Code coverage is a metric that can help you understand how much of your source is tested. It's a very useful metric that can help you assess the quality of your test suite, and we will see here how you can get started with your projects.

Code review is a software quality assurance activity in which one or more people check a program, mainly by viewing and reading parts of its source code, either after implementation or as an interruption of implementation

Tools Which can be used for easier dev experience

**Flutter Extension (Vs Code)**

This VS Code extension adds support for effectively editing, refactoring, running, and reloading Flutter mobile apps**.**

**Dart Code Metrics**

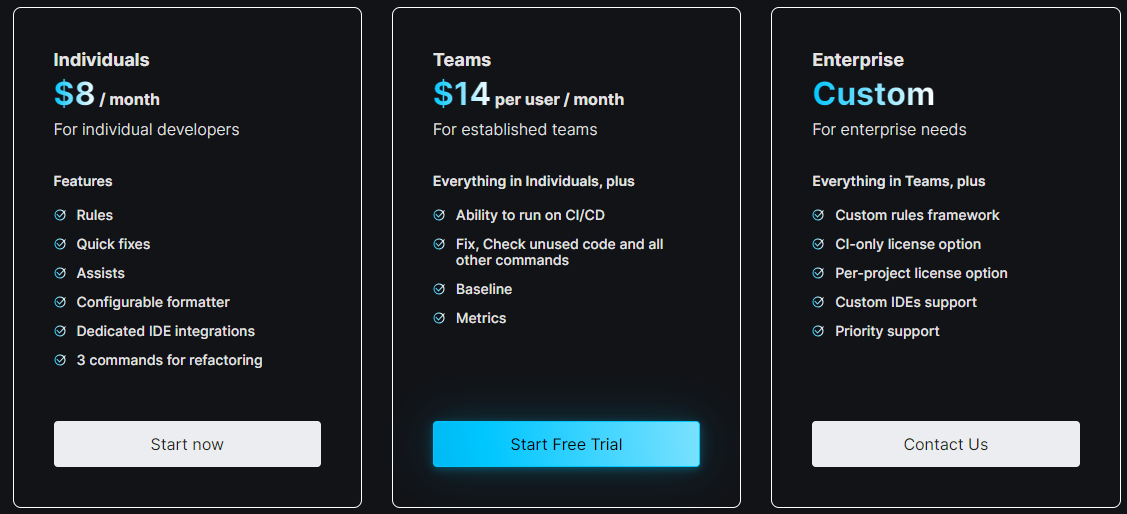
It used to be an open source project but now has changed to paid software.

DCM (formerly Dart Code Metrics) is a tool that helps improve the quality and consistency of Dart code by identifying and reporting problems, such as bugs and code that doesn't follow best practices.

It also collects analytical data on the code through calculating code metrics and can be configured to set thresholds for these metrics.

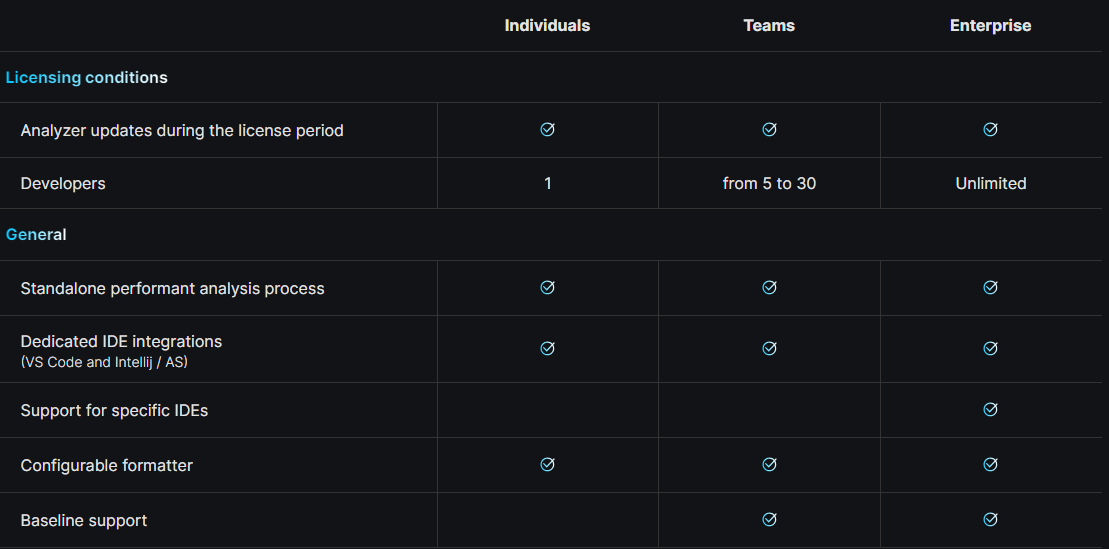
It can be launched via the command line, connected as a plugin to the Dart Analysis Server, or as a library. Launching via the command line allows you to easily integrate the tool into the CI/CD process, and you can get results in Console, HTML, JSON, CodeClimate, or GitHub. Connecting the tool as a plugin to the Analysis Server allows you to receive real-time feedback directly from the IDE.

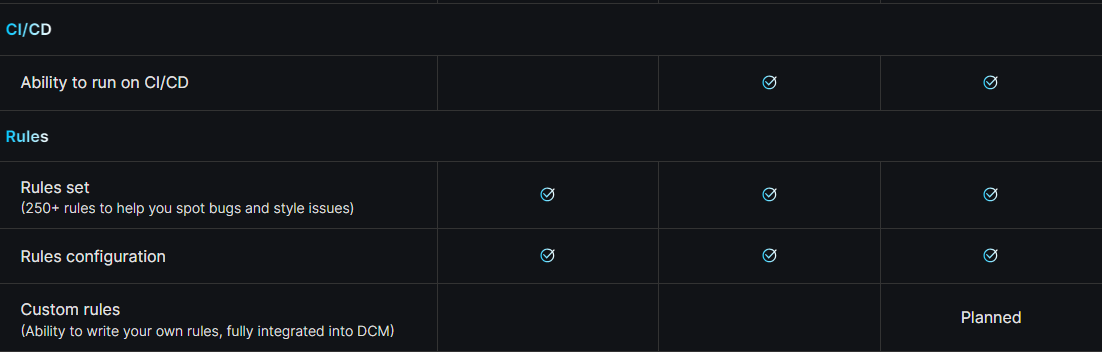
**Pricing**

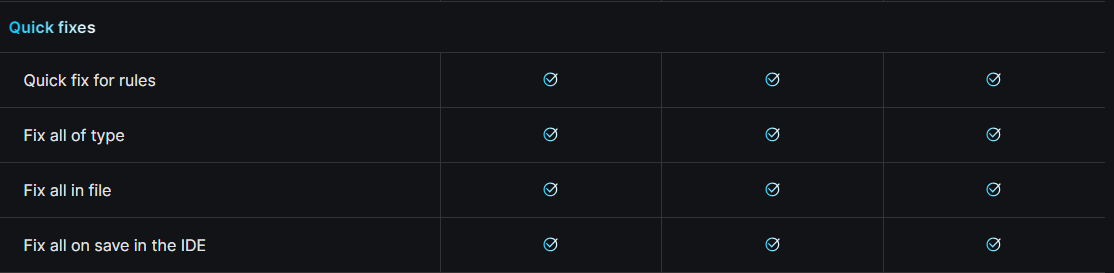
****

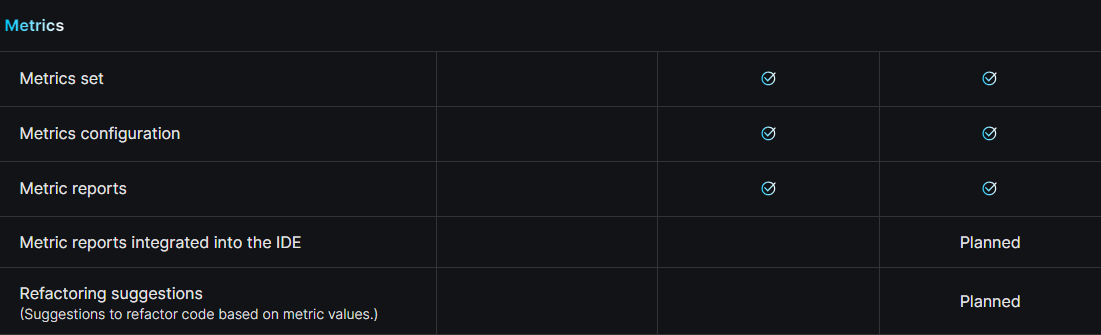
Companies using the tool

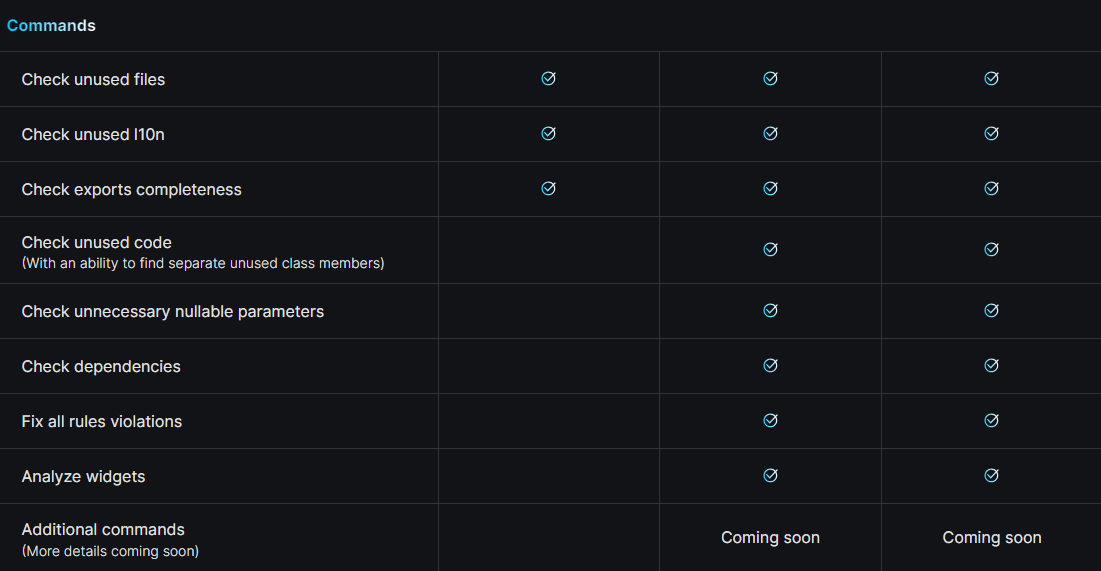
1. Google
2. Netglade
3. Mews
4. Chili
5. Breeze
6. Wrike
7. Tonal
8. Surf

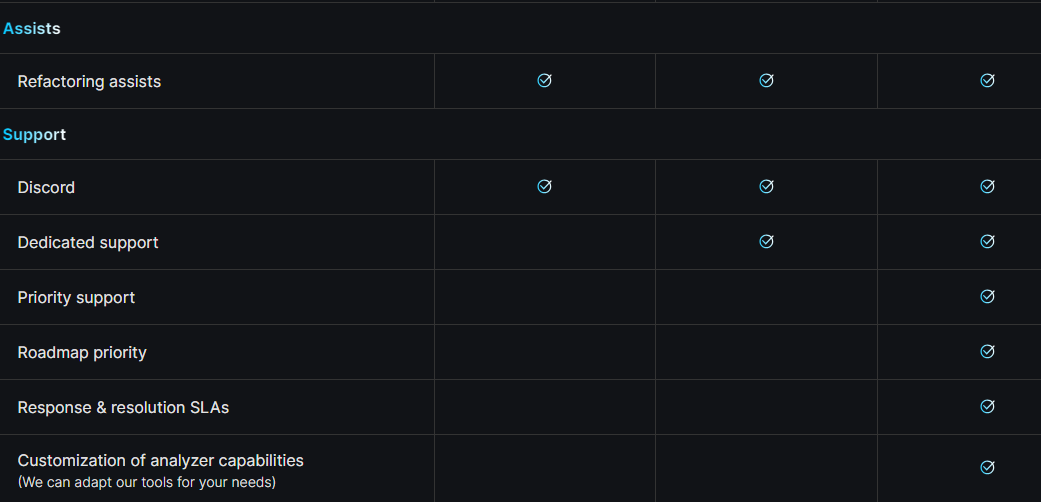












Strengths of DCM:

* Comprehensive metrics: Covers a wide range of code quality measures, giving you a holistic view of your codebase.
* Strong rule set: Provides many out-of-the-box rules for best practices and common issues, saving you time and effort.
* Highly configurable: You can customize rules and metrics to fit your specific needs and coding conventions.
* Seamless integration: Integrates well with popular IDEs and CI/CD pipelines for a smooth workflow.

Paid tool: DCM is a paid product, unlike some free open-source alternatives.

* Learning curve: While powerful, understanding the full potential of DCM and customizing rules can require some learning.
* Focus on metrics: DCM primarily focuses on quantitative metrics, which might not tell the whole story about code quality.

**Installation**

***C:/> choco install dcm***

**Top Review**

**"On the Dart and Flutter DevTools team, we use DCM rules to continuously validate our code. A rule that was very valuable for our codebase was detection of dead code, as it keeps the codebase smaller and thus reduces maintenance cost. Other rules have improved our dev velocity by increasing code readability and making debugging easier."**

**Jacob Richman, Staff Tech Lead Manager, Google**

**Mega Linter**

Mega-Linter integrates with over 59 linters, covering 24 languages and 20 tooling formats. This extensive range ensures thorough checks for various code aspects, including style, best practices, and potential errors.

This tool is 100% open-source and free for all uses (personal, professional, public and private repositories)

At each pull request it will automatically analyze all updated code in all languages

MegaLinter can run on any CI tool and be run locally: no need to authorize an external application, and your code base never leaves your tooling ecosystem

Supports: **c#, dart, JavaScript, Go, Kotlin, PHP, Swift, .NET** and many more

**Features:**

* Linting: Checks your code against a set of configurable rules to identify stylistic inconsistencies, potential errors, and unused code.
* Auto-Fixes: For some issues, MegaLint can automatically suggest and apply fixes, saving you time and effort.
* Customizable Rules: You can fine-tune the ruleset to match your coding style and project requirements.
* IDE Integration: Works seamlessly with popular IDEs like VSCode and IntelliJ IDEA, displaying linting errors and suggestions directly in your code editor.
* Command-Line Interface: Run MegaLint from the command line for quick analysis or integration with build systems.
* Community-Driven: MegaLint has a thriving community that contributes new rules and features regularly.

**Advantages:**

* Improved code quality: Consistent code style, fewer errors, and better adherence to best practices lead to more maintainable and reliable code.
* Increased developer productivity: Linting helps catch errors early, reducing the time spent debugging later. Auto-fixes and clear error messages further streamline the development process.
* Enhanced teamwork: Consistent coding style makes code reviews easier and ensures everyone is following the same standards.
* Free and Open-Source: No hidden costs or limitations, accessible to individuals and teams of all sizes.

**Disadvantages:**

* Learning curve: Setting up custom rules and configuring MegaLint for your specific needs can require some initial effort.
* Over-precision: Some linting rules may be overly strict or flag stylistic differences as errors, causing false positives.

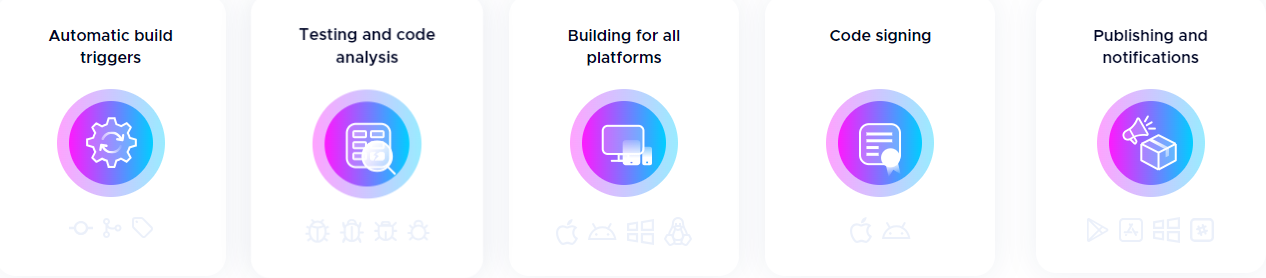
**Installation**

CLI installation - ***npx mega-linter-runner –install***

**Codemagic**

Codemagic is a cloud-based Continuous Integration/Continuous Delivery (CI/CD) product specifically designed for mobile developers building apps with Flutter, React Native, native iOS, native Android, Unity, Kotlin Multiplatform Mobile, and Ionic.

Codemagic automates the process of app building, testing, and deployment to app stores such as the Apple App Store, Google Play, Microsoft Store, and Huawei App Gallery. New builds can be triggered when code is pushed to your repository, tags are added, or pull requests are merged. You can configure multiple workflows to make the app available to your dev team, distribute the latest build to your test team, or publish your app to production.

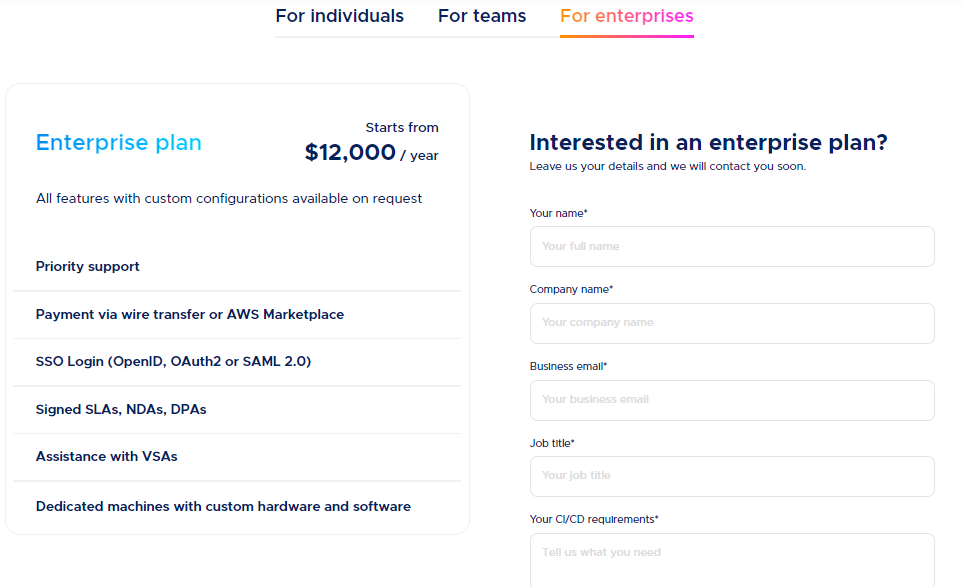


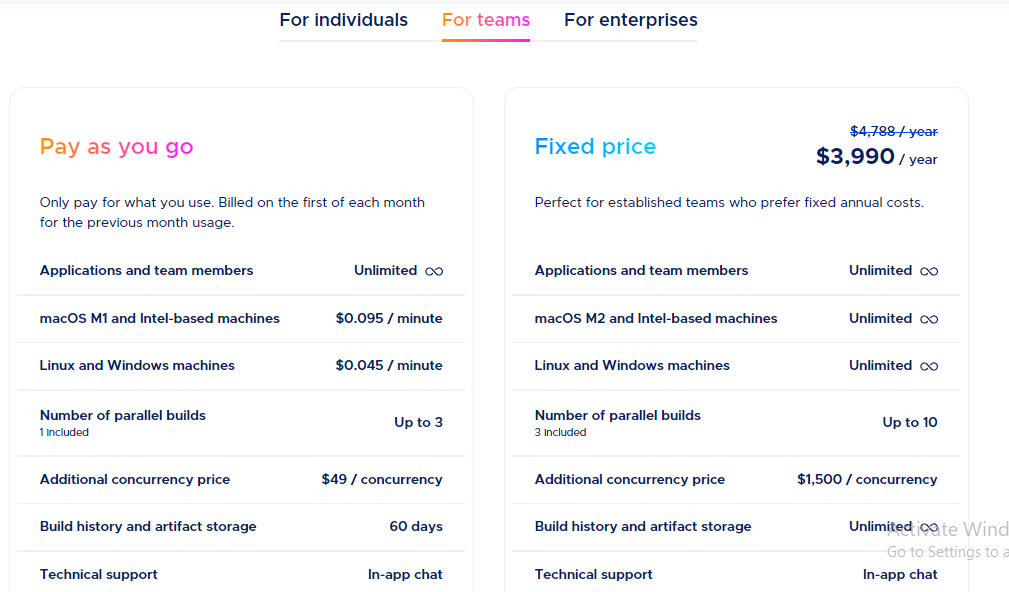
Support for: Flutter, React Native, Native IOS, Native Android, Ionic, Unity

Companies Using Codemagic

1. Toyota
2. Ascis
3. ShareChat
4. HootSuite
5. Sportsbet
6. Lotum
7. Flutter Flow
8. Koa Health
9. Taskrabbit
10. Remote Coach

**Pricing:**





**Advantages:**

1. Focus on Mobile Apps: Codemagic caters specifically to the needs of mobile devs, streamlining workflows for Flutter, React Native, and native (Android/iOS) projects.
2. Easy to Use: The interface is user-friendly and intuitive, even for developers new to CI/CD.
3. Automated Builds and Tests: Automate your build and testing processes, saving time and reducing human error.
4. Parallel Builds: Codemagic utilizes cloud infrastructure to run parallel builds, significantly speeding up your development cycle.
5. Continuous Integration and Delivery: Set up automated builds and deployments to your app stores, enabling rapid delivery of new features and bug fixes.
6. Comprehensive Testing: Offers a variety of testing options, including unit tests, integration tests, UI tests, and performance tests.

**Disadvantages:**

1. Price: While offering a free tier for small projects, paid plans can be expensive for larger teams or projects with high build volumes.
2. Limited Customization: While configurable, the platform may not offer the same level of customization as some more advanced CI/CD tools.
3. Steeper Learning Curve for Advanced Features: While the basic interface is easy to use, mastering advanced features and configuration options might require some technical expertise.

**How it works**

* Codemagic CI/CD integrates with Azure DevOps, GitHub, GitLab, Bitbucket and other self-hosted or cloud-based Git repositories.
* Build mobile apps using standard or premium instances of macOS, Linux and Windows build machines.
* Assure the quality of your apps with automated testing. Run unit tests or test your apps on simulators, emulators or real devices as part of your continuous integration and delivery pipeline.
* Set up build notifications for fast feedback on your builds. Debugging is easy thanks to clear logs and remote access to the build machine. Codemagic provides integration to workspaces like Slack.
* Codemagic will automatically publish your app with every build or on terms that work for you so you can release your mobile apps in record time and get to the market faster.

**Installation**

Adding applications to Codemagic is a simple and straightforward process of connecting your Git repository and selecting the repository root for the application.

Comparison Between DCM, MegaLint and Codemagic

|  |  |  |  |
| --- | --- | --- | --- |
| Feature | Dart Code Metrics | MegaLint | Codemagic |
| Primary Focus | Code quality and maintainability | Code linting and static analysis | CI/CD for mobile apps |
| Code metrics | Yes | No | Yes (coverage metrics) |
| Code linting | No | Yes | No |
| CI/CD | No | No | Yes |
| Integrations | IDEs, build systems | IDEs, build systems | Git hosting, app stores |

**APP FLAVORING**

**Flutter Flavorizr**

Flavors are a way of creating different environments for our application. It allows us to specify the app configuration for each of our cases. For example, we want to create a development flavor that is going to use different API endpoints than the production flavor which will be available to customers.

It’s a free tool which can be added using pubspec.yaml file

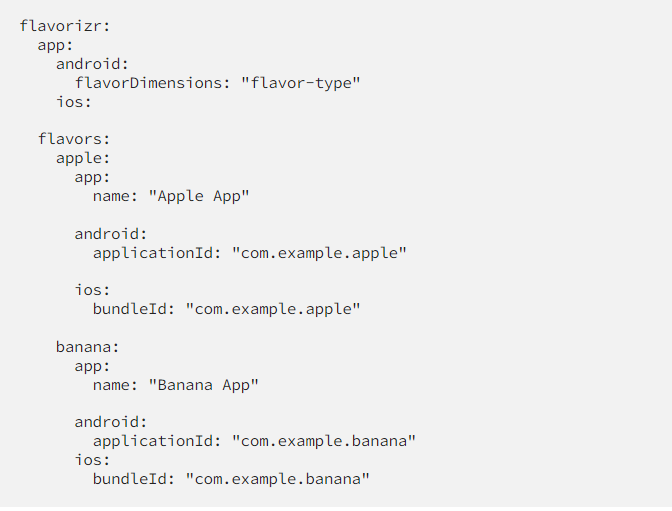
Before running Flutter Flavorizr, you must install the following software:

* Ruby
* Gem
* Xcodeproj (through RubyGems)

These prerequisites are needed to manipulate the iOS and macOS projects and schemes.

With these well-configured flavors, the CI/CD is also simplified.

Configuration Example

****

How to set Custom Flavors:

import 'package:flutter/material.dart';

enum Flavor { apple, banana, barcelona, orange }

class F {

  static Flavor? appFlavor;

  static String get name => appFlavor?.name ?? '';

  static String get title {

    switch (appFlavor) {

      case Flavor.apple:

        return 'Apple App';

      case Flavor.banana:

        return 'Banana App';

      case Flavor.barcelona:

        return 'barcelona App';

      case Flavor.orange:

        return 'orange App';

      default:

        return 'title';

    }

  }

  static Color get color {

    switch (appFlavor) {

      case Flavor.apple:

        return Colors.red;

      case Flavor.banana:

        return Colors.yellow;

      case Flavor.barcelona:

        return Colors.blue;

      default:

        return Colors.white;

    }

  }

}

To execute a specific build: ***flutter run --flavor [flavor\_name] -t lib/[file\_name]***

To run a specific process: flutter pub run flutter\_flavorizr -p <processor>

Tutorial / Doc:

<https://pierre-dev.hashnode.dev/get-the-best-out-of-flutter-flavors-with-flutterflavorizr>

Official Doc:

<https://github.com/AngeloAvv/flutter_flavorizr>

Last version released on: Jun 9 2023

Last Push in the repo: Last Month

Drawback: Needs to be implemented in a fresh project for best use, otherwise it will be more complicated