|  |  |
| --- | --- |
| ДИСЦИПЛИНА | Программирование корпоративных систем |
| ИНСТИТУТ | Институт перспективных технологий и индустриального программирования |
| КАФЕДРА | Кафедра индустриального программирования |
| ВИД УЧЕБНОГО МАТЕРИАЛА | Практические задание |
| ПРЕПОДАВАТЕЛЬ | Адышкин Сергей Сергеевич |
| СЕМЕСТР | 5 семестр, 2025-2026 гг. |

**Практическое занятие №1**

**Установка инструментария и настройка среды разработки мобильных приложений. Настройка режима терминала.**

**Цели занятия**

1. Освоить процесс установки Flutter SDK на разных операционных системах (Windows, macOS, Linux).
2. Настроить Android Studio для работы с Android SDK и эмулятором.
3. Подключить и использовать VS Code как основную IDE для Flutter.
4. Научиться работать с терминалом для проверки установки и запуска приложений.
5. Создать и запустить первое приложение Flutter.

## Теоретическая часть

Современная мобильная разработка представляет собой один из наиболее востребованных и быстроразвивающихся сегментов IT-индустрии. Миллионы пользователей ежедневно взаимодействуют с приложениями на смартфонах и планшетах, что создаёт высокий спрос на специалистов, умеющих создавать удобные, производительные и кроссплатформенные решения.

**Flutter** — это открытый SDK от компании Google, позволяющий разрабатывать приложения под Android, iOS, Web и даже настольные платформы, используя единый язык программирования Dart и общую кодовую базу. Такой подход значительно снижает затраты времени и ресурсов, что делает Flutter популярным инструментом как среди крупных компаний, так и среди независимых разработчиков и стартапов.

Однако работа с Flutter невозможна без правильно настроенной среды разработки. Для этого требуется:

* **Flutter SDK**, включающий в себя компиляторы, библиотеки и инструменты командной строки.
* **Android SDK и эмулятор**, поставляемые вместе с Android Studio, позволяющие запускать и тестировать приложения без физического устройства.
* **Среда разработки (IDE)** — в нашем курсе основным инструментом будет Visual Studio Code, которая отличается лёгкостью, высокой скоростью работы и гибкой системой расширений. Android Studio используется только как поставщик SDK и эмуляторов.

Особое значение имеет работа в **терминале**. Хотя графическая оболочка IDE делает разработку удобнее, терминал является обязательным инструментом профессионального разработчика. Через него можно:

* диагностировать состояние системы и корректность установки (flutter doctor),
* управлять устройствами (flutter devices),
* запускать и отлаживать приложения (flutter run),
* управлять зависимостями и сборкой (flutter pub get, flutter build).

В рамках данного занятия студенты получат навыки:

* установки и настройки Flutter SDK под разные операционные системы (Windows, macOS, Linux),
* настройки VS Code и Android Studio,
* проверки работы SDK и устройств через терминал,
* запуска первого тестового проекта.

Эти шаги являются фундаментом, без которого дальнейшее освоение мобильной разработки на Flutter невозможно. Понимание того, как устроена рабочая среда, как настраиваются переменные окружения и каким образом терминал управляет процессами разработки, позволит студентам уверенно работать с более сложными инструментами в будущих занятиях.

**Практическая часть**

**Общий алгоритм выполнения работы**

1. Ставим Flutter SDK → добавляем в PATH.
2. Ставим Android Studio → докачиваем Android SDK + эмулятор (AVD).
3. Ставим VS Code + расширения Flutter/Dart.
4. Принимаем Android-лицензии → flutter doctor без ошибок.
5. Запускаем эмулятор/подключаем телефон → flutter devices.
6. Создаём проект → flutter run → проверяем hot reload.

**Контрольная точка 0:** Понимание плана. Дальше идём по своей ОС.

**1) Подготовка ОС и терминала**

**1.1 Windows 10/11**

* **Терминал:** PowerShell (рекомендуется) или VS Code Terminal. Git Bash — по желанию.
* **Виртуализация:** зайти в BIOS/UEFI и включить *Virtualization* (Intel VT-x / AMD-V).
* **Ускорение эмулятора:** в Windows компонентах включить
  + *Virtual Machine Platform*
  + *Windows Hypervisor Platform*

Перезагрузиться.

* **Проверка:**
* systeminfo | findstr /I "virtualization"

Должно быть **Enabled**.

**1.2 macOS (Intel/Apple Silicon)**

* Терминал: стандартный **Terminal** или **iTerm2**; в VS Code — встроенный терминал.
* Виртуализация активна по умолчанию (Hypervisor.framework).

**1.3 Linux (пример Ubuntu/Debian)**

* Терминал: **bash**/**zsh** (VS Code Terminal допустим).
* Проверка поддержки виртуализации:
* egrep -c '(vmx|svm)' /proc/cpuinfo

Число > 0 = ок.

* Установка KVM/QEMU и добавление пользователя в группу:

sudo apt update

sudo apt install qemu-kvm libvirt-daemon-system libvirt-clients bridge-utils

sudo usermod -aG kvm $USER

sudo usermod -aG libvirt $USER

Перелогиниться/перезайти в сессию.

**Контрольная точка 1:** виртуализация включена; терминал выбран и рабочий.

**2) Установка Flutter SDK и добавление в PATH**

**Установка Flutter SDK**

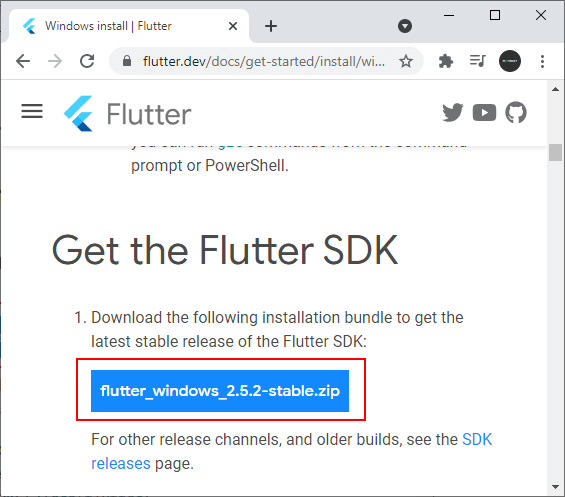
На странице <https://flutter.dev/docs/get-started/install/> можно найти ссылки на на загрузку SDK для разных операционных систем. Рассмотрим, как установить Flutter SDK на Windows и MacOS.

**Установка Flutter на Windows**

Чтобы начать разрабатывать приложения с помощью Flutter, необходимо установить SDK. Но прежде всего следует отметить, что для работы Flutter SDK в Windows в системе должен быть установлен Git for Windows, который можно найти по адресу <https://git-scm.com/download/win>.

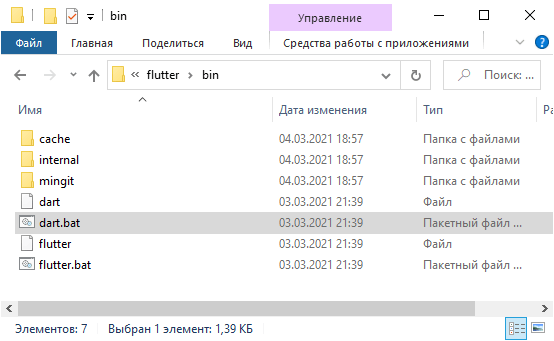
Также Flutter использует Android SDK. Самый легкий способ установить Android SDK - сразу установить Android Studio, вместе с которой будут установлены вс необходимые инструменты.

Непосредственно для установки Flutter SDK перейдем на страницу <https://flutter.dev/docs/get-started/install/windows>. На этой странице найдем в секции Get the Flutter SDK ссылку на zip-архив с Flutter SDK и загрузим его:

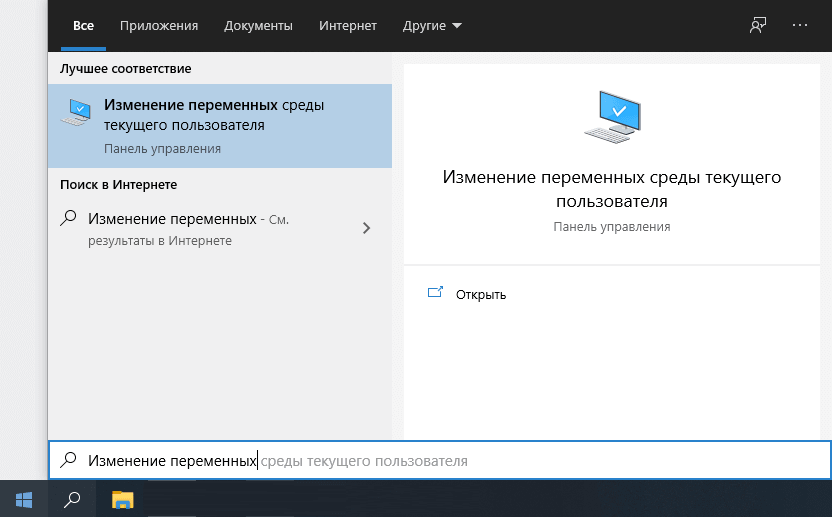


Далее распакуем архив, например, на диске С. Например, в моем случае архив распакован в папку C:\flutter.

В распакованном архиве в папке flutter\bin мы найдем инструменты для компиляции приложения. Например, в моем случае полный путь к этой папке C:\flutter\bin:

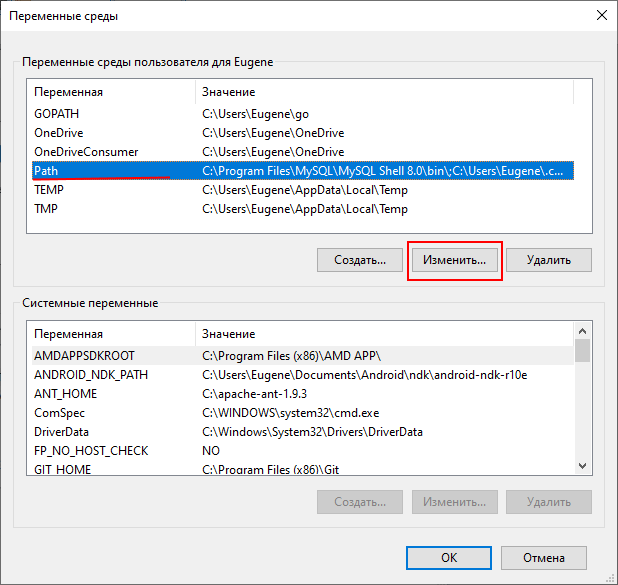


Если мы работаем в Windows, то для добавления переменной среды через поиск найдем параметр "Изменение переменных среды текущего пользователя". Для этого введем в поле поиска "Изменение переменных":.

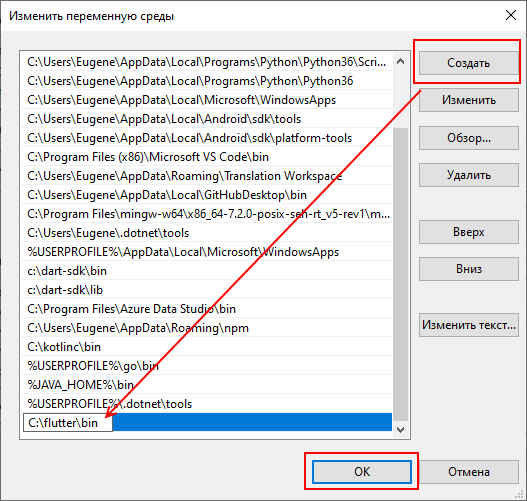


Выберем пункт «Изменение переменных среды текущего пользователя». Затем нам откроется окно, где мы можем увидеть все переменные среды. (Также можно перейти через Параметры и пункт Система ->Дополнительные параметры системы ->Переменные среды)

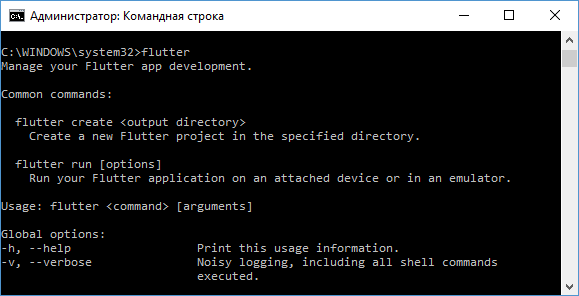
Здесь нам надо изменить переменную Path, добавив путь к папке bin в Flutter SDK. Для этого выберем пункт Path и нажмем на кнопку "Изменить":



Далее нажмем на кнопку «Создать» и появившееся поле ввода введем путь к папке bin из Flutter SDK:



Чтобы проверить корректность установки Flutter, откроем командную строку и введем команду flutter. Если Windows распознает ее, и последует вывод некоторой справочной информации (например, как использовать те или иные команды в консоли), то flutter установлен и настроен.



В качестве альтернативы установить Flutter SDK можно через Git. Для этого необходимо в командной строке перейти к папке, где должен располагаться SDK и далее выполнить в командной строке следующую команду:

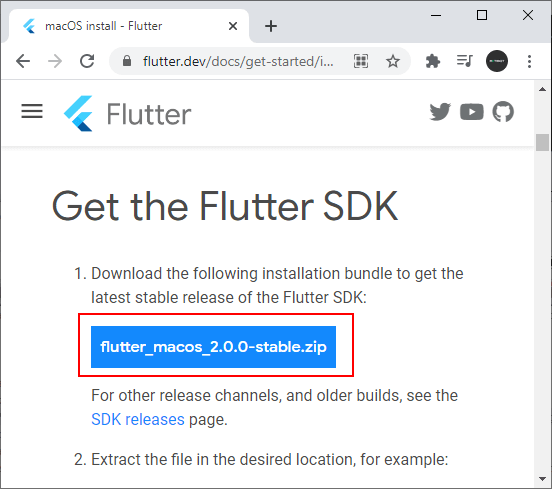
git clone -b stable https://github.com/flutter/flutter.git

В данном случае с репозитария на гитхабе будет загружена последняя стабильная версия Flutter SDK. И опять же подчеркиваю, что для выполнения этой команды необходимо установить Git for Windows, как было написано выше.

**Установка Flutter на MacOS**

Для установки и обновления Flutter использует git, поэтому необходимо установить git. Однако если установлен Xcode, то git устанавливать не надо, так как Xcode уже включает git.

Для установки Flutter SDK перейдем на страницу <https://flutter.dev/docs/get-started/install/macos> и найдем на странице ссылку на Flutter SDK и загрузим его:



Далее поместим папку с Flutter SDK где-нибудь на жестком диске. Например, в моем случае Flutter SDK расположен в корневой папке текущего пользователя (Users/eugene).

Для использования Flutter SDK в треминале, необходимо добавить путь к папке bin в SDK в системные переменные. Добавление для текущего окна терминала:

export PATH="$PATH:[Путь к папке с Flutter SDK]/flutter/bin"

При первом использовании команды flutter произойдет загрузка Dart SDK.

**Обновление Flutter**

Если впоследствии необходимо будет обновить Flutter, то для этого достаточно будет выполнить в командной строке команду

flutter upgrade

Поскольку Flutter активно развивается, то лучше обновляться регулярно.

**Контрольная точка 2:** команда flutter --version выдаёт версию без ошибок.

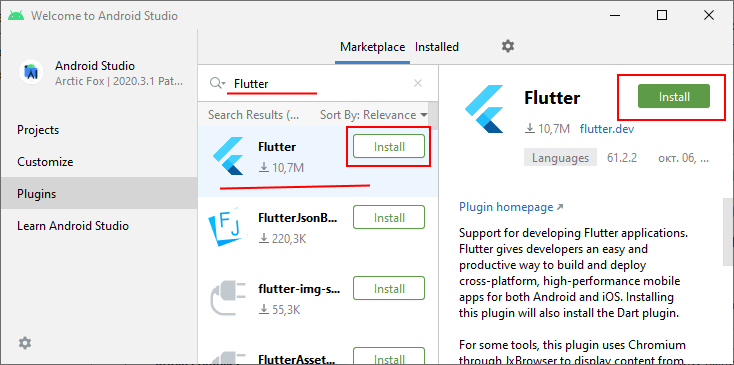
**3) Установка Android Studio и Android SDK/инструментов**

**3.1 Установка Android Studio (все ОС)**

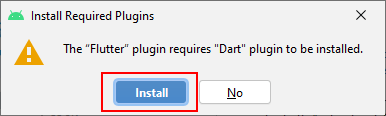
Для разработки под Flutter нередко выбирается такая среда разработки как Android Studio. Хотя мы можем набирать код и в простейшем текстовом редакторе и компилировать его в консоли, но среда разработки существенно позволяет упостить процесс написания и построения приложения. Причем Android Studio позволяет создать приложения на Flutter не только собственно под Android, но и под другие поддерживаемые платформы.

Для работы с Android Studio ее естественно вначале надо установить. Инсталлятор можно загрузить по ссылке <https://developer.android.com/studio>.

По умолчанию Android Studio не поддерживает Flutter, поэтому нам надо установить соответствующий плагин. Для этого в Android Studio на стартовом экране выберем пункт Plugins (либо в открытой студии перейдем в меню **File -> Settings** и далее в открывшемся окне также выберем пункт **Plugins**). И в панели плагинов найдем плагин **Flutter**:



Для упрощения поиска нужного плагина мы можем ввести в поисковую стоку слово "Flutter", и первый результат будет как раз тем, который надо установить. При установке плагина также отобразится окно с предложением установить плагин для Dart. Также нажмем на ОК для его установки:



После установки плагина необходимо будет перезагузить Android Studio.

**3.2 Выбор и установка системного образа**

* **AVD Manager → Create Virtual Device →** выберите, например, *Pixel 7*.
* System Image: последняя стабильная (Android 14/15), ABI: x86\_64/arm64-v8a (на Apple Silicon — arm64).
* Завершите создание AVD.

**3.3 Аппаратное ускорение эмулятора**

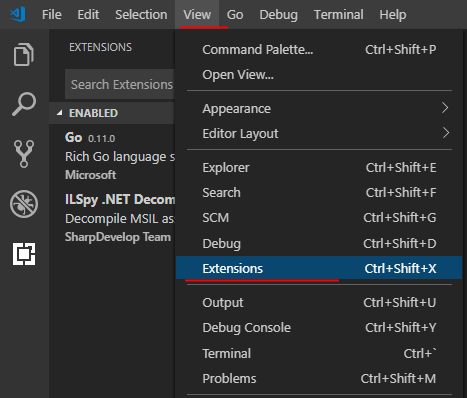
* **Windows:** при включённых *Virtual Machine Platform* и *Windows Hypervisor Platform* эмулятор использует WHPX/Hyper-V.
* **macOS:** используется Hypervisor.framework автоматически.
* **Linux:** эмулятор использует KVM. Если не стартует — проверьте группы kvm/libvirt.

**Контрольная точка 3:** в AVD Manager есть хотя бы одно виртуальное устройство.

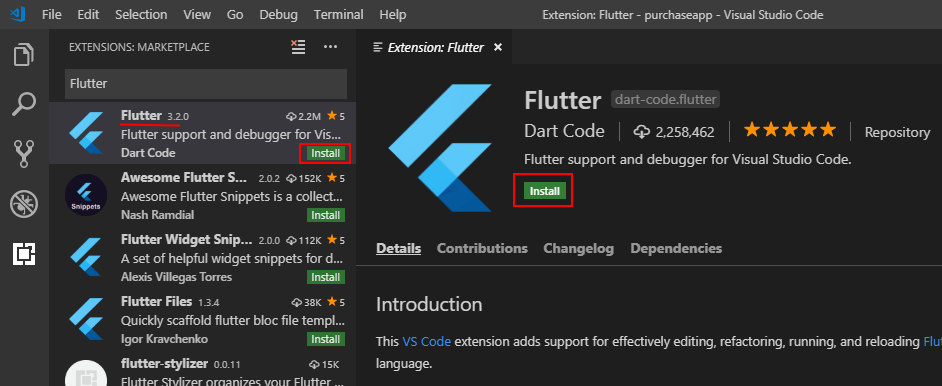
**4) Установка и настройка VS Code**

**4.** Одним из популярнейших текстовых редакторов на сегодняшний день является Visual Studio Code. И Google предоставляет для него плагин для разработки под Flutter.

Для его установки перейдем к пункту меню **View -> Extensions.**



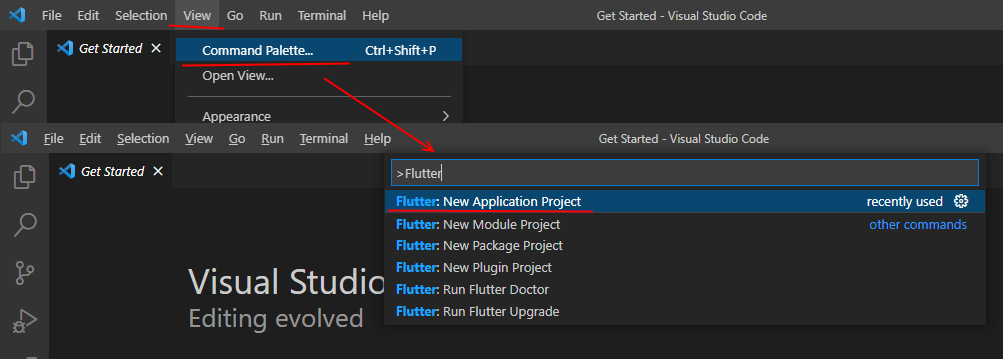
После этого откроется панель расширений. Введем в поле поиска слово "Flutter":



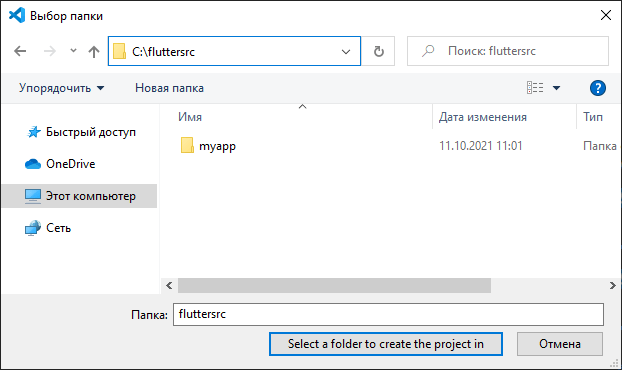
Нужный плагин носит имя "Flutter" (как правило, первый результат среди результатов поиска). И для его установки нажмем на кнопку Install.

Поскольку этот плагин зависит от плагина для Dart, то плагин Dart также автоматически будет установлен.

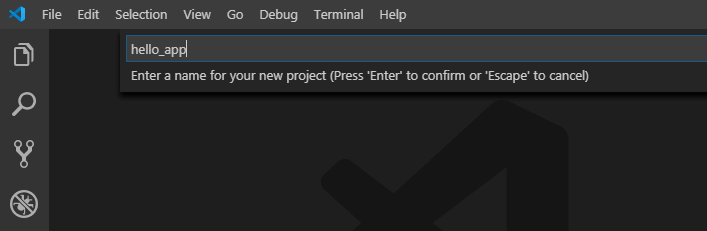
После установки плагина создадим первый проект. Для этого перейдем к пункту меню **View -> Command Palette**. В открышееся окно поиска введем слово "flutter" и среди результатов выберем **Flutter: New Application Project**.



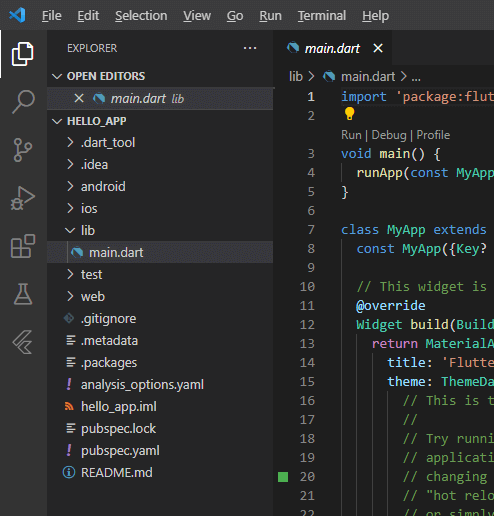
Затем в новом окне выберем папку, где будет располагаться проект



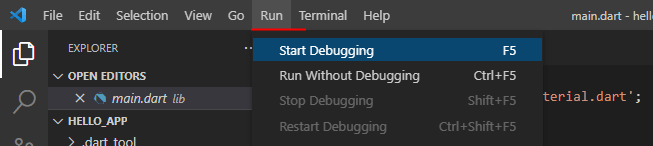
Затем в новом окне введем название проекта, например, "hello\_app" и введем Enter



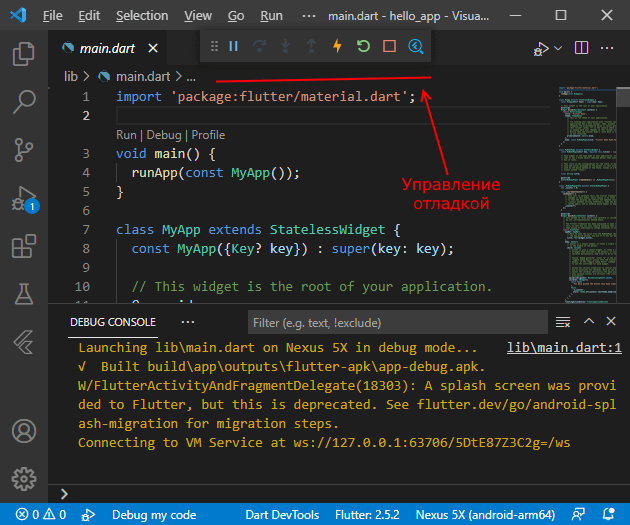
В итоге будет создан проект с той же структурой, что и в предыдущих статьях. А в текстовом редакторе будет открыт файл **main.dart**:



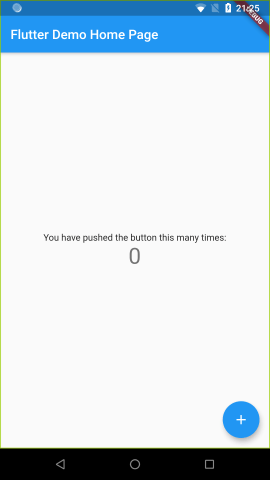
Для запуска приложения подключим мобильное устройство с Android к компьютеру, а в Visual Studio Code перейдем к пункту меню **Run -> Start Debugging**. (Если отладка нам неважна, и мы хотим просто запустить проект, то выбирается пункт **Run Without Debugging**)



В результате будет построено приложение. Сверху VS Code отобразится панель для управления отладкой, а снизу отобразится вся отладочная информация.



А на мобильном устройств будет запущено приложение.



**Контрольная точка 4:** расширения Flutter/Dart активны, терминал в VS Code работает.

**5) Принятие Android-лицензий и диагностика**

В терминале вашей ОС:

flutter doctor

Если видите предупреждение Android license status unknown:

flutter doctor --android-licenses

Нажимайте y для принятия. Снова:

flutter doctor

Стремимся к состоянию без ошибок (warnings могут остаться, например по отсутствии iOS на macOS — это допустимо, если работаем только с Android).

**Контрольная точка 5:** flutter doctor без критических ошибок.

**6) Запуск эмулятора и/или подключение физического устройства**

**6.1 Эмулятор (все ОС)**

* Откройте Android Studio → **AVD Manager** → **Run** напротив устройства.
* Дождитесь загрузки Android.

**6.2 Физический телефон (опционально)**

* Включите **Для разработчиков** и **USB-отладку** на устройстве.
* **Windows:** при необходимости поставьте драйвер (Google USB Driver через SDK Manager или фирменный от производителя).
* Подключите по USB, разрешите RSA-ключ на телефоне.

**6.3 Проверка из Flutter**

flutter devices

Должно показать emulator-XXXX или модель телефона и платформу android.

**Контрольная точка 6:** в flutter devices есть хотя бы одно доступное устройство.

**7) Создание и старт первого проекта**

**7.1 Через терминал**

flutter create hello\_flutter

cd hello\_flutter

flutter run

* Убедитесь, что выбрано нужное устройство (VS Code → статус-бар → устройство).
* Приложение откроется на эмуляторе/телефоне.

**7.2 Горячая перезагрузка и перезапуск**

* В запущенном flutter run нажмите r — **hot reload** (мгновенно применяет изменения UI и кода без полной перезагрузки).
* Нажмите R (Shift+R) — **hot restart** (перезапуск с начальным состоянием).

**7.3 Запуск из VS Code (альтернатива)**

* Откройте папку проекта → **Run and Debug (F5)** → выберите *Dart & Flutter*.
* В нижнем баре выберите устройство (эмулятор/телефон).

**Контрольная точка 7:** шаблонное приложение Flutter запускается, hot reload работает.

**8) Полезные операции в терминале (ежедневная практика)**

* Обновить зависимости:

flutter pub get

* Очистить проект (когда «ломается» сборка):

flutter clean

flutter pub get

* Посмотреть подключённые устройства:

flutter devices

* Подробный лог запуска (для диагностики Gradle/SDK):

flutter run -v

**9) Сборка установочного файла (Android)**

**9.1 APK (debug/release)**

flutter build apk --debug

flutter build apk --release

APK появится в build/app/outputs/flutter-apk/.

**9.2 App Bundle (для публикации в Play)**

flutter build appbundle --release

Файл: build/app/outputs/bundle/release/app-release.aab.

**Контрольная точка 8:** успешная сборка apk (release/или debug).

**10) Частые проблемы и быстрые решения**

**10.1 Общие**

* **flutter не найден:** проверьте PATH, закройте/откройте терминал; на Unix — проверьте, что добавили путь в правильный rc-файл и выполнили source ~/.bashrc/~/.zshrc.
* **Android license status unknown:** выполните flutter doctor --android-licenses.
* **No connected devices:** убедитесь, что эмулятор запущен; для телефона — включена USB-отладка и принято уведомление «Allow USB debugging».
* **Эмулятор зависает/не стартует:** перезагрузите AVD (*Cold Boot*), удалите и создайте заново системный образ, обновите Emulator из SDK Manager.

**10.2 Windows**

* **Конфликт Hyper-V/HAXM:** используйте *Windows Hypervisor Platform* (современный путь). Убедитесь, что включены *Virtual Machine Platform* и *Windows Hypervisor Platform*.
* **Нет драйвера телефона:** поставьте *Google USB Driver* (SDK Manager → SDK Tools) или драйвер производителя.
* **Gradle кеш сломан:** flutter clean → удалите %USERPROFILE%\.gradle\caches → flutter pub get → flutter run.

**10.3 macOS**

* **Нужен iOS (опционально):** установите Xcode из App Store → запустите Xcode один раз → sudo xcodebuild -license → sudo xcode-select -s /Applications/Xcode.app. (Это **не обязательно** для Android-таргета.)
* **Права на каталоги SDK:** убедитесь, что Flutter распакован в каталог пользователя, где у вас есть права записи.

**10.4 Linux (Ubuntu/Debian)**

* **Нет OpenGL/GLU для эмулятора:**

sudo apt install libglu1-mesa

* **Не добавлен в kvm:**

groups $USER # посмотрите группы

sudo usermod -aG kvm $USER

Перелогиниться.

* **Зависимости для Flutter tooling (часто полезно):**

sudo apt install clang cmake ninja-build pkg-config libgtk-3-dev liblzma-dev

**11) Что именно сдаём (контроль на каждом этапе)**

1. Скрин flutter doctor после принятия лицензий.
2. Скрин запущенного эмулятора (AVD Manager) **и/или** вывод flutter devices.
3. Скрин работающего приложения **hello\_flutter** на эмуляторе/телефоне.
4. Текстом: ОС, используемый терминал, какие проблемы встретились и как решили (2–5 пунктов).
5. (Опционально) Путь, который вы добавили в PATH, и вывод:

where flutter # Windows

which flutter # macOS/Linux

**12) Быстрая проверка (чек-лист)**

* flutter --version и flutter doctor без критичных ошибок.
* Есть AVD и он запускается.
* flutter devices показывает устройство.
* Проект создаётся и запускается; hot reload работает.
* Собран apk (debug или release).

**Полезные команды для терминала**

* flutter doctor — диагностика окружения.
* flutter devices — список подключённых устройств.
* flutter run — запуск приложения.
* flutter clean — очистка кеша проекта.
* flutter build apk — сборка APK для Android.
* flutter pub get — установка зависимостей.

**Контрольные задания**

1. Сделать скриншот вывода flutter doctor.
2. Сделать скриншот списка устройств после flutter devices.
3. Сделать скриншот первого приложения **hello\_flutter**.
4. Указать в отчёте:
   * операционную систему (Windows, macOS, Linux),
   * используемый терминал (PowerShell, Git Bash, zsh, bash, VS Code Terminal).

**Требования к отчёту**

* Отчёт должен содержать скриншоты всех контрольных этапов.
* Текстовый файл отчёта: 3–5 страниц.
* В конце — краткое описание encountered проблем и способов их решения.