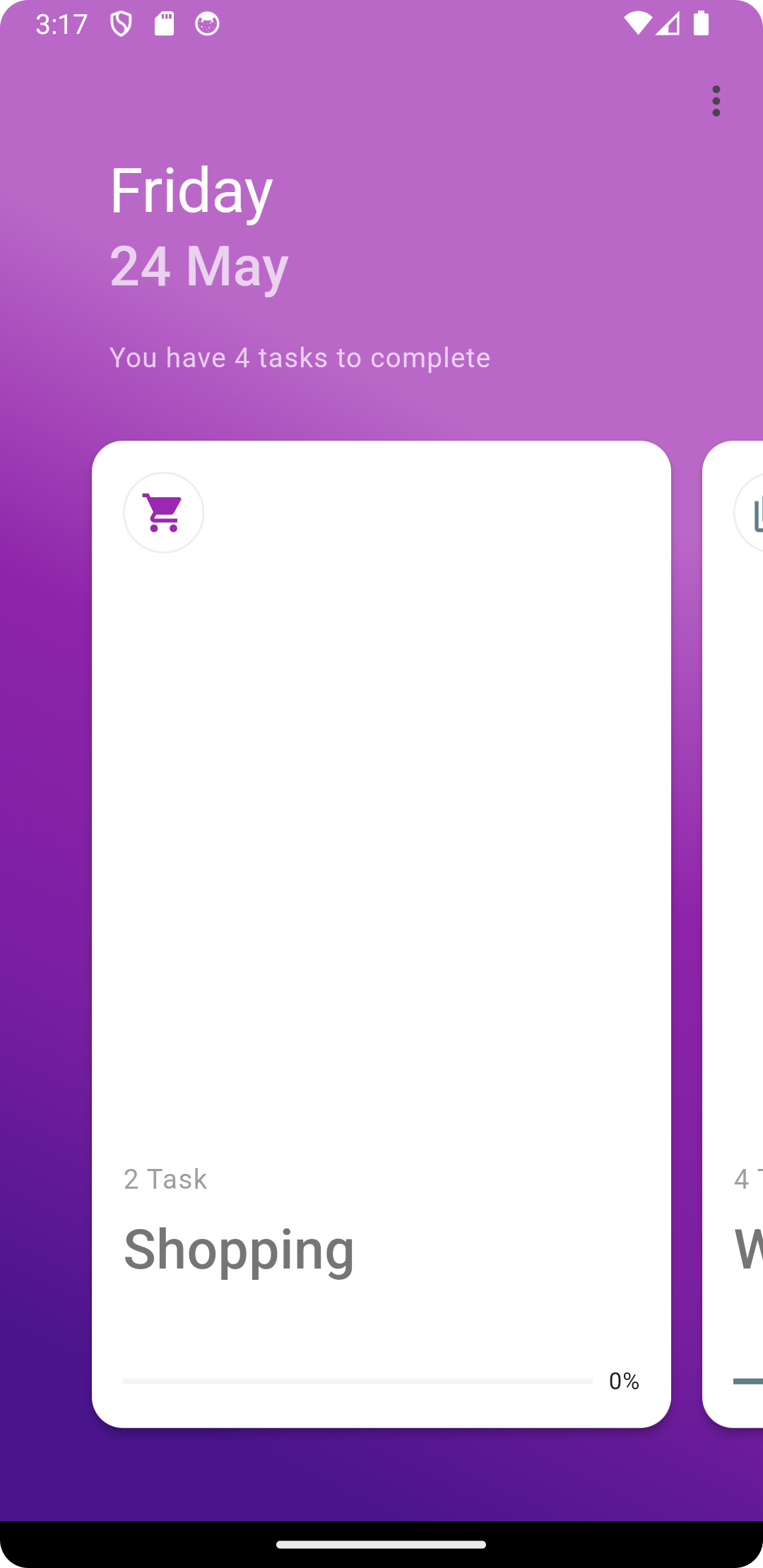
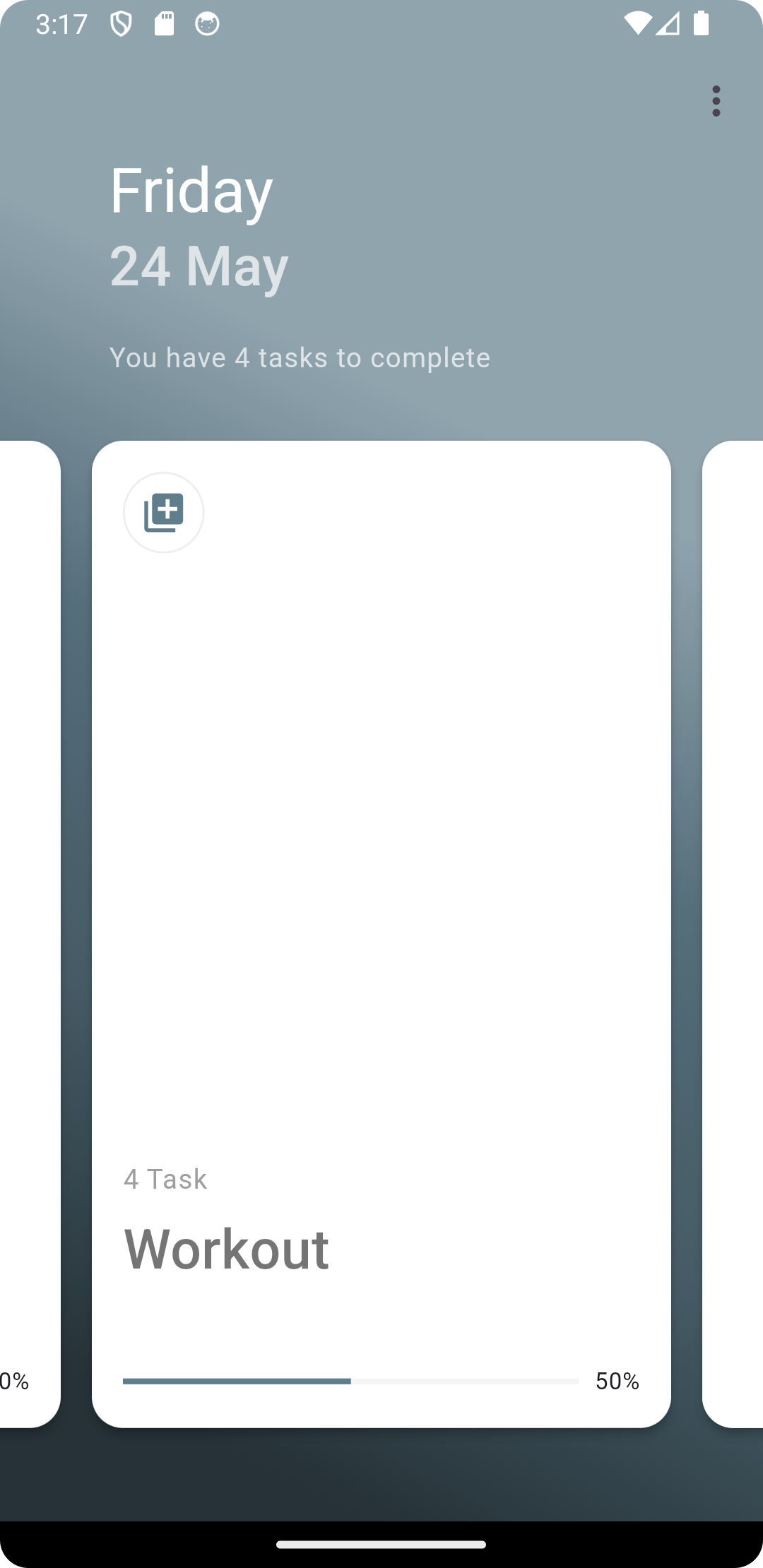
Приложение Hello World

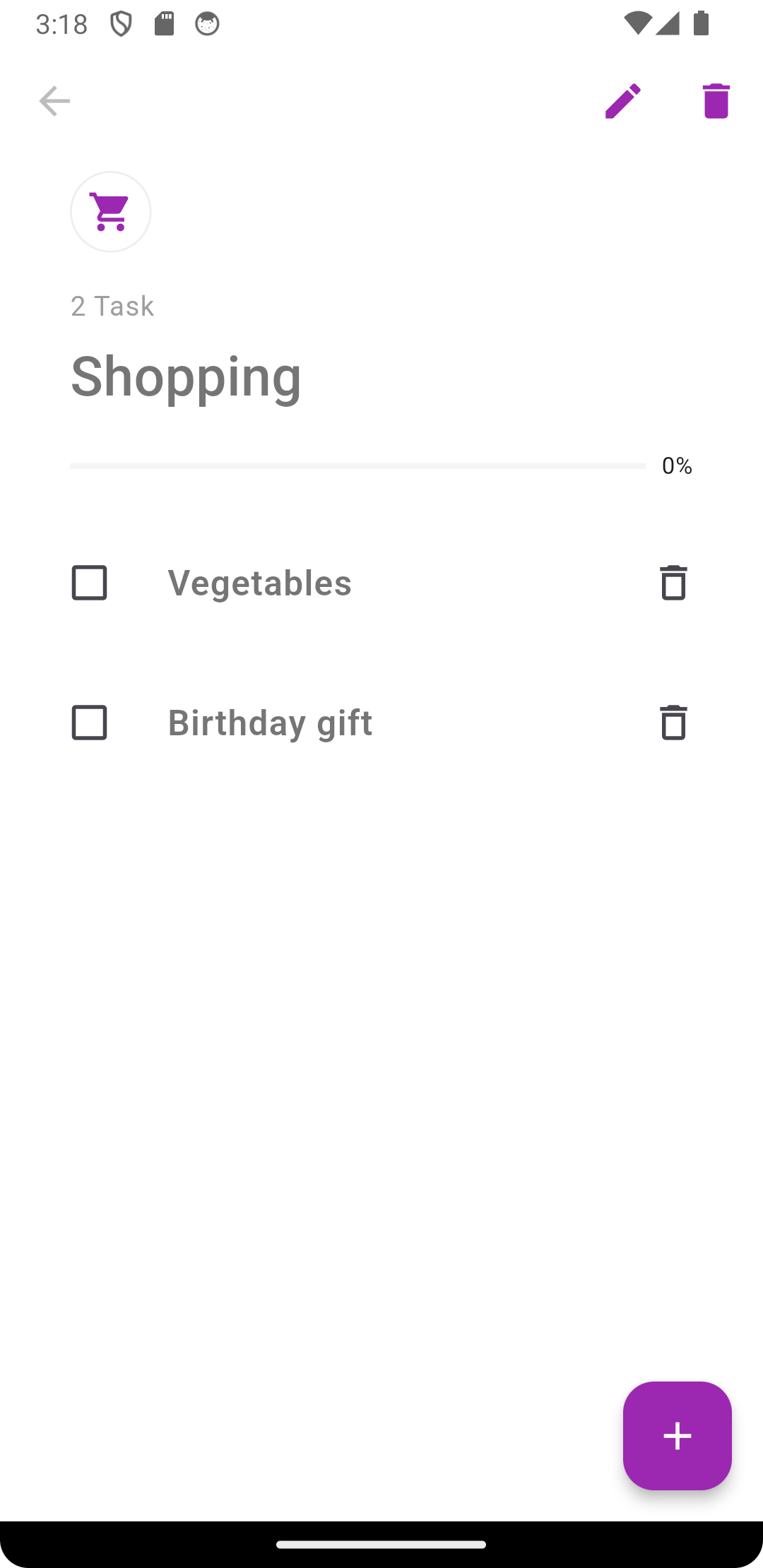
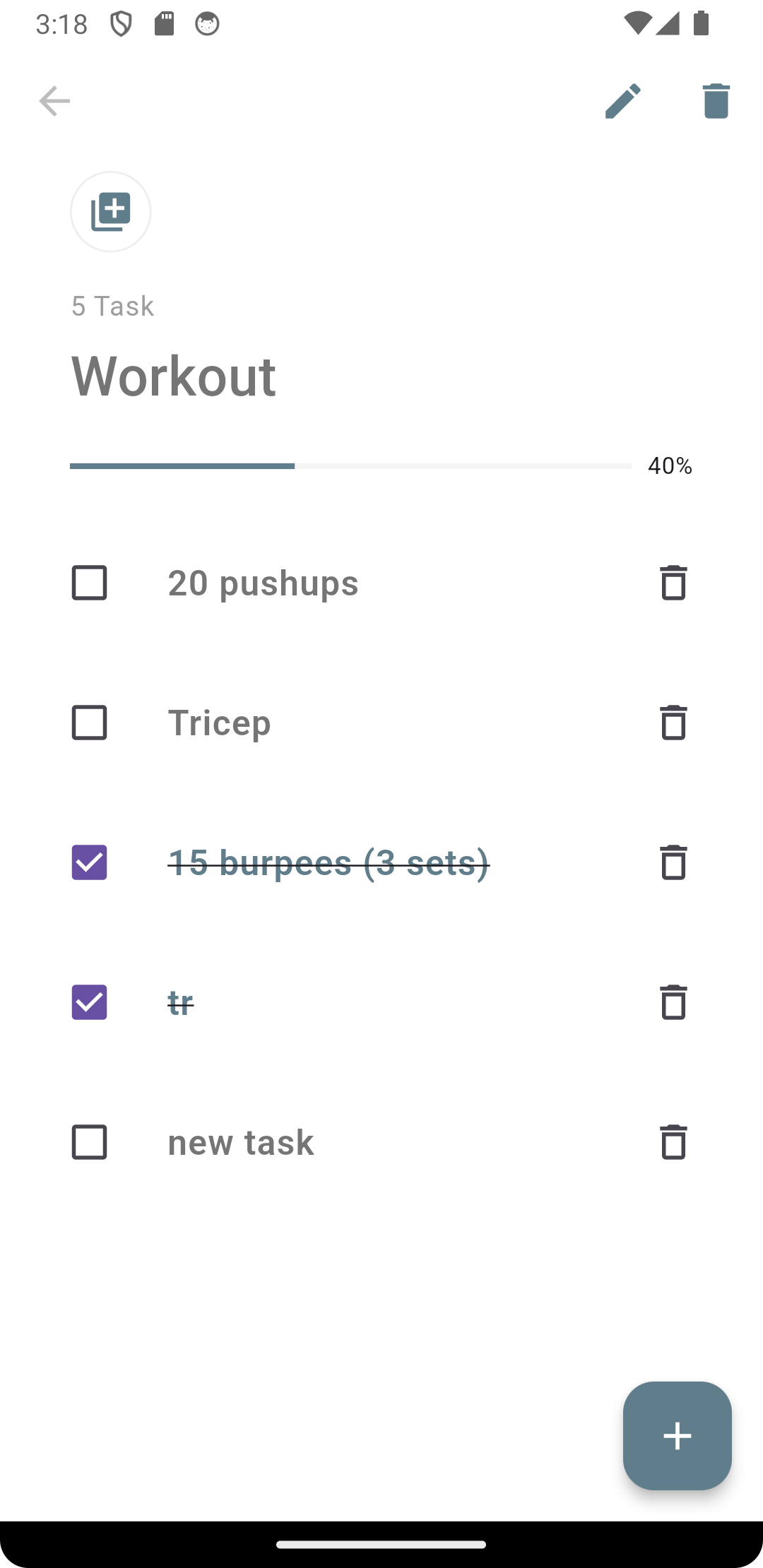




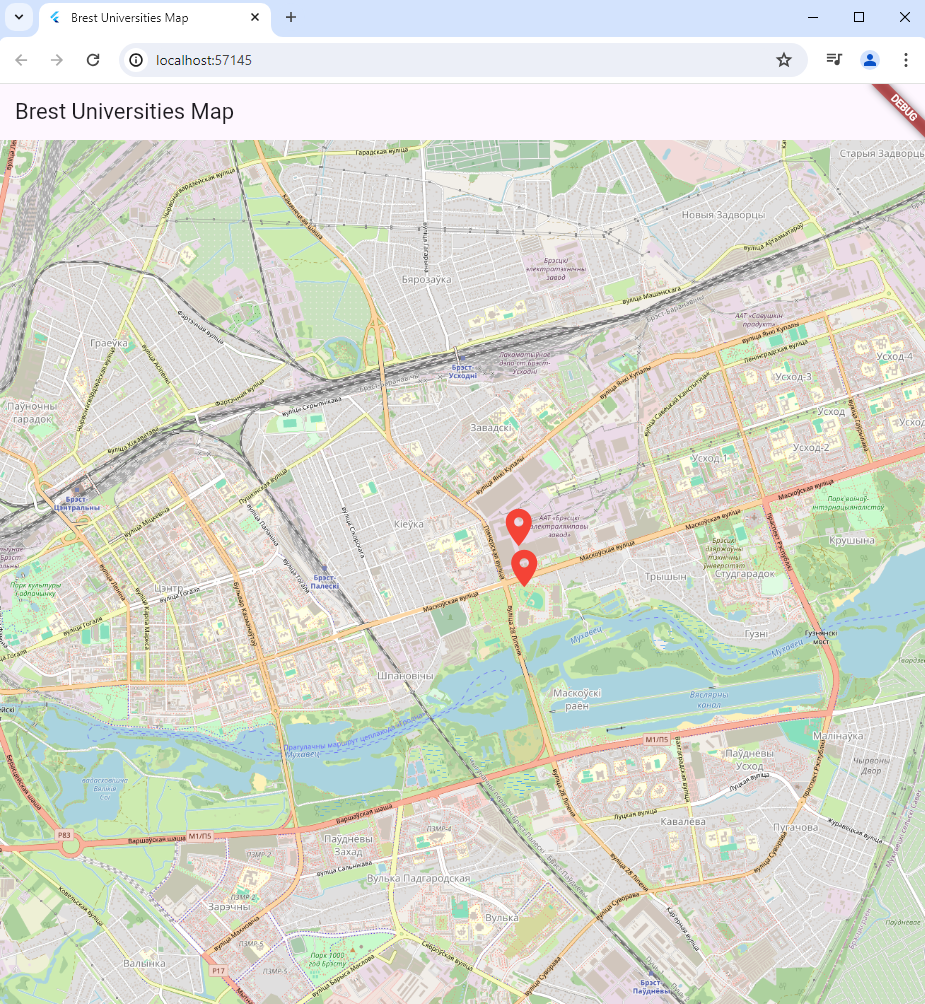
Приложение «Fluttery ToDo»

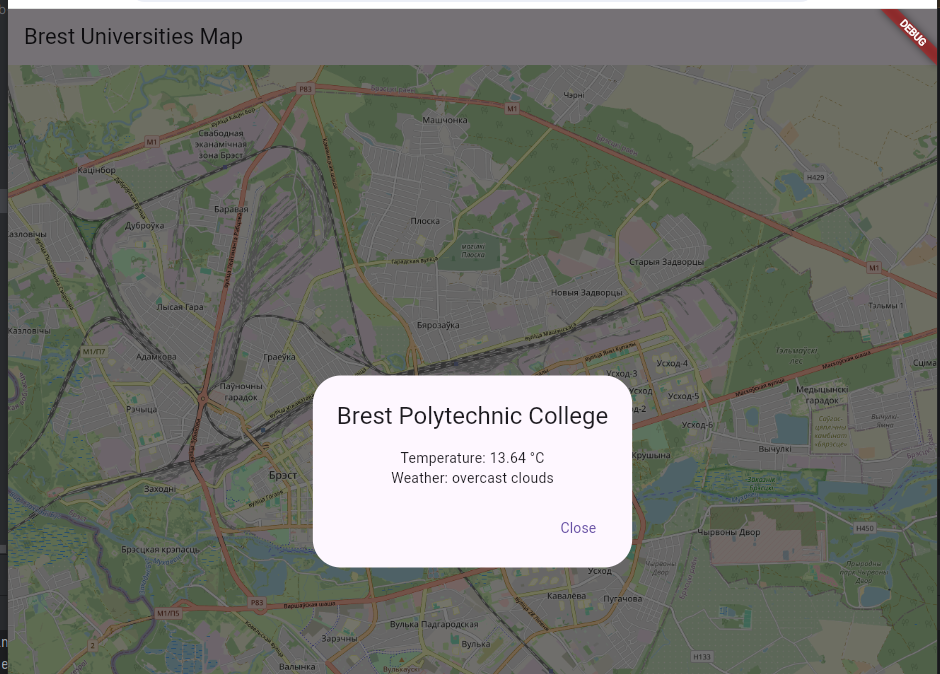


Задание 2





**Контрольные вопросы**

1. Что такое Flutter?

Flutter - это кроссплатформенный фреймворк для разработки мобильных приложений, созданный компанией Google. Flutter использует язык программирования Dart и позволяет создавать нативные приложения для iOS и Android из одного кодового базиса.

2. Что такое Dart и почему он используется во Flutter?

Dart - это объектно-ориентированный язык программирования, разработанный компанией Google. Он используется в Flutter, потому что Dart был специально создан для построения высокопроизводительных клиентских приложений. Dart обеспечивает быстрый компиляцию в нативный код, что позволяет Flutter создавать производительные и отзывчивые мобильные приложения.

3. Какие различные типы виджетов существуют во Flutter?

Во Flutter существует множество различных типов виджетов, включая:

* Базовые виджеты (например, Text, Image, Icon)
* Контейнерные виджеты (например, Row, Column, Stack)
* Виджеты управления состоянием (например, StatefulWidget, InheritedWidget)
* Виджеты макета (например, Padding, Center, Align)
* Виджеты взаимодействия (например, Button, TextField, Slider)
* Виджеты навигации (например, Navigator, Route, AppBar)
* Виджеты анимации (например, AnimatedContainer, AnimatedBuilder)

Эти различные типы виджетов позволяют разработчикам создавать сложные и функциональные пользовательские интерфейсы в Flutter.

4. В чем разница между StatelessWidget и StatefulWidget во Flutter?

StatelessWidget:

* Не изменяет своего внутреннего состояния после создания.
* Отрисовывается один раз и больше не обновляется, если не изменились входные параметры.
* Полезен для виджетов, которые не требуют обновления, например, иконок, текста, статических изображений.

StatefulWidget:

* Может изменять свое внутреннее состояние после создания.
* При изменении состояния, виджет перестраивается и отображается обновленная версия.
* Используется для виджетов, которым требуется динамическое обновление, например, счетчиков, форм, анимаций.

5. Для чего нужен ключ (Key) во Flutter?

Ключ (Key) во Flutter нужен для идентификации виджетов. Он помогает Flutter отслеживать изменения в дереве виджетов, что улучшает производительность при перестроении интерфейса.

6. Какие типы ключей используются во Flutter?

Flutter поддерживает несколько типов ключей:

* GlobalKey - уникальный глобальный ключ, который может быть использован для доступа к виджету из другой части приложения.
* ValueKey - ключ, который основан на значении, обычно используется для списков.
* ObjectKey - ключ, который основан на объекте, полезен для уникальной идентификации виджетов.
* UniqueKey - автоматически генерируемый уникальный ключ для временных виджетов.

7. В чем разница между MaterialApp и WidgetsApp во Flutter?

MaterialApp:

* Использует визуальные компоненты и тему Material Design, разработанную Google.
* Обеспечивает стандартный набор виджетов и поведение, соответствующее дизайну Material.
* Лучше подходит для приложений, ориентированных на материальный дизайн.

WidgetsApp:

* Использует более базовые виджеты и не применяет никакой платформенной темы по умолчанию.
* Предоставляет более низкоуровневый доступ к виджетам и управление состоянием приложения.
* Может быть полезен для создания кроссплатформенных приложений, которые не следуют строго определенной дизайн-системе.

8. Перечислите методы оптимизации приложений Flutter.

Вот некоторые методы оптимизации производительности во Flutter:

* Использование StatelessWidget вместо StatefulWidget, когда это возможно
* Применение ключей для идентификации виджетов
* Оптимизация построения списков с помощью ListView.builder()
* Использование PlatformView для встраивания нативных виджетов
* Ленивая загрузка ресурсов (изображения, видео)
* Кэширование данных и агрессивная оптимизация сетевых запросов
* Применение пакетов и библиотек для оптимизации
* Профилирование приложения и устранение "узких мест"

9. В чем разница между Navigator и Router во Flutter?

Navigator:

* Представляет собой виджет, который управляет навигацией между экранами приложения.
* Использует стек маршрутов для отслеживания и перемещения между экранами.
* Предоставляет набор методов, таких как push(), pop(), and replace(), для управления навигацией.

Router:

* Более высокоуровневый подход к навигации, основанный на URI-маршрутах.
* Использует сложную систему маршрутизации, которая позволяет создавать глубокие ссылки и поддерживать состояние экранов.
* Обеспечивает больше гибкости и контроля над навигацией в сложных приложениях.

10. Что такое State в Flutter?

State (состояние) в Flutter - это информация, которая определяет поведение и внешний вид виджета. Оно может быть:

Immutable (неизменяемое):

* Данные, которые не меняются после создания виджета, например, заголовок, икона.
* Хранится в полях виджета и передается через конструктор.

Mutable (изменяемое):

* Данные, которые могут меняться во время жизненного цикла виджета, например, счетчик, текстовое поле.
* Хранится в отдельном объекте State, связанном с виджетом.

11. Для чего используется параметр BuildContext в методе build() виджета во Flutter?

BuildContext во Flutter - это объект, который представляет местоположение виджета в дереве виджетов. Он используется:

* Для доступа к унаследованным от предков данным, таким как Theme, Localizations.
* Для вызова методов, связанных с текущим контекстом, например, Navigator.of(context).push().
* Для определения, как виджет должен быть построен и как он связан с другими виджетами.

12. В чем разница между методами push и pushReplacement во Flutter?

Методы push() и pushReplacement() во Flutter Navigator отличаются тем, как они изменяют стек маршрутов:

push():

* Добавляет новый маршрут в стек навигации.
* При возврате (pop()) пользователь вернется к предыдущему маршруту.
* Обеспечивает стандартную навигацию "вперед-назад".

pushReplacement():

* Заменяет текущий маршрут в стеке новым.
* При возврате (pop()) пользователь вернется к маршруту, предшествовавшему текущему.
* Используется, когда нужно полностью заменить текущий экран, не сохраняя его в стеке.

13. Для чего нужен инспектор виджетов во Flutter?

Инспектор виджетов (Widget Inspector) во Flutter - это мощный инструмент, который позволяет:

* Визуализировать дерево виджетов приложения в реальном времени.
* Получать информацию о свойствах и состоянии каждого виджета.
* Находить и исправлять проблемы с макетом и взаимодействием виджетов.
* Экспериментировать с различными виджетами и их свойствами.

14. Для чего нужен виджет MediaQuery во Flutter?

Виджет MediaQuery во Flutter предоставляет информацию об окружающих аппаратных характеристиках устройства, таких как:

* Размер экрана
* Ориентация экрана
* Плотность пикселей
* Безопасные области (где нет помех от системных элементов)
* Системные настройки (яркость, контраст, шрифт и т.д.)

Этот виджет помогает создавать отзывчивые и адаптивные пользовательские интерфейсы, которые выглядят и работают одинаково хорошо на разных устройствах. Он позволяет настраивать макет и поведение виджетов в зависимости от характеристик устройства.

15. Для чего нужен виджет SafeArea во Flutter?

Виджет SafeArea во Flutter обеспечивает безопасную область для размещения контента, избегая перекрытия системными элементами, такими как:

* Notch (вырез) на экране
* Системная панель навигации
* Системная строка состояния

Использование SafeArea важно для создания приложений, которые выглядят и работают корректно на разных устройствах, с разными формами экрана и системными интерфейсами. Он помогает разместить критически важный контент в видимой и доступной области экрана.

Применение SafeArea гарантирует, что пользовательский интерфейс не будет перекрыт или обрезан системными элементами, обеспечивая оптимальный опыт взаимодействия для пользователей.