POLITECNICO DI TORINO

Master's degree
in Computer Engineering

Master's Degree Thesis

IMPLEMENTATION OF A FLUTTER MOBILE APPLICATION TO SUPPORT A HEALTHIER LIFESTYLE

Supervisor
prof. Maurizio Morisio

Candidate
Domenico Gagliardo

Academic Year 2024-2025

Contents

Introduction

The work that is being presented in the next pages is based on two main arguments: the importance of following a healthy lifestyle and how technology can help in achieving this goal. Infact, our project work along with my colleagues concerned the further implementation of a partially developed application suited for achieving a good lifestyle. The application, composed both by a frontend part and a backend one, allow the users to track their data regarding nutrition, as well as physical activity, sleep and emotional state. These data can be either inserted manually or collected through a wearable device. The application still requires the user to insert his basics information, sich as username, age, sex, weight and height. They are then all available in the application, where the user can eventually change them. The application also integrates a quiz gamification approach through a specific section of the app, that allow the users to be more engaged, but also learn and deepen their knowledge about the topic. Finally, the application also involves the usage of recommendation as a way to provide suggestion about health and lifestyle to the user and improve his healthy journey. The thesis is then structured in 5 chapters: the first one deepen the health and well being topic, considering the guidelines that literature has found over the years of studying the topic. It then focuses on how technology can help us in achieving a better lifestyle, by considering which are the main software and hardware tools that can be used, such as smartphone application and wearable devices that allow to collect data and share them with application to create a more complete picture of the user's health. The second chapter covers the system, by firstly consider the updated requirements related to it (both functional and non-functional) and then moving to the system architecture, also redesigned consequently. The third chapter moves into more tecnical aspects, by considering the technology stack that has been used to develop the application, starting from the used programming languages and automation tools, moving then to the IDE and to the backend side. The fourth chapter then talks about the overall implementation of the application, by focusing on my particular contribution. Finally, the fifth chapter talks about the performance analisys of the application, by considering the different testing devices with different hardware specs that have been used to test the application performances, and then concludes the work by considering the results that have been achieved and the future work that can be done to improve the application.

1 Health and Well Being

Health is one of the most important, if not the most important aspect of a person's life. For this reason, over the years, different organisations have established guidelines on how to stay healthy, thus increasing people's life expectancy and quality of life. Among these, the most widely worldwide recognized is the World Health Organization (WHO) [1] that provides several guidelines not only in term of physical activity, covering also other health aspects.

As far as concerns the physical activity, the WHO estimates that 1 in 3 adults and 4 in 5 adolescents do not do enough physical activity, with adolescents girls less active than adolescents boys and with inactivity levels that increases after 60 years of age. This level is expected to rise due to country economic development (more use of technology, change of cultural values and more sedentary behaviour). This trend sadly keep going in the wrong direction, despite the fact that physical activity has countless benefits, like reducing the risk of heart disease, cancer, diabetes, hypertension and depression.

Food is also crucial in order to be healtier. Having a healthy diet helps to prevent several diseases (like heart disease, diabetes and cancer) and also malnutrition in all its forms. However, care has to be taken in choosing the right food sources that have good quality and avoid processed foods. Eating noble foods like fruits, vegetables, legumes, nuts, and whole grains, while limiting the intake of salt, sugar, and fats, is the key to a healthy diet. For all these reasons, both physical activity and diet are strongly promoted by the WHO through his global action plan, by calling international partners, private sector and also civil society to take action in order to support them.

1.1 Guidelines

1.1.1 Physical Activity Guidelines

As far as concerns the physical activity, the WHO gives some recommendation based on the age group [3]:

- 5-17 years:

- Children and Adolescents:
  - **Regular physical activity** enhances fitness, cardiometabolic health, bone strength, cognitive and mental health while reducing body fat.
  - **Sedentary behavior** leads to increased adiposity, poorer cardiometabolic health, behavioral issues, and reduced sleep duration.

- Adults and Older Adults:
  - **Active adults** experience lower body fat, risks of all-cause mortality, cardiovascular diseases, hypertension, specific cancers, and type-2 diabetes. They also enjoy improved mental health, cognitive function, and sleep quality.
  - **Sedentary lifestyles** are associated with higher mortality rates and increased incidences of chronic diseases like cardiovascular issues and cancer.

- Pregnant and Post-Partum Women:
  - **Engaging in physical activity** decreases the risks of pre-eclampsia, gestational hypertension, gestational diabetes, excessive weight gain, newborn complications, and postpartum depression, while having no negative effects on birth weight or stillbirth risk.

    Active vs Sedentary lifestyle[2].

  - Should do at least 60 minutes of physical activity with moderate/vigorous-intensity daily (of course more than 60 minutes provides additional benefits), as well as bone-strengthening and muscle-strengthening activities.

- 18-64 years:
  - Should do at least 150 minutes of physical activity with moderate-intensity in a week or at least 75 minutes of physical activity with vigorous-intensity in a week or an equivalent combination of both (increasing moderate-intensity will provide additional benefits), but also muscle-strengthening activities by involving major muscle groups.

- 65 years and above:

  - Should do at least 150 minutes of physical activity with moderate-intensity in a week or at least 75 minutes of physical activity with vigorous-intensity in a week or an equivalent combination of both (increasing moderate-intensity will provide additional benefits), recruiting major muscle groups with muscle-strengthening activities but also including exercises to enhance balance and prevent falls in case of poor mobility.

1.1.2 Healthy Diet Guidelines

Regarding having an healthy diet, also here the WHO gives some guidelines, emphasizing that a good diet includes legumes, fruit, vegetables, animal sources foods (like meat, fish, eggs, and milk), cereals (like wheat and barley) and also tubers (like potato and yam). It also gives some further recommendations[4]:

- Babies and young children breastfeeding:
  - Breastfeeding promotes healthy growth, as well as having long-term benefit, like reducing the risk of developing nonncommunicable diseases, overweight, obesity. From birth until 6 month of life is important to feed the baby only with breastmilk, while from 6 month to 2 years of age is important to introduce also additional complementary foods, while still breastfeeding.

- Eat lots of vegetables and fruit:
  - These foods are rich in vitamins, minerals, dietary fiber, antioxidants and plant protein, which help to prevent heart disease, stroke, diabetes, obesity and some cancers.

- Eat less fat:
  - Fats and oils are concentrated source of energy, so it is important to limit them, especially saturated and industrially-produced trans-fat that can increase the risk of stroke and heart disease. To avoid gaining weight in an unhealthy way because of them, care has to be taken in using unsaturated vegetable oils (like olive oil) instead of animal fats or oils high in saturated fats (like butter or palm)

and in any case fat consumption should not exceed 30% of total energy intake.

- Limit sugars:
  - Sugar consumption should be the 10% of total energy intake. This should be achieved by limiting soft drinks, soda and other drinks high in sugars (fruit juices or yogurt drinks) and also by avoiding the consumption of processed foods high in sugars (like cookies, cakes, chocolate). Better to choose fresh fruits instead of them.

1.2 Technology Role in Health

Having clear in mind the importance of an healthy lifestyle and a good dieting, it is also important the role that technology can have in this. Even though it is still possible to achieve a good lifestyle without technology, it has to be said that using technology sure makes it easier across several aspects. Several researches in this aspect have been performed by the National Institutes of Health (NIH) [5], an american health organization driven by the U.S. Department of Health and Human Services. The NIH found notable improvement in diet and activity habits with usage of mobile technology.
They took 204 adult people that met these constraints:

- Being obese or at least overweight.
- Having a diet high in saturated fat and low in fruits and vegetables.
- Perform a small amount if daily physical activity.
- Having lots of sedentary time.

then they divided these people into four groups, where each one had a specific diet. Furthermore, a mobile device was given to them and they had to enter their diet and activity data into the device for a 20-week follow-up period. Coaches would then receive the data during this period to monitor these people, as well as contacting them in order to provide encouragement and support towards an healthy change. The results found overall improvements in all four groups, emphasizing how technology can improve a fitness

journey, also as a means to provide support and motivation [6].

Another aspect in which technology surely can help is about measurements: during physical activity and dieting, several aspects require measurements: the amount of calories burned, the heart rate during physical activity, the amount of calories taken, as well as the types of food consumed, their macronutrients (carbohydrates, proteins and fats) and so on. In this aspect, technology can provide several tools to help, like smartphone applications or wearable devices.

In the first place, technology helps in easing the process of performing measurements and gather these data (both for physical activity and dieting) that can be boring to do repetitively for us humans. In the second place, technology can provide a more accurate measurement of these data, more difficult to achieve manually. Related to this aspect, a study of the NIH showed how physical activity measurements taken by devices proved to be more punctual compared to the one taken manually with a diary [7].

1.2.1 Smartphone Applications

Moving to the technological tools that can be employed, smartphones surely are one of the most used devices and they allow to exploit several aspects related both to dieting and physical activity. Also here a study of the NIH [8] showed that users were more stimulated into following a healthy diet, particularly liking applications that were quick and easy to administer, and those that increase awareness of food intake and weight management. Even though work has to be made to increase food awareness, the study recognizes the importance of smartphone applications in this aspect. Dually another study has been done also on physical activity [9], showing that smartphone apps can be efficacious in promoting physical activity. Also in this case users tend to prefer applications that are user-friendly, thanks to their capability of automatically track physical activity (e.g., steps taken), track progress toward physical activity goals, as well as be flexible enough to be used with different types of physical activity. Countless of these smartphone applications are available to support an healthy lifestyle. They are cross-platform, so they can be used on both Android and iOS devices, in order to reach the largest audience possible. Here are some of the most popular applications, where for each the main features and the feature that distinguishes the application from other on the market are listed:

App Name	**Features (Distinguishing Features In Bold)**
MyFitnessPal	Food logging with a large database, barcode scanning, calorie and macro tracking, personalized insights, exercise logging, and integration with other apps. **Share and View Your Diary with Others.**
Fitbit	Step tracking, heart rate monitoring, sleep analysis, GPS tracking, food logging, and activity reminders. **Comprehensive health metrics tracking, including stress levels and Active Zone Minutes.**

Google Fit	Step counting, activity tracking, heart points, integration with health apps, and customizable fitness goals. **Collaborates with the American Heart Association and WHO for heart health insights.**
Nike Training Club	Guided workout programs, personalized fitness plans, workout tracking. **Free access to a variety of workouts, from yoga to high-intensity interval training.**
Strava	GPS tracking for running, cycling, performance metrics, social sharing, and route planning. **Community-focused with support for sharing routes and competing with others.**
Noom	Weight loss coaching, food logging, biometric tracking, and habit-building tools. **Focus on the psychological aspects of diet and health for long-term results.**
JEFIT	Exercise logging, workout planning, personalized workout, and performance tracking. **Training optimization with advanced analytics.**
Cronometer	Nutrition tracking, biometrics tracking, diary and diary groups, and micronutrient breakdown. **Advanced nutrient tracking suitable for specific diets.**
Lifesum	Food tracking, calorie counting, diet plans, water tracking, and nutrient breakdown. **Visual and user-friendly meal planning tailored to dietary preferences.**
Yazio	Calorie counting, meal planning, recipes, nutrition tracking, and progress reports. **Extensive recipe database and meal planning features.**

Table 1: Overview of popular diet and fitness apps with distinguishing features highlighted

1.2.2 Wearable Devices

Considering the Wearable Devices, even though they are less used compared to smartphones, their usage is growing more and more. Furthermore they are a good tool in order to track physical activity and dieting. Their main advantage is that they can be worn on the body, like a watch or a bracelet, but they are equipped with sensors allowing to collect data about the body, like the heart rate, the number of steps taken, the calories burned, the quality of sleep, and so on. They can also be connected to a smartphone in order to share these data with it, so that the user can have a more detailed view of his health status. Given their diffusion, applications have been introducing a way to connect to these devices, in order to exploit their data. This has been done by carefully considering their diffusion [10].

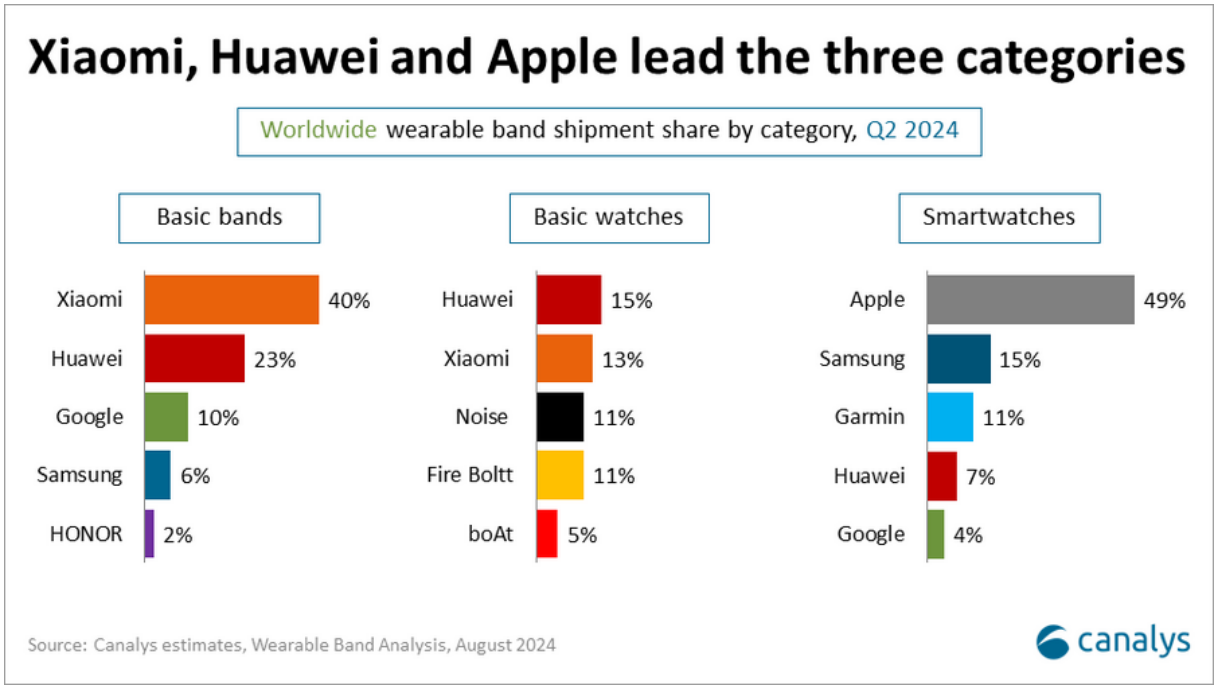

Figure 1: Wearable diffusion by major brands [10].

2 System Specifications

This section of the elaborate will involve the discussion of the system requirements, as well as how the system have been designed. Finally, the technological stack and the major tools adopted for the project will be covered, such as the programming languages, IDE, libraries, automation dependency tools.

2.1 System Requirements

Given the fact that this project was already existing, it had some implemented features (and consequently some basic requirements related to them), our work was based on extending the app capabilities by implementing new features or refactor existing ones.
Considering this aspect, I will classify the functional requirements that regards our work as follows:

FR1	**User Management**	
FR 1.1	Home Page	Allow a user to enter and also visualize data regarding his emotional state.
FR 1.2	Home Page	Allow a user to visualize data on the charts that exceeds the related goals differently (they visually differ from the other ones).
FR 1.3	Health Measures Page	Allow a user to view his vital metrics by using charts organized in different tabs instead of raw values, for a better understanding.
FR 1.4	Personal Information Page	Allow a user to provide his personal measures as well as his activity goals, used as upper bound into the charts of the Home Page.
FR 1.5	Notification System	Allow a user to receive notification with different frequencies to prompt him into inserting data regarding food, emotional aspect, balance capability, strenght capability.
FR 1.6	Assessment	Allow a user to visualize a periodic assessment produced thanks to the data that were previously inserted.
FR 1.7	User Registration	Allow a user to enter his demographics (age,sex,...) and body (height,weight,...) data when the registration is in progress.

Table 2: Overview of Functional Requirements related to the User Management.

FR2	**Data Management**	
FR 2.1	Data Source Migration	Migrating the main app data source from FitBit Server to Health Connect and Apple Health (Health data management and integration platforms, developed by Google and Apple respectively), while still keeping Google Firebase as additional data source.
FR 2.2	User Goal Data Integration	Allow a user to retrieve, visualize, insert and edit his goals.
FR 2.3	User Emotional Data Integration	Allow a user to retrieve, visualize and insert his emotional data, based on the selected time period.
FR 2.4	Health Data Backup	Added the necessary logic to perform a backup of the users's health data.
FR 2.5	Food Data Integration	Allow a user to insert his food data once received the notification.
FR 2.6	Emotional Data Integration	Allow a user to insert his emotional data once received the notification.
FR 2.7	Balance Capability Data Integration	Allow a user to insert his balance capability data once received the notification.
FR 2.8	Strenght Capability Data Integration	Allow a user to insert his strength capability data once received the notification.

Table 3: Overview of Functional Requirements related to the Data Management.

As far as concerns the non-functional requirements, we can classify them as follows:

NFR	**Type**	**Description**
NFR1	Reliability	The system should ensure at least 80% accuracy and functionality over the course of a year.
NFR2	Portability	The application must be capable of running on Android and IOS devices.
NFR3	Security	Robust login mechanisms should be implemented to protect user data and limit access only to authorized individuals.
NFR4	Usability	The application should be intuitive enough for users of all ages and skill levels, requiring minimal training.
NFR5	Data Privacy	User data must adhere to OAUTH for secure data handling.
NFR6	Performance	The app should smoothly load and handle user interactions within two seconds under typical conditions in order to achieve an optimal user experience.
NFR7	Interoperability	The application should seamlessly connect with third-party platforms like Health Connect and Apple Health, maintaining data accuracy and improving functionality.
NFR8	Localization	The app should support at least English and Italian languages, adapting content and formats (e.g., date, currency) accordingly.
NFR9	Modularity	The app's architecture should support modular development to facilitate future updates without impacting the entire codebase.

Table 4: Overview of Non-Functional Requirements related to the system.

2.2 System Architecture And Design

Focusing on the system architecture and his design, it can be summarized as follows:

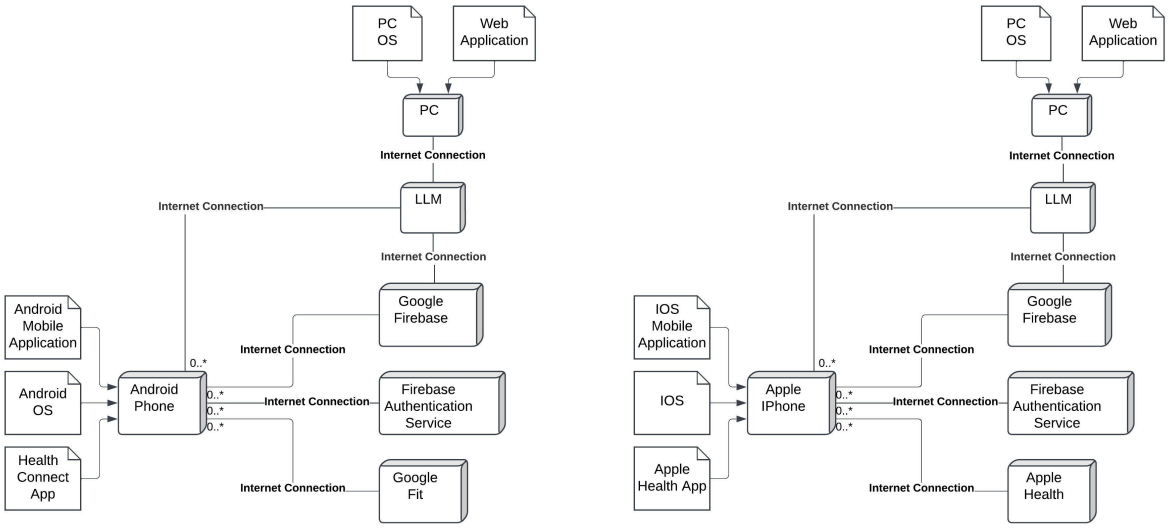

Figure 2: Overview of the System Architecture.

We can clearly distinguish the two main use cases (Android and IOS) with their distinguishing components and the common ones:

- Android Architecture:
  - **Android Phone** that represent the physical smartphone, along with his operating system, our application installed and health connect installed, used to manage the health data on Android, that further interacts with google fit server whenever is needed.
  - **Google Fit** server, used among the health connect app data sources.
- IOS Architecture:
  - **Apple IPhone** that represent the physical smartphone, along with his operating system, our application installed and apple health installed, used to manage the health data on IOS, that interacts with apple health server whenever is needed.
  - **Apple Health** server, used as data source for the apple health application.

- Common Components:
  - **Firebase Authentication Service** server, employed in order to perform and enforce authentication.
  - **Google Firebase** employed as a data source for the application logic, containing other needed data, such as profile information, in addition to the backup of the health data for each user (see table 3 FR 2.1) stored in Firebase Storage, Google Firebase service focused on storing user-generated content such as photos, videos, and other types of files.
  - A **Large Language Model** pre-trained and then fine tuned on the health data of the users employed to generate recommendation for the user.
  - A **Personal Computer** along with his operating system and a web application that allows to interact with the llm on administrator side to adjust recommendation as well as editing sentences that prompt the user into inserting data.

In addition to this architecture, the system can be further enriched by integrating a wearable device, that can be a very useful tool to gather data, as previously explained in section 1.2.2. In this scenario the wearable device will be connected to the smartphone:

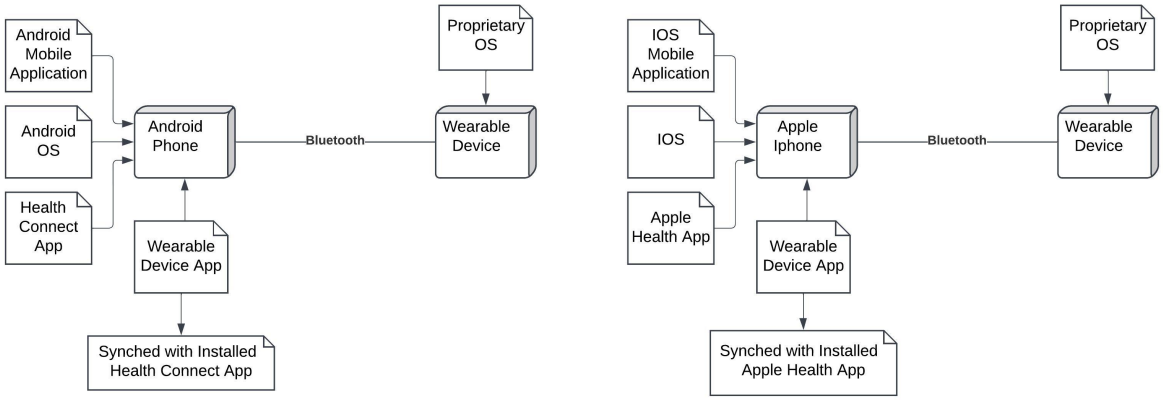

Figure 3: Overview of the Mobile System Architecture with Wearables.

In this case the architecture has been extended with the wearable devices, along with his operating system, that throuugh bluetooth connection can communicate with the smartphone (Android or IOS). On the smartphone side, the wearable application will be able to retrieve the data from the wearable device to then sync with the respective data management service (respectively Health Connect or Apple Health).

3 Technology Stack

This chapter covers the software technologies used for developing the mobile app.

3.1 Framework

3.1.1 Flutter

Figure 4: Logo of the Flutter Framework.

In order to develop the application, the flutter framework was employed. Flutter is an open-source framework created by Google. It allows to build applications that are natively compiled for mobile, web, embedded and desktop from a single codebase. The flutter code compiles into ARM, Intel machine code or JavaScript, depending on the machine for better performance on any device. It gives also the possibility to fine control the layout to be able to create a customized, adaptive designs that look and feel great on any screen. It is also very productive from the developer point of view, thanks to the hot reload feature, that allows to see the changes in real-time without losing any state. The developer experience is also enhanced by automated testing and developer tools that further allow to build production-quality apps. Among these the most relevant are widget and layout inspectors, network and memory profilers, extensive docs, the pub package manager (see section 3.3.1) as well as lots of pre-built widgets and layouts in the SDK. The framework is widely known for its stability and reliability: infact, it is used by Google who made it but also trusted by other well-known brands around the world, and maintained by a community of global developers, giving the possibility to collaborate on the open source framework, contribute to the package ecosystem on pub.dev, and find help wherever you need it. The framework also offer a seamless integration with google services, allowing to streamline development and reach a wider audience. Among these services Firebase, Google Ads, Google Play, Google Pay, Google Wallet, Google Maps stand out. The framework is based upon Dart (see section 3.2.1) as programming language [11]. To practically use Flutter, just install the Flutter SDK, which includes the full Dart SDK, and then configure any text-editor or integrated development environment (IDE) combined with Flutter's command-line tools. However, most popular options that include also a guided setup are Android Studio, IntelliJ IDEA, and Visual Studio Code [12].

3.2 Programming Languages

3.2.1 Dart

Figure 5: Logo of the Dart Programming Language.

The programming language on which Flutter is based upon and that makes possible most of his features is dart. Dart is a client-optimized language for making fast apps on any platform and try to offer the most productive programming language for multi-platform development, along with a flexible execution runtime platform. Dart is the foundation of FLutter, but also helps in several core delevoper tasks like formatting, analyzing and testing code. Among his features, the most interesting surely are the instant hot reload, that thanks to the Dart VM reflect immeditely any code change, and the possibility to build application for different devices but from a single codebase.

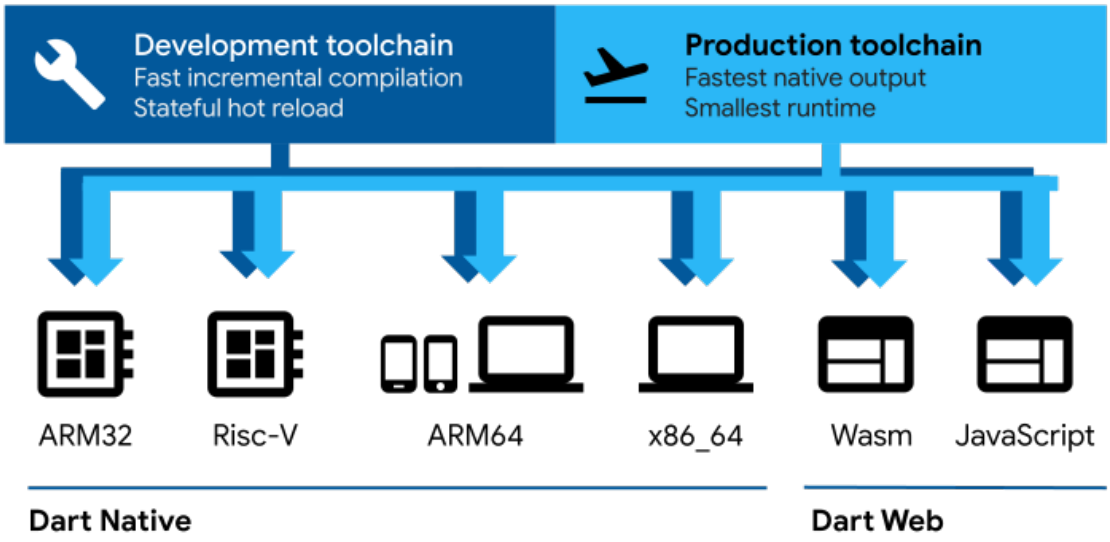

Figure 6: Overview of all the devices that dart is able to reach.

As we can see from fig. 6, the devices covered by a single code base range from embedded device architectures to mobile and laptop devices (by using Dart VM with just-in-time (JIT) compilation and an ahead-of-time (AOT) compiler) to web applications, with his web compiler that translate Dart into JavaScript or WebAssembly. Additionally, the language is type safe, so it uses static type checking to ensure that a variable's value

always matches the variable's static type. This is done without sacrificing flexibility, since the language still permits the use of a dynamic type combined with runtime checks, which can be useful during experimentation or for code that needs to be especially dynamic, through the usage of the `dynamic` keyword. The language has also built-in null safety, so values can't be null unless the programmer explicitly say they can be. In this way dart can protect from null exceptions at runtime through static code analysis. Unlike other null-safe languages, this non-nullability is also retained at runtime, so if dart determines that a variable is non-nullable, that variable can never be null. The language also comes with a mature and complete `async-await` syntax for UI with event-driven code, all paired with a concurrency system based on dart isolates (separates memory-isolated threads of execution used to achieve concurrency). Dart comes with a rich set of libraries, ranging from core libraries (`dart:core`) from the ones to parse json (`dart:convert`), for concurrency (`dart:isolate`), web (`package:web`) and so on [13].

3.2.2 Groovy

Figure 7: Logo of the Groovy Programming Language.

Groovy is an object-oriented programming language, with dynamic typing for the Java Platform, alternative to the Java language. It can also be employed as a scripting language, intended to manage application automations for the Java Platform. The language also has features similar to languages such as Python, Ruby, Perl, and Smalltalk. Based on the Java platform, the language uses a Java-like syntax, based on curly brackets and is dynamically compiled in bytecode via the Java Virtual Machine. The Groovy compiler can be used to generate standard Java bytecode to interoperate seamlessly on any Java project. Among his features, the interoperability with java (java files are valid groovy files) makes him very versatile, but groovy code can be more compact. It is characterized by object-oriented features (like operator overloading and polymorphic iteration) as well as safe navigation operator `?.` to check automatically for null pointers. The latest versions add the support to newer features like modularity, static compilation, type checking and mullticatch blocks. Groovy has also native support of markup languages such as XML and HTML by using an inline Document Object Model (DOM) syntax. A Groovy script is fully parsed, compiled, and generated all before execution (similarly to Pyrthon and Ruby) and differently from Java, a groovy

source code file can be executed as an uncompiled script under some circumstances[14]. Groovy is also used in the Gradle build system (see section 3.3.2), which is the official build system for Android applications, and it is used to define the build configuration and dependencies for the android part of the project.

3.2.3 Ruby

Figure 8: Logo of the Ruby Programming Language.

Ruby is an high-level, interpreted and general-purpose programming language, designed by having in mind the programmer productivity, simplicity and fun, but also minimizing programmer work and possible confusion. In Ruby, everything is an object, also primitive data types. The language is dynamically typed, uses garbage collection and just-in-time compilation and it is a multi-paradigm programming language by allowing procedural, object-oriented, and functional programming. Since in ruby everything is an object, everything has certain built-in abilities called methods. Every function is a method, methods are always called on an object and methods defined at the top level scope become methods of the object class. Ruby also supports inheritance with dynamic dispatch, mixins and singleton methods, but does not support multiple inheritance. However there is still the possibility for classes to import modules as mixins. Among its other most relevant features we have dynamic and duck typing which allows variables to hold objects of any type and change types at runtime, garbage collection that automatically manages memory, reducing the need for manual memory management and helping prevent memory leaks. Also robust exception handling mechanisms are provided using `begin,rescue,ensure` and `raise`, allowing developers to gracefully handle runtime errors, and operator overloading, giving the possiblity to redefine the behavior of operators for their custom classes, providing more intuitive and readable code. Finally, concurrent programming support is enabled through native thread and fibers and a package manager (called RubyGems) is used to provide a standardized way to distribute, install, and manage Ruby libraries and applications[15]. Ruby is also used in the CocoaPods build system (see section 3.3.3), which is the official build system for iOS applications, and it is used to define the build configuration and dependencies for the iOS part of the project.

3.2.4 Yaml for flutter pub package manager

Figure 9: Logo of the Yaml Language.

YAML is a data serialization language that is human readable. His common usage regards configuration files and applications where data is being stored or transmitted. The language targets many of the same communications applications as Extensible Markup Language (XML) but his minimal syntax differs from Standard Generalized Markup Language (SGML). The language also supports JSON style inside the same file. In Yaml custom data types are allowed but tipically they are not needed, since the language natively encodes scalars (strings, integers, and floats), lists, and associative arrays (so maps, dictionaries or hashes) all based on the Perl language. In particular, there is the possibility for both lists and associative arrays to form nested structures. It reuses escape sequences like in C and uses whitespace wrapping for multi-line strings like in HTML. Read and write support of Yaml is available for many programming languages and editors, where their extension is tipically `.yaml` or `.yml`. Regarding the syntax, whitespace indentation is used for denoting structure like in Python and typically uses UTF-32 encoding in order to have JSON compatibility. It is possible to comment a line with the `#` character, use strings with single or double quotes, and specify lists and arrociative arrays respectively through square brackets (`[...]`) or curly braces (`{...}`)[16]. In the project, Yaml is used to define the dependencies in the `pubspec.yaml` file, which is the configuration file for the Flutter pub package manager (see section 3.3.1).

3.3 Automation Dependencies Tools

While dependency management on the project was mainly done by using the pub package manager (see section 3.3.1), native Android/iOS plugins require Gradle (Android) and CocoaPods (iOS) behind the scenes as additional tools. In this section these tools and their functionalities will be covered.

3.3.1 Pub Package Manager

For managing dependencies in our project at the project level, the pub package manager has been employed. It uses a `.yaml` file (pubspec.yaml) to list the dependencies and has

a command-line interface that works both for flutter framework and dart programming language (in our case we exploited it with flutter). The syntax is relatively simple and it works by using the `pub` command followed by a subcommand.
There are several subcommands that are divided into three main categories, based on functionalities[17]:

- **Managing Package Dependencies:** in this category the most relevant are the `get` or `upgrade` commands, that respectively fetch or upgrade the dependencies that are used by a package. Also other commandss are available, like `downgrade` to downgrade the dependencies at their lowest version for testing, or dually `upgrade` that upgrades the dependencies to their highest version.
- **Running command-line apps:** in this category fall the `global` command, which allows to make a package globally available, so that is possible to run scripts from his bin directory (the directory must also be added to the PATH variables).
- **Deploying packages and apps:** the `publish` command here is used to share developed dart packages with the community. This is done by uploading the package to the pub.dev website, acting as a package repository where all developer can download the published packages.

3.3.2 Gradle

Figure 10: Logo of the Gradle Automation Tool.

For managing dependencies/plugins in our project on the android part of the project, gradle has been employed. Gradle is a build automation tool for software development which uses either Groovy or Java/Kotlin as language (in our case we leveraged on Groovy). It offers support for all phases of a build process, from compilation to verification, dependency resolving but also test execution, source code generation, packaging and publishing. Gradle is a multi-language tool, supporting languages like Java, Kotlin, Groovy, Scala, C, C++ and Javascript. Gradle operates with his own domain-specific language in contrast with the maven approach with XML. It has been designed to handle multi-project builds, that can be very large. Infact, it supports a series of build tasks that can run serially or in parallel, and build components can be also cached. Regarding his main features, gradle follows a convention (folder

structure of the project, standard tasks and their order as well as dependency repositories) over configuration approach, all the build phases can be described in short configuration files. All conventions can still be overriden if it is necessary. One crucial gradle component is the plugin, that allow to integrate a set of configurations and tasks into a project, and can be either downloaded from a central plugin repository or custom-developed[18].

3.3.3 CocoaPods

Figure 11: Logo of the CocoaPods Dependency Manager.

For managing dependencies/plugins in our project on the android part of the project, cocoapods has been employed. CocoaPods is an application level dependency manager for Objective-C, Swift and other languages that uses Objective-C as runtime (like RubyMotion), is written in Ruby and uses it to manage the dependencies. It provides a standard format for managing external libraries and is inspired by the RubyGems and Bundler combination. CocoaPods is executed through the command line and installs the application dependencies by specifying them rather than copying source files. The dependencies are specified into a text file (Podfile). CocoaPods will then recursively resolves dependencies between libraries and then fetch the needed source code. His dependency resolution system is powered by Molinillo, which is also used by other large projects such as Bundler and RubyGems[19].

3.4 Integrated Development Environment

Figure 12: Logo of the Android Studio IDE.

As Integrated Development Environment for the project, Android Studio has been employed. Android Studio is the official IDE for Android app development, but it seamlessly supports flutter through the flutter plugin. Based upon JetBrains' IntelliJ IDEA software, it inherite most of his features (like code completion, refactoring, debugging but also built-in tools and a plugin ecosystem). These features, combined with the fact that it is available for download for windows, macOs and linux, has helped to make it one of the most used IDE (together with VsCode) on this field. Android Studio is licensed under the Apache license but it ships with some SDK updates that are under a

non-free license, so it is not completely open source. The supported languages are Java, Kotlin (the actual Google's preferred language for Android app development), C++ and more with extensions, such as Go and Dart[20].

3.4.1 Core features

Several are the core features, based on IntelliJ Idea and then further extended and enriched, that made this IDE a de-facto standard on mobile development:
The **Intelligent code editor features** have been further extended to enhance the developers productivity. Particularly focused on the Android side, there is a clean *Project Structure*, where each project contains one or more modules (Android App, Library and Google App Engine Modules), each one with source code files and resource files. The Android App module in particular then contains the manifest, the source code and then all the non-code resources in the `res` folder. Then, regardless of the mobile platform, Android Studio includes *Debug and profile tools* that help in debugging and improving the performance of code, including inline debugging and performance analysis tools. It is possible to leverage on inline debugging to enhance code walkthroughs in the debugger view with inline verification of references, expressions, and variable values. Also the Performance Profiler allows to easily track memory and CPU usage, find deallocated objects, optimize graphics performance, locate memory leaks and analyze network requests. The Memory Profiler can be used instead to track memory allocation and watch where objects are being allocated when you perform certain actions, since can be useful to optimize the app performance and memory use by adjusting the method calls. Also heap dump is allowed in a specific format, to analyze memory usage and find memory leaks. A solid Code Inspection system is also provided. At compile time, the IDE automatically runs configured lint checks and other IDE inspections to help you easily identify and correct problems with the code quality. The lint tool checks the project source files for potential bugs and optimization improvements for correctness, security, performance, usability, accessibility, and internationalization. In addition to that, IntelliJ code inspections validates annotations to streamline coding workflow. Finally, there is a Log System to view adb output and device log messages when building or running the app and also an Annotation System is supported to annotate variables, parameters, and return values to help find bugs, such as null pointer exceptions and resource type conflicts [21].

The **Flexible Gradle Build System** is employed, with specific Android capabilities provided by the Android Gradle Plugin. Leveraging on the Gradle flexibility allows us to easily manage dependencies, than also customize, configure, and extend the build process, as well as create multiple APKs. All this without modifying the app core source files, only the gradle files (either using groovy or kotlin). By going deeper into the build system it is possible to clearly distinguish some of the main aspects: the *build types*, that define certain properties that Gradle uses when building and packaging your app (for example, the debug build type enables debug options and signs the app with the debug key, while the release build type signs your app with a release key for distribution). At least one build types must be defined and the IDE creates debug and release build types by default; The *product flavors* that are different versions of your app (like free and paid one) that can be released to users: they can be customized to use different code and resources while sharing the common parts. The *dependencies* are managed through the local file system and remote repositories: this eliminates the nedd to manually search, download, and copy binary packages into the project. Related to the APKs, the *Code and resource shrinking tool* allow to shrink code and resources by using its built-in shrinking tools and applying the appropriate set of rules: as result the APK size may be reduced significantly. Finally, the *multiple APK support* allows to automatically build different APKs that contains only the code and resources needed for a specific screen density or Application Binary Interface (ABI)[22].

The **Android Emulator** simulates Android devices on the computer so that it is easy to test an application on a variety of devices and Android API levels without needing to have each physical device. In most cases, the emulator is the best option to test an application on Android (alternatively it is possible to deploy the application on a physical device). Use the Emulator offers *flexibility*, since it is able to simulate lots of devices and comes also with predefined configurations for Android phone, tablet, Wear OS, Android Automotive OS, and Android TV devices. It also offers *high fidelity* because it offers almost all capabilities of a real android device, sich as phone calls, messages, location, play store, networks, sensors and much more. Finally, on some aspects (like data trasfer) it is possible to gain *speed*, since data tranfer is higher to the emulator compared with a real device speed on USB. Each instance of the Android Emulator uses an Android virtual device (AVD) to specify the Android version and hardware characteristics of the simulated

device and each AVD work as a separate device (it is treated independently from the others). Each AVD's data is stored inside a specific directory (directory then used to load the data when booting it up). It is also possible to test an application with WearOs Devices by using the Wear OS pairing assistant that provides a step-by-step guide through pairing Wear OS emulators with physical or virtual phones directly in the IDE. Use the emulator is pretty simple: it is possible to simulate the touch with the mouse in the same way and use the provided buttons on the emulator panel for additional functionalities[23]. The **APK Analyzer** gives more insight into the APK/Android Bundle composition once the build has been completed. By leveragin on this tool it is possible to reduce the debugging time related to DEX files and resources in the app and help reduce the APK size easily. APKs are files that follow the ZIP file format, and the APK Analyzer *displays each file or folder* as an expandable entity that can be used to to navigate into folders. For each entity three metrics are shown: Raw File Size (entity contribution to the total APK size), Download Size (estimated compressed size of the entity as it would be delivered by Google Play) and the percentage of Total Download Size (how much that entity impact on the overall APK size). The tool also allow to *view the AndroidManifest.xml* file, to understand any changes that might have been made to your app during the build. For example, since product flavors and libraries have their own manifest file, this is then merged with the appllication one and the tool helps in visualizing it. It also provides some lint capabilities, with warnings or errors that appear in the top-right corner. The APK Analyzer also provides a *DEX file viewer* that allows to see underlying information in the DEX files (class, package, total reference, and declaration counts), which can assist in deciding whether to use multidex or how to remove dependencies to get below the 64K DEX limit (Also the possibility to Filter the DEX file tree view is provided). Through the APK Analyzer there is also the possibility to *see code, resource entities and textual/binary files* inside the final APK, since during the build process shrinking rules can alter the code, and image resources can be overridden by resources in a product flavor. Finally, the tool allows to *compare different files*: specifically, the tool allows to compare the size of the entities in two different APK or app bundle files, and this is particularly useful to understand why the app increased in size compared to a previous release[24].

3.4.2 New features

As a result of the massive involvement of artificial intelligence and cloud computing, by moving all into the cloud, also Android Studio has adapted by integrating two new major features into the IDE[25]:

- **Gemini**, coding assistant powered by artificial intelligence, that can understand natural language. Helps into achieving more productivity by generating code, finding relevant resources, learning best practices. However it has to be said that like any AI model, sometimes might provide inaccurate, misleading, or false information while presenting it confidently.
- **A Web Version** of the IDE (in early access preview), that leverages on IDX (web-based workspace for full-stack application development made by google itself). It could be used as a convenient way to open up samples but also open an existing project on Github inside the browser.

3.5 Dependencies

3.5.1 Library1 Involved

3.5.2 Library2 Involved

3.6 Integrations

3.6.1 Google Firebase

Firebase Authentication Service Google Firebase Storage Google Firebase RealTime Database

Bibliography

[1] World Health Organization (WHO), https://www.who.int/ [3].

[2] WHO Physical Activity Benefits, https://www.who.int/en/news-room/fact-sheets/detail/physical-activity [4].

[3] WHO Physical Activity Guidelines, https://www.who.int/publications/i/item/9789240015128 [3].

[4] WHO Healthy Diet Guidelines, https://www.who.int/initiatives/behealthy/healthy-diet [5].

[5] National Institutes of Health (NIH), https://www.nih.gov/ [6].

[6] NIH mobile technology studies, https://www.nih.gov/news-events/news-releases/nih-funded-study-examines-use-mobile-technology-improve-diet-activity-behavior [7].

[7] NIH Comparison of Self-Reported and Device-Based Measurements, https://pubmed.ncbi.nlm.nih.gov/33920145/ [7].

[8] NIH Smartphone Applications for Promoting Healthy Diet and Nutrition, https://pubmed.ncbi.nlm.nih.gov/26819969/ [7].

[9] NIH Smartphone Applications for Promoting Physical Activity, https://pubmed.ncbi.nlm.nih.gov/27034992/ [7].

[10] Global wearable band market in Q2 2024, https://www.canalys.com/newsroom/worldwide-wearable-band-market-Q2-2024 [10].

[11] Flutter Framework, https://flutter.dev/ [17].

[12] Flutter Framework, https://docs.flutter.dev/get-started/install/windows/mobile [17].

[13] Dart Programming Language, https://dart.dev/overview [19].

[14] Groovy Programming Language, https://dart.dev/overview [20].

[15] Ruby Programming Language, https://en.wikipedia.org/wiki/Ruby_(programming_language) [20].

[16] Yaml Language, https://en.wikipedia.org/wiki/YAML [21].

[17] Pub Package Manager, https://dart.dev/tools/pub/cmd [22].

[18] Gradle Build Tool, https://en.wikipedia.org/wiki/Gradle [23].

[19] CocoaPods Package Manager, https://en.wikipedia.org/wiki/CocoaPods [23].

[20] AndroidStudio IDE, https://en.wikipedia.org/wiki/Android_Studio [24].

[21] AndroidStudio Code Editor Feature, https://developer.android.com/studio/intro [24].

[22] AndroidStudio Build System Feature, https://developer.android.com/build [25].

[23] AndroidStudio Emulator Feature, https://developer.android.com/studio/run/emulator [26].

[24] AndroidStudio APK Analyzer Feature, https://developer.android.com/studio/debug/apk-analyzer [26].

[25] AndroidStudio New Features, https://developer.android.com/studio [27].

List of Figures