Flutter - структура проекта

Dart — язык программирования, разработанный компанией Google, а Flutter — кросс-платформенный UI-фреймворк, написанный на языке Dart. Этот инновационный фреймворк позволяет разрабатывать приложения, которые могут работать на шести платформах: Android, iOS, Linux, MacOS, Web и Windows.
Благодаря этому разработчики экономят время и ресурсы: не нужно создавать и поддерживать отдельные версии приложений для каждой платформы.

Установка Flutter

Начать писать на Flutter легко — достаточно установить необходимые инструменты по шагам:

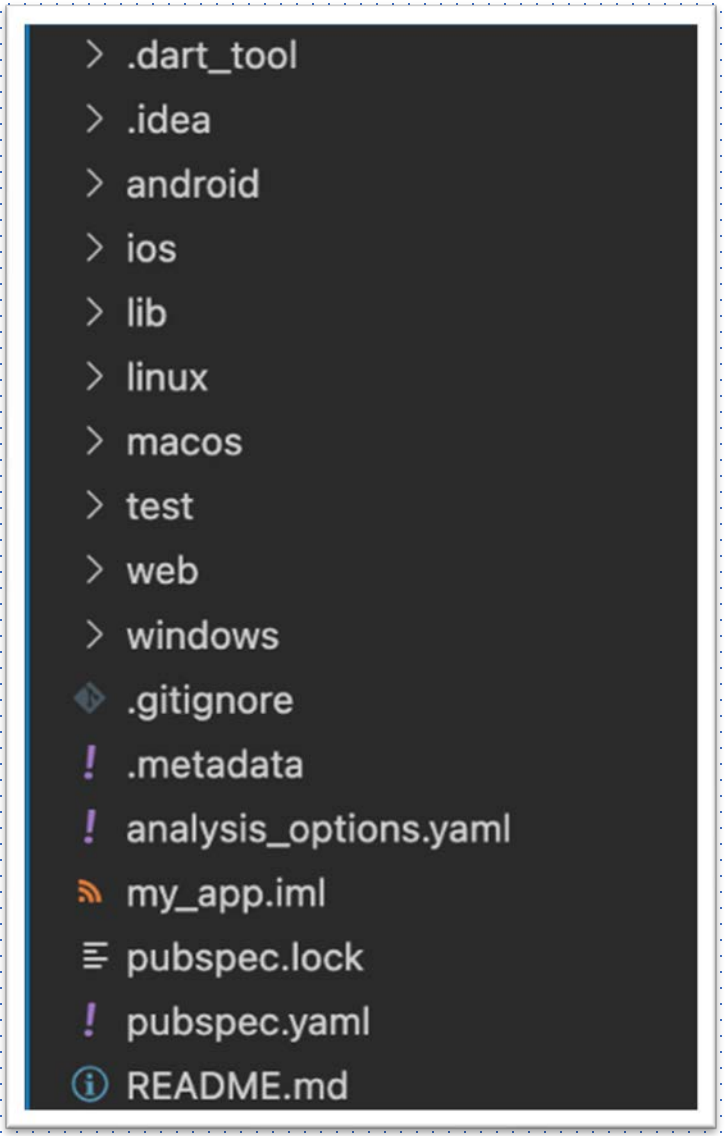
1. Зайти на docs.flutter.dev.
2. Установить окружение и инструменты по инструкции.
3. Выбрать удобную среду разработки.
4. Создать проект с помощью команды flutter create my_app — приложение готово.

pub

В лекциях часто будет упоминаться инструмент командной строки pub, используемый для управления зависимостями и пакетами в Dart и Flutter-проектах. Он входит в состав Dart SDK и предоставляет удобные средства для разработки приложений. Запуск любого проекта начинается с команды flutter pub get, которая устанавливает необходимые зависимости, в том числе и Flutter.

Структура проекта

- Проекты во Flutter делятся на два типа: приложения и библиотеки. Приложения предназначены для установки конечным пользователем на устройство, а библиотека — это набор компонентов и инструментов, которые могут переиспользоваться в других проектах.
- И приложения, и библиотеки имеют схожую структуру и содержат следующие папки и файлы…



android, ios, linux, macos, web, windows

- Папки android, ios, linux, macos, web и windows в проекте Flutter содержат код и настройки, специфичные для каждой платформы. В папках находятся файлы и каталоги, необходимые для сборки, упаковки и запуска приложения на соответствующей платформе.
- Каждая из этих папок имеет свой собственный набор инструментов и зависимостей, которые нужно установить и настроить, чтобы успешно собрать и запустить приложение на нужной платформе. При разработке Flutter-приложений важно учитывать эти различия и тестировать приложение на всех платформах, для которых оно предназначено.
- Если планируется, что проект будет поддерживать только определенные платформы, то их можно перечислить в параметре **--platforms**, перечислив через запятую список необходимых платформ. Например, если планируется, что проект будет поддерживать только **Android и iOS**, то команда будет выглядеть следующим образом: **flutter create --platforms=ios,android**.

lib

- Папка lib в проекте Flutter — это основная папка, в которой располагается весь исходный код приложения. В ней находятся все Dart-файлы проекта, определяющие логику и интерфейс приложения. В этой папке лежит файл main.dart.

main.dart

- Файл main.dart в проекте Flutter — точка входа в приложение, где находится главный метод main(). Этот файл содержит основной код приложения и выступает отправной точкой для его запуска.

- В файле main.dart не рекомендуется размещать много логики или множество UI-компонентов, так как это может привести к перегруженности и неудобству в разработке и поддержке кода. Часто в этом файле создаётся лишь функция main, а более сложная логика выносится в отдельные файлы и классы.

pubspec.yaml

- Файл pubspec.yaml относится к важным компонентам проекта Flutter или Dart. Он определяет конфигурацию проекта и используется, чтобы определить зависимости проекта, версии библиотек, настройки сборки и другие свойства проекта.

```
description: A new Flutter project.

publish_to: 'none'
version: 1.0.0

homepage: https://my_project_site.com
repository: https://github.com/my_project_url
issue_tracker: https://github.com/<my_project_url>/issues
documentation: https://my_project_site.com/documentation

environment:
  sdk: '>=2.18.0 <3.0.0'

flutter:
  uses-material-design: true
```

Есть обязательные параметры файла pubspec.yaml:

- name — имя проекта. Чаще всего оно совпадает с папкой, в которой лежит проект.
- environment — параметр, отвечающий за окружение проекта — версию Dart и Flutter. В нашем pubspec.yaml используется параметр sdk для указания версии Dart.

Есть также обязательные для библиотек параметры:

- version — актуальная версия библиотеки;
- description — описание проекта. Для каждого нового проекта оно будет как в примере выше — **A new Flutter project**.

И несколько необязательных параметров:

- homepage — ссылка на главную страницу проекта, чаще оставляют ссылку на репозиторий;
- repository — ссылка на репозиторий проекта с исходным кодом;
- issue_tracker — ссылка на трекер проблем и предложений от тех, кто пользуется проектом. Например, иногда указывают ссылку на секцию **issue на GitHub**;
- documentation — ссылка на документацию проекта;
- executables — скрипты проекта в виде исполняемых файлов, которые могут быть запущены из командной строки.

- platforms — платформы, которые поддерживает проект;
- publish_to — ссылка на сервер публикации пакетов, где находится библиотека. Если данный параметр пустой, то публикуемые пакеты отправляются на **pub.dev**. Если не хотите публиковать проект, можно указывать '**none**'. Также этот параметр можно использовать для указания пользовательского сервера публикации пакетов.
- funding — список ссылок, по которым можно перейти и проспонсировать автора разработки;
- false_secrets — секция для указания файлов, для которых мы не проверяем наличие секретов. При публикации пакета **pub** проверяет, нет ли в коде различных секретов: **API ключей** или криптографических данных. Если такие вдруг обнаружились, то пакет не публикуется. Но проверка может сработать ложно — принять за секрет файл, который им не является. Поэтому если проверка срабатывает на файл, но вы уверены, что он не содержит секретов, то его можно добавить в поле false_secrets, и проверка будет его игнорировать;
- screenshots — скриншоты, связанные с библиотекой: примеры виджетов или других визуальных элементов проекта:
- и зависимости: dependencies, dev_dependencies и dependency_overrides.

Файл **pubspec.yaml**, использующий все доступные поля, будет выглядеть следующим образом:

```yaml
description: A new Flutter project.

publish_to: 'none'
version: 1.0.0

homepage: https://my_project_site.com
repository: https://github.com/my_project_url
issue_tracker: https://github.com/<my_project_url>/issues
documentation: https://my_project_site.com/documentation

environment:
  sdk: '>=2.18.0 <3.0.0'

dependencies:
  flutter:
    sdk: flutter
  cupertino_icons: ^1.0.2

dev_dependencies:
  flutter_test:
    sdk: flutter
  flutter_lints: ^2.0.0
```

```yaml
executables:
    ## При активации с помощью команды pub global a
    my_command_1: main
    ## При активации с помощью команды pub global a
    my_command_2

flutter:
        ## Этот проект поддерживает все платформы
        platforms:
          android:
          ios:
          linux:
          macos:
          web:
          windows:
  uses-material-design: true
```

analysis_options.yaml

- analysis_options.yaml — файл позволяет настраивать конфигурацию линтера (англ. **linter**). Библиотеки для линтера предоставляют стандартный набор правил, который может быть применён в проекте, но пользователь может изменять и дополнять этот набор в соответствии с требованиями проекта.

- В примере ниже используется пакет **lints**. Это официальная библиотека для Dart-проектов, содержащая все рекомендуемые правила. Помимо этого, в примере добавлены правила, которые вносят ограничения дополнительно к тем, что есть у самой библиотеки.

```yaml
include: package:lints/recommended.yaml

analyzer:
  exclude: [build/**]
  language:
    strict-inference: true
    strict-raw-types: true

linter:
  rules:
    - cancel_subscriptions
```

- Поле include позволяет задавать правила из другого файла. В примере выше правила из файла recommended.yaml в пакете lints переиспользуются в analysis_options.yaml.
- Поле analyzer — для настройки статического анализа: включения более строгих проверок типов, исключения файлов, игнорирования определённых правил, изменения строгости правил или включения экспериментов. Так, флаг strict-inference: true будет ругаться на строку кода final myMap = {};.
- Поле linter используется для кастомизации линтера, для удаления правил и добавления новых.

test

- В этой папке находятся юнит - и виджет-тесты, написанные разработчиком.

.gitignore

- Это файл, который позволяет указать, какие файлы или папки проекта нужно игнорировать в системе контроля версий **git**.

```
folder_name/                # Игнорируем всю папку folder_name
folder_name/subfolder_name # Игнорируем всю папку subfolder_name, лежащую в папке folder_name
**/folder_name/             # Игнорируем все папки в проекте с названием folder_name
**/*.png                    # Игнорируем все файлы проекта с расширением png
folder_name/file_name.dart  # Игнорируем файл file_name.dart в папке folder_name
```

- Для Dart и Flutter-проектов, помимо всего остального, рекомендуется добавлять папки .dart_tool и build в файл .gitignore, а для библиотек добавлять в тот же файл дополнительно к папкам, перечисленным ранее, pubspec.lock файл.

README

- README — это файл, который содержит информацию о проекте, его назначении, функциональности и использовании. Это первый файл, который разработчик или пользователь читает, когда знакомится с проектом, он обычно расположен в корневой директории проекта.
- Файл README часто используется для предоставления следующей информации:
- описание проекта — это описание проекта, его назначения и области применения. Это обычно первый абзац файла README;
- установка и запуск — инструкции по установке и запуску проекта, которые должны помочь пользователям начать работу с проектом;
- использование — детальное описание функциональности проекта и того, как использовать различные компоненты или функции;
- примеры — примеры использования проекта, которые могут помочь пользователям лучше понять, как он работает и что можно сделать с его помощью;
- зависимости — список зависимостей и версий пакетов, которые необходимы для работы проекта;
- лицензия — информация о правах и ограничениях использования проекта, включая его лицензию.

- Хорошо написанный файл README помогает пользователям быстро вникнуть в проект и начать работу с ним. Он также может служить важным источником документации для проекта, который может быть использован как разработчиками, так и конечными пользователями.

- Важно помнить, что файл README открыт для всех, поэтому в нём не следует размещать конфиденциальную информацию.

LICENSE

- Для публичных репозиториев принято добавлять файл LICENSE, чтобы лицензировать код. Это позволяет другим пользователям добавлять, изменять и распространять код. Некоторые лицензии не накладывают никаких ограничений на разработчиков, которые хотят переиспользовать написанный код, а некоторые обязывают указывать имя репозитория или автора или даже платить за использование пакета.

- Существует большое количество лицензий, и иногда найти подходящую достаточно сложно. Эту проблему поможет решить проект **choosealicense**, разработанный **GitHub**. Он помогает разобраться в том, какие лицензии существуют, какие есть между ними различия и какая подходит именно для вашего проекта.

.dart_tool

- Папка .dart_tool создаётся автоматически при выполнении операций, связанных с инструментами и пакетами в проекте. Внутри этой папки содержатся временные файлы, кеши, артефакты и другие данные, которые генерируются при компиляции, анализе и выполнении кода.

Вот несколько распространённых файлов и папок, которые могут находиться внутри папки .dart_tool:

- build — папка, где сохраняются результаты сборки проекта. В ней могут быть подпапки, связанные с конфигурацией среды выполнения и инструментами сборки;
- .dart_tool/package_config.json — файл, содержащий информацию о зависимостях проекта, включая пути к пакетам, используемым в проекте;
- .dart_tool/package_config_subset — данный файл содержит подмножество информации о зависимостях, необходимых для быстрого запуска или сборки проекта;
- .dart_tool/pub — эта папка содержит временные файлы, связанные с менеджером пакетов **pub** для установки и обновления зависимостей проекта.

.metadata

- .metadata — приватный файл, который среда разработки использует, чтобы отслеживать параметры Flutter-проекта. В таком файле может находиться информация о версии фреймворка, который используется в проекте, вспомогательные файлы для миграции проекта на другую версию фреймворка и проч. Такой файл автоматически создаётся средой разработки и не должен редактироваться вручную.

dart doc
Создавать документацию для библиотек и проектов помогает dartdoc. Это команда, которая позволяет генерировать документацию по комментариям в коде.

```
///
/// Короткий комментарий для класса Model
///
class Model {}

/*
   * Это длинный комментарий для функции functionGetModel
     Он может указывать на [Model], будет автоматически оставлена ссылка в д
     Он также может указывать на [function], будет автоматически оставлена с

*/
Model functionGetModel() {
    ...
}
```

Полезные ссылки

Документация о pubspec.yaml

Файл gitignore

Официальная документация про расширение правил линтера

Список того, что не стоит коммитить для проекта

Официальная документация про правила линтера

Заключение

- В данной лекции мы рассмотрели важные папки и файлы проекта: папки платформ, конфигурационные файлы, файл настройки линтера и др. Эти части проекта будут встречаться нам ещё неоднократно в процессе разработки.

- В следующих лекциях мы поговорим о нюансах самого языка Dart, постепенно приближаясь к созданию профессиональных проектов на Flutter