**Рабочая тетрадь по курсу «Мобильное программирование на платформе Flutter»**

**Введение**

Рабочая тетрадь составлена на основе одноимённого курса и рассчитана на поддержку самостоятельной работы студентов. Задания позволяют развивать критическое мышление, укреплять теоретические знания и отрабатывать практические навыки программирования на языке Dart и платформе Flutter.

**Структура рабочей тетради:**

* **Часть I.** Теоретические задания и рабочие листы по каждому занятию курса.
* **Часть II.** Практические задания и индивидуальные проекты.
* **Часть III.** Контрольные вопросы и тесты для самоконтроля.

|  |  |
| --- | --- |
| Официальный сайт [Flutter](https://flutter.dev/) | Официальный сайт [Dart](https://dart.dev/) |

**Часть I. Теория и рабочие листы**

ЗАНЯТИЕ №1: ВВЕДЕНИЕ В МОБИЛЬНУЮ РАЗРАБОТКУ И ПЛАТФОРМУ FLUTTER. ЭКОСИСТЕМА FLUTTER, УСТАНОВКА SDK

**Цели занятия:** научиться устанавливать и настраивать Flutter SDK

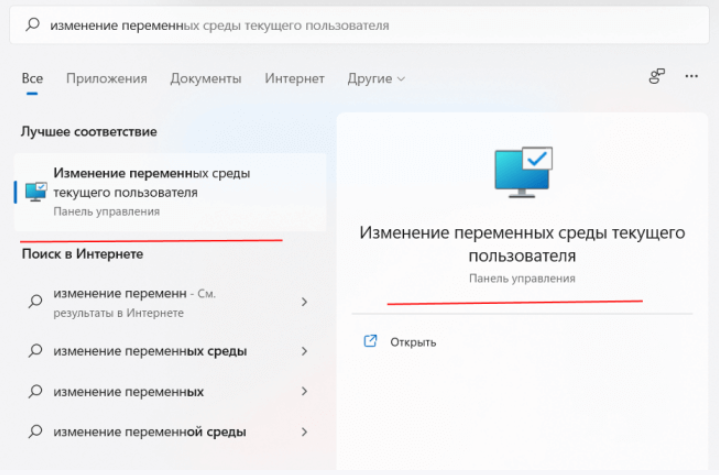
**Терминологический словарь:**

* **Flutter**: это фреймворк для создания красивых и производительных мобильных приложений на основе единого исходного кода для обеих платформ (Android и iOS). Его главное достоинство заключается в высокой производительности и возможности повторного использования кода, что снижает затраты на разработку.
* **Dart**: основной язык программирования, используемый в Flutter.
* **Cross-platform**: термин, обозначающий возможность разработки одного приложения сразу для нескольких платформ (например, Android и iOS).

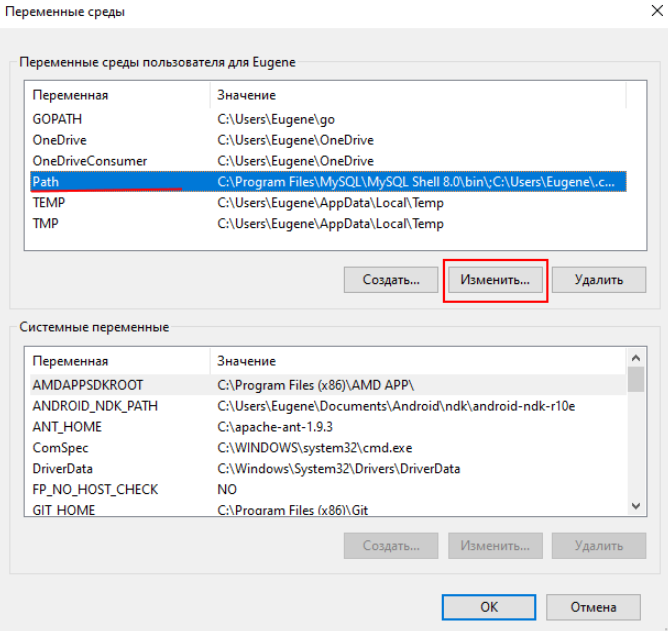
**Установка Flutter SDK и IDE**

**Алгоритм установки Flutter на Windows:**

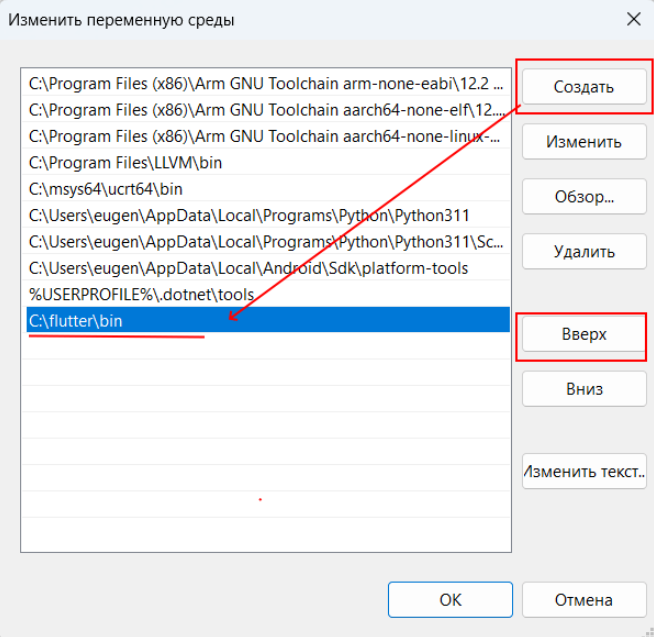
1. Скачать архив Flutter SDK с официального сайта: <https://docs.flutter.dev/get-started/install/windows/desktop?tab=download>.
2. Распаковать архив в удобное место на диске, рекомендуется такой путь: **C:\flutter**.
3. Добавить путь к каталогу bin в переменную окружения PATH.
   1. Для добавления переменной среды через поиск найдем найдем параметр "Изменение переменных среды текущего пользователя". Для этого введем в поле поиска "Изменение переменных":



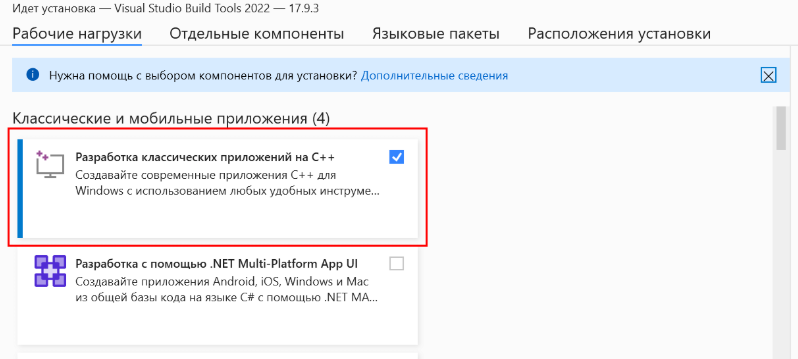
* 1. Выбрать пункт "Изменение переменных среды текущего пользователя".
  2. Затем откроется окно, где можно увидеть все переменные среды. (Также можно перейти через Параметры и пункт **Система -> Дополнительные параметры системы -> Переменные среды**).



* 1. Нажать на кнопку "Создать" и появившееся поле ввода ввести путь к папке bin из Flutter SDK:



1. Установить необходимые зависимости.
   1. Для установки Git for Windows, необходимо перейти по ссылке: <https://git-scm.com/downloads/win>
   2. Для установки пакета **Microsoft C++ Build Tools**, перейти по ссылке <https://visualstudio.microsoft.com/ru/visual-cpp-build-tools/> и нажать на кнопку «Загрузить Build Tools», далее выбрать «Разработка классических приложений на C++»



1. Проверить правильность установки командой:

flutter doctor

! Если необходимо установить обновление Flutter, то достаточно запустить в командной строке или терминале команду:

flutter upgrade

**Алгоритм установки Flutter на Linux:**

Дальнейшие действия будут происходить в терминале.

* 1. Установить Git:

sudo apt install git

* 1. Установить Flutter пакетным менеджером snap:

sudo snap install flutter --classic

* 1. Установить зависимости, вводя по очереди следующие команды:

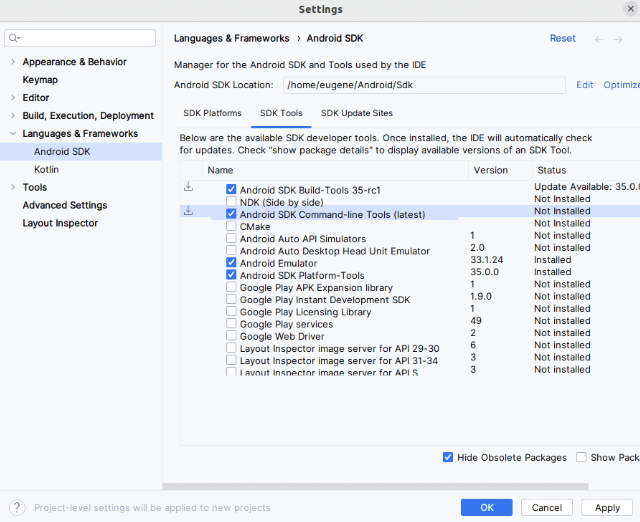
sudo apt install clang

sudo apt install cmake

sudo apt install ninja-build

sudo apt install libgtk-3-dev

* 1. Установить пакет **Android SDK Command Line Tools**. Этот можно сделать, загрузив пакет с официального сайта <https://developer.android.com/studio#command-line-tools-only>, либо через Android SDK Manager, который можно открыть в Android Studio:



* 1. Проверить правильность установки командой:

flutter doctor

**Полезные подсказки:**

* Во время установки внимательно читайте инструкции и устанавливайте рекомендованные пакеты.
* Следите за тем, чтобы ваша версия Android Studio соответствовала минимальным требованиям Flutter.

**Заполните чеклист ниже, отметив выполненные пункты.**

* Загружен Flutter SDK
* Добавлен путь к Flutter в PATH
* Установлены зависимые компоненты
* Проверен статус Flutter командой flutter doctor

ЗАНЯТИЕ №2: ОСНОВЫ ЯЗЫКА ПРОГРАММИРОВАНИЯ DART

**Цели занятия:**

* Изучить основы синтаксиса и конструкций языка программирования Dart.
* Получить навыки написания простых программ на Dart.
* Осознать связь между синтаксисом Dart и реализацией логики приложений на Flutter.

**Терминологический словарь:**

* **Dart**: мультипарадигменный язык программирования, разработанный компанией Google.
* **Переменная**: именованный участок памяти, предназначенный для хранения данных определённого типа.
* **Тип данных**: категория данных, определяющая допустимый диапазон значений и доступные операции.
* **Isolate:** самостоятельная единица выполнения, которая работает в отдельной области памяти. С помощью изолятов в Dart осуществляется параллельное выполнение задач.

**Теоретическая часть.**

**1. Что такое Dart?**

Dart — мультипарадигменный язык программирования общего назначения, разработанный компанией Google. Первоначально Dart задумывался как замена JavaScript, но позже стал известен благодаря своему применению в фреймворке Flutter для создания мобильных приложений.

Основные особенности Dart:

* **Строгая типизация**: позволяет избежать многих ошибок на этапе компиляции.
* **Однопоточность**: Dart работает в одном потоке, но поддерживает изоляцию потоков через isolate.
* **Синхронизация и асинхронность**: поддерживает оба режима выполнения операций.
* **Простота синтаксиса**: похож на JavaScript и Java, что облегчает вход в язык.

**2. Синтаксис Dart**

Dart обладает чистым и лаконичным синтаксисом, близким к другим популярным языкам, таким как Java и TypeScript.

Программа на языке программирования Dart всегда начинается с главной функции

void main() {}

**Приветственный пример ("Hello, World") на Dart:**

void main() {

print("Hello, World!");

}

**3. Конструкции языка**

**Переменные и типы данных**

Dart поддерживает строгую типизацию, позволяя указывать типы данных заранее:

// Целочисленные переменные

int a = 10;

// Числовые переменные с плавающей точкой

double b = 3.14;

// Логические переменные

bool isTrue = true;

// Строковые переменные

String greeting = "Привет!";

// Символьные переменные

char letter = 'a';

// Универсальный тип dynamic (используется редко)

dynamic unknownTypeVariable = null;

Более подробная информация содержится в примечании **№1** **Основные типы переменных в Dart/Flutter**

**Основные операторы и выражения**

**Присваивание**: $= (простое присваивание), +=, -=.

**Логические операторы**: && (логическое AND), || (логическое OR), ! (NOT).

**if-else**: условные операторы для принятия решений.

int num = 10;

if (num > 0) {

print("Число положительное");

} else if (num < 0) {

print("Число отрицательное");

} else {

print("Число равно нулю");

}

**for**: оператор цикла для многократного исполнения блока кода.

// Простой цикл for

for (int i = 0; i < 5; i++) {

print(i);

}

**while**: цикл с условием, проверяемым перед каждым исполнением тела цикла.

// while цикл

int j = 0;

while (j < 5) {

print(j++);

}

**Функции**

Функции в Dart определяются с помощью ключевого слова Function или анонимно:

// Обычная функция

void greet(String name) {

print("Привет, $name!");

}

// Функция с возвращаемым типом

int add(int a, int b) => a + b;

// Анонимная функция

var multiply = (int x, int y) => x \* y;

**4. Запуск первой программы**

Для запуска программы на языке программирования Dart необходимо воспользоваться Visual Studio Code или любым другим текстовым редактором.

Сначала создается файл с названием main и расширением .dart:



Далее, после написания программы и сохранения его в файл, необходимо запустить командную строку, перейти к расположению файла командой:

cd C:\путь\_к\_файлу

И запустить программу командой:

dart main.dart

**Практическое задание.**

**Задача:** определить четность введенного числа и вывести соответствующий результат.

void main() {

int number = 10;

if (number % 2 == 0) {

print('Число $number четное');

} else {

print('Число $number нечетное');

}

}

**Работа с функциями и классами**

**Понятие функций и классов:**

* **Функция**: независимый блок кода, принимающий аргументы и возвращающий результат.
* **Класс**: конструкция, содержащая поля и методы, описывающие сущность (объект).

**Упражнение:**

Создайте класс с именем Person, содержащий поля name и age, а также метод вывода информации о человеке.

class Person {

String name;

int age;

void info() {

print('Имя: ${this.name}, Возраст: ${this.age}');

}

}

void main() {

var person = Person();

person.name = 'Иван';

person.age = 30;

person.info(); // Имя: Иван, Возраст: 30

}

**Самостоятельная работа.**

1. Решите задачу: Напишите программу на Dart, которая выводит на экран последовательность чисел от 1 до 10 включительно.
2. Дополнительно (по желанию): дополните класс Person методом подсчета возраста в месяцах.

**Рефлексия**

Подведите итог сегодняшнего занятия, заполнив пропуски:

Сегодня я познакомился с \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.

Самым трудным оказалось \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.

Новое, что я вынес с занятия: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.

ЗАНЯТИЕ 3. ПЕРВОЕ ЗНАКОМСТВО С ФРЕЙМВОРКОМ FLUTTER

**Цели занятия:**

* Изучить основные концепции и структуру фреймворка Flutter.
* Познакомиться с различием между видами виджетов (StatelessWidget и StatefulWidget).
* Освоить создание простого Flutter-приложения с использованием базовых виджетов.

**Термины и понятия:**

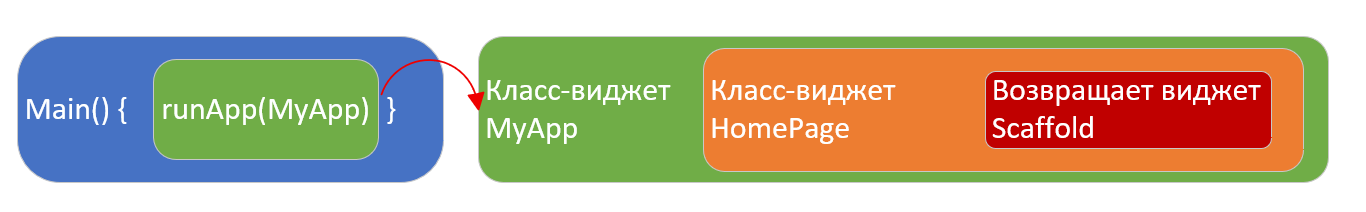
* **Flutter**: Cross-platform фреймворк для разработки мобильных приложений, основанный на языке программирования Dart.
* **Widget**: Основной строительный блок в Flutter, ответственный за рисование пользовательского интерфейса.
* **StatelessWidget**: Виджет, который не изменяется после инициализации.
* **StatefulWidget**: Виджет, имеющий изменяемое внутреннее состояние, которое может меняться во время работы приложения.

**Теоретическая часть**

**Виджеты и структура приложения**

В Flutter все элементы пользовательского интерфейса представляют собой виджеты.

Упрощенная структура приложения на Flutter выглядит так:



**Виджет Scaffold**

В большинстве случаев для отрисовки экрана приложения используется виджет **Scaffold**, так-как он представляет структуру и визуальную основу для экрана, чтобы разработчикам не приходилось писать все с нуля:

Scaffold(

appBar: ..., // Верхняя панель (заголовок, кнопки)

body: ..., // Основное содержимое экрана

floatingActionButton: ..., // Кнопка, "плавающая" внизу

bottomNavigationBar: ..., // Нижняя панель навигации

drawer: ..., // Меню сбоку (свайп или кнопка)

endDrawer: ..., // То же самое, но справа

bottomSheet: ..., // Нижняя панель, фиксированная или временная

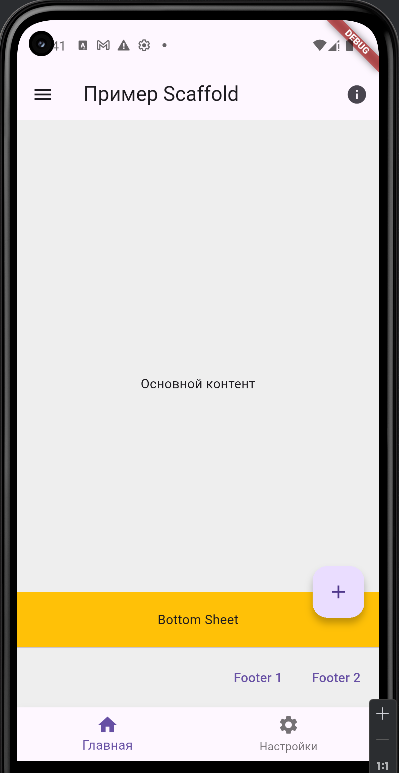
persistentFooterButtons: [...], // Постоянные кнопки внизу

backgroundColor: ..., // Цвет фона

resizeToAvoidBottomInset: true, // Авто-сдвиг при появлении клавиатуры

);

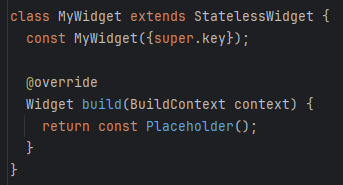
Ниже на скриншоте показан пример использования этого виджета, в котором отображены практически все его элементы:



Приложение использует **MaterialApp**, **Scaffold**, **AppBar** и **Text** — все это виджеты. Большинство из них — это StatelessWidget, потому что они не меняют свое состояние. Однако, если необходимо добавить интерактивность (например, счётчик нажатий), потребуется использовать StatefulWidget.

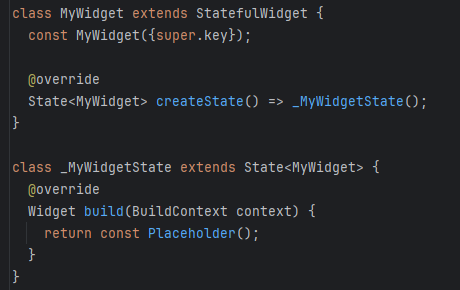
* **StatelessWidget**: Это неизменяемые виджеты, которые остаются постоянными после инициализации. Они используются для представления статичных частей интерфейса, таких как надписи, кнопки и изображения.

На скриншоте представлен класс MyWidget, который наследуется от StatelessWidget



* **StatefulWidget**: Такие виджеты содержат изменяемое состояние, что позволяет изменять содержимое виджета в ответ на действия пользователя или внешние факторы (например, изменение данных). Это может включать обработку взаимодействий с пользователем, получение данных с серверов и многое другое.

На скриншоте представлен класс MyWidget, который наследуется от StatefulWidget и класс состяние \_MyWidgetState который наследуется от State

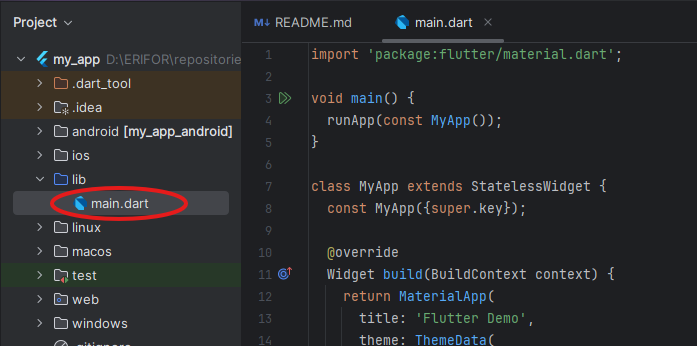


**Практическая часть**

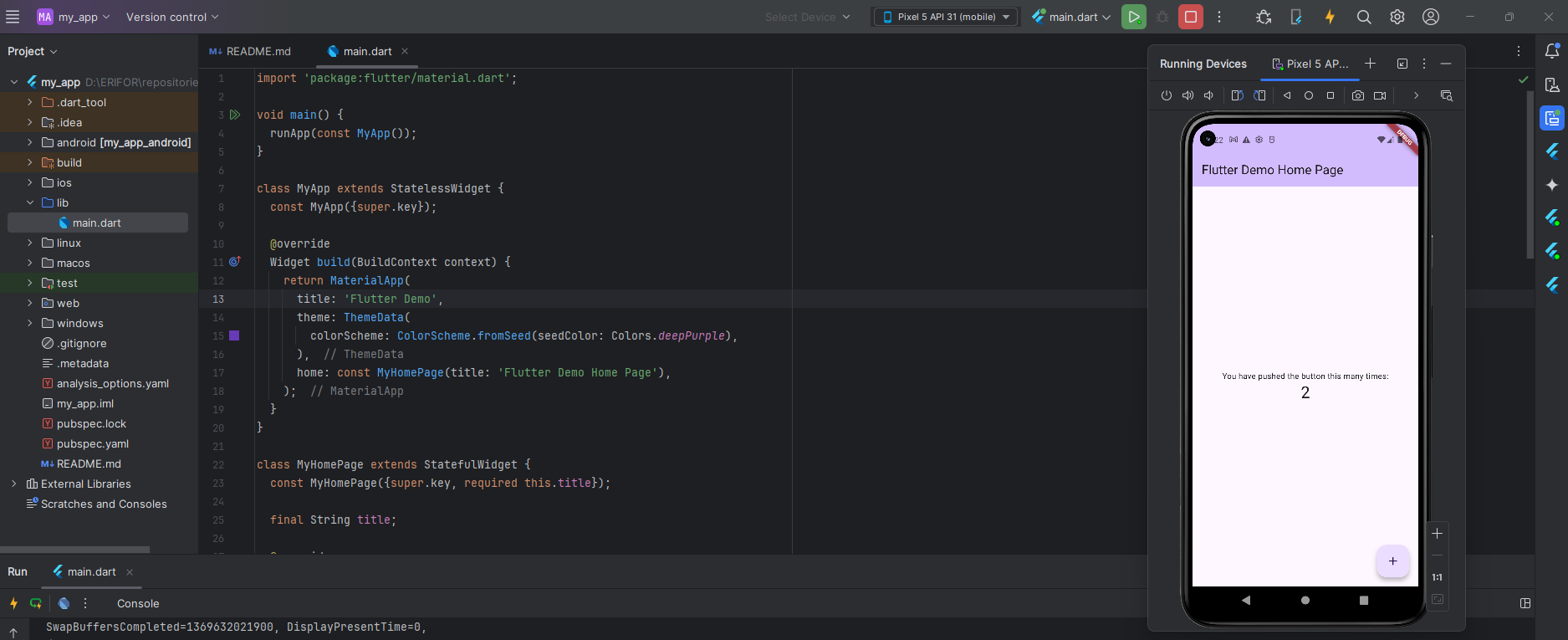
1. Запустите Android Studio.
2. Создайте новое Flutter-приложение. Для этого запустите новую команду в консоли:

flutter create my\_app

1. Откройте папку с создавшимся проектом в Android Studio, откройте файл main.dart:



1. Проанализируйте, зачем нужны StatelessWidget и StatefulWidget.
2. Запускаете эмулятор Android устройства. Если эмулятора нет, перейдите в раздел примечание №2 Создание эмулятора Android.
3. Дождавшись запуска эмулятора, запустите проект, нажав на кнопку Run:



**Самостоятельная работа**

1. Сделайте так, чтобы кнопка удваивала счетчик нажатий.
2. Поменяйте значок кнопки.

ЗАНЯТИЕ 4. ПОЛЬЗОВАТЕЛЬСКИЙ ИНТЕРФЕЙС (UI)

**Цели занятия:**

* Изучить устройство пользовательского интерфейса в Flutter.
* Познакомиться с основными контейнерами компоновки (layout widgets).
* Научиться строить собственные интерфейсы с использованием виджетов.

**Теоретическая часть**

**Виды компоновочных контейнеров (Layout widgets):**

В Flutter существует несколько типов контейнеров для размещения дочерних виджетов:

* **Row**: Горизонтальный контейнер, размещающий виджеты слева направо.
* **Column**: Вертикальный контейнер, располагающий виджеты сверху вниз.
* **Stack**: Накладывающийся контейнер, позволяющий размещать виджеты поверх друг друга.
* **Expanded/Flexible**: Растягиваются на все доступное ему пространство. Позволяют распределять пространство между детьми в контейнерах.
* **Container**: Общий универсальный контейнер, предоставляющий опции позиционирования, отступов, бордюра и фона.

**Различие между Row и Column:**

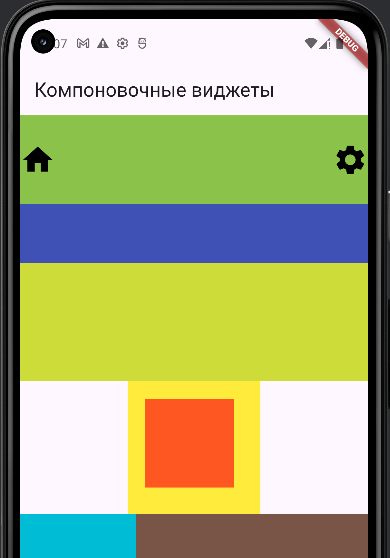
* **Row**: Расположенные горизонтально виджеты выстраиваются вдоль оси X.
* **Column**: Расположенные вертикально виджеты располагаются вдоль оси Y.

**Практическое задание:**

1. Перейдите по ссылке <https://github.com/therarErifor/workbook/blob/main/Листинги/listing_for_4_classes> или по QR-коду:



1. Скопируйте код, представленный на листинге, и вставьте в созданный проект вместо старого кода.
2. Запустите проект на эмуляторе и проанализируйте, какие виджеты и в какой иерархии используются в программе:



Каждый виджет окрашен для упрощения распознавания

**Самостоятельное задание:**

Создайте собственный пользовательский интерфейс. Ваш интерфейс может быть любым из предложенных вариантов:

* Калькулятор (simple calculator),
* Панель настроек (settings panel),
* Календарь,

Вам нужно спроектировать и реализовать интерфейс с использованием рассмотренных контейнеров компоновки и других нужных виджетов.

**Критерии оценки:**

* Грамотное использование контейнеров компоновки (Row, Column, Stack и т.д.)
* Ясность и эстетичность интерфейса.
* Правильное размещение виджетов относительно друг друга.
* Наличие комментариев в коде.

**Занятие 5. Навигация**

**Тестовые вопросы:**

1. Как организовать навигацию между страницами в Flutter?

**Занятие 6. Работа с асинхронностью**

**Индивидуальное задание:** Используя Future и async/await, напишите приложение, загружающее данные с сервера и обрабатывающее их.

**Занятие 7. Управление состоянием приложения при помощи BLoC**

**Решение проблем:**

1. Предложите оптимальный способ передачи данных между компонентами.

**Лабораторная работа:** Организуйте многопоточный доступ к состоянию приложения с использованием BLoC.

**Занятие 8. Доступ к данным**

**Практическое упражнение:** Создайте клиент-серверное приложение, выполняющее запросы к внешнему API и сохраняющее данные в локальной базе данных.

**Занятие 9. Локализация и международные рынки**

**Домашнее задание:** Добавьте в своё приложение поддержку английского и русского языков.

**Занятие 10. Работа с анимациями и графикой**

**Эскизы:** Придумайте концепцию собственного кастомного виджета с анимацией и реализуйте его.

**Занятие 11. Оптимизация производительности**

**Исследовательская работа:** Проведите профилирование производительности вашего приложения и предложите меры оптимизации.

**Занятие 12. Unit-тестирование**

**Задача:** Создайте набор тестов для проверки функциональности вашего приложения.

**Занятие 13. Подготовка к публикации**

**Подготовка:** Составьте инструкцию по упаковке и публикации приложения в Google Play и App Store.

**Занятие 14. Итоговый проект**

**Самостоятельная работа:** Подготовьте презентацию своего итогового проекта, защитите его перед преподавателем и группой.

**Часть II. Практические задания и индивидуальные проекты**

Каждый модуль сопровождается творческими заданиями, направленными на выработку уверенных навыков программирования и понимания внутренних механизмов работы Flutter и Dart.

**Часть III. Контрольные вопросы и тесты**

Заключительная часть предлагает студентам серию контрольных вопросов и тестов для самопроверки, что позволяет оценить степень овладения материалом каждого раздела.

**Заключение**

Рабочая тетрадь обеспечит глубокую проработку курса и создаст прочную основу для последующего развития в области мобильной разработки на платформе Flutter.

Примечания

1. **Основные типы переменных в Dart/Flutter:**

**Final**: Переменная, значение которой задаётся один раз и не изменяется впоследствии.

final String myName = 'Иван';

**Const**: Константная переменная, значение которой известно на этапе компиляции и остаётся неизменным.

const int pi = 3.14;

**Var**: Динамическая переменная, тип которой определяется автоматически на основе присвоенного значения.

var x = 10; // Тип будет int

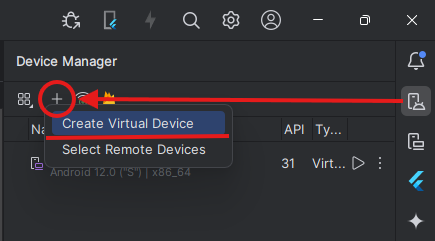
**Типизированные переменные**: Явно объявленные типы данных, такие как String, int, double и др.

double radius = 5.5;

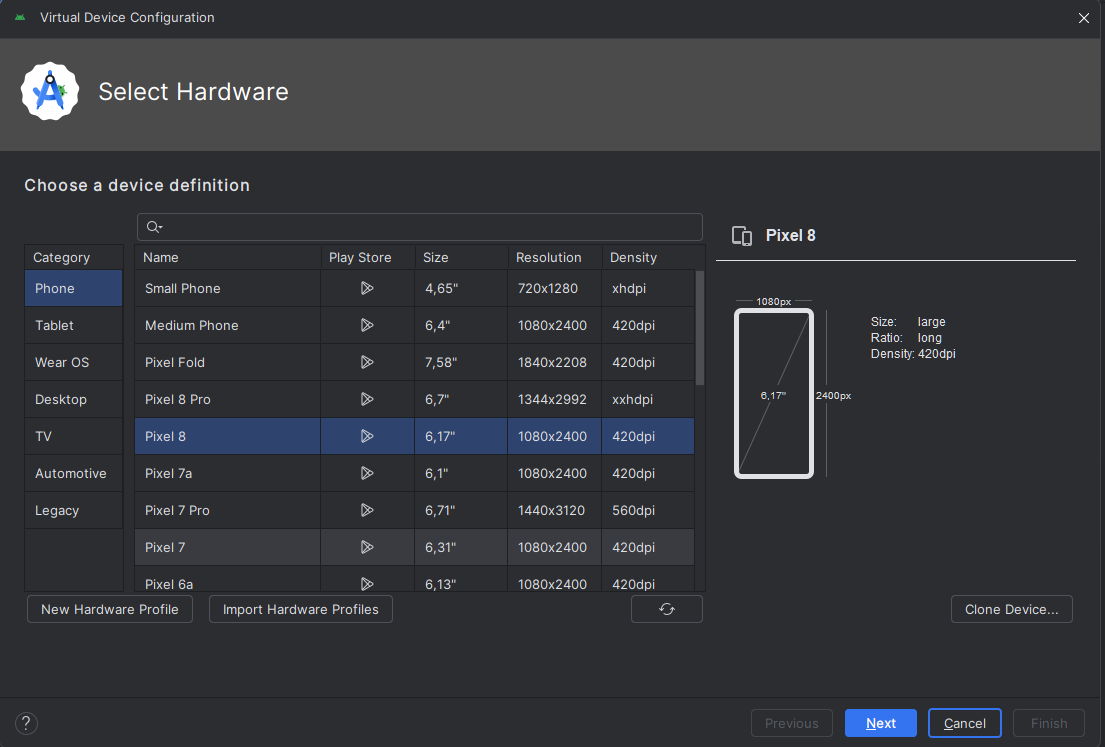
**Nullable variables**: Переменные, допускающие нулевое значение (null), используются вместе с оператором вопроса (?) и утверждением безопасности (!).

String? userInput; // Может содержать null

1. Создание эмулятора Android
2. нажмите на вкладку device manager, нажмите на кнопку с плюсом и выберите Create Virtual Device:



1. Выбираете нужное вам устройство. Главное учитывать соотношение и диагональ экрана эмулятора. Рекомендуется брать распространенный вариант диагонали примерно 6,7:



1. Выбираете рекомендуемые образы системы (system image), на скриншоте указан API 36. Если он не загружен, нажимаете на кнопку загрузки (выделено красным), после загрузки нажимаете Далее.
2. После этого оставляете все по умолчанию и нажимаете Finish