ENSEIRB-MATMECA

S8 Project

S8 Project
Monitoring Cross-platform Application: SuperFlutt

Supervisor :

Mr Charles Consel

charles.consel@bordeaux-inp.fr

Students :

Alexandra Abulaeva, Doha Benzaouia, Marc-Antoine Garcia, Luna Joanie, Hicham Larchi, Tim Teppa

September 29, 2020

Contents

1 Introduction

SuperFlutt project aims at developing a mobile and web application using the cross platform mobile development framework Flutter. Flutter is a mobile development SDK used to create apps for Android and IOS in a single code, in this case, written in Dart. Dart is this new language created by Google, which is used to program Flutter applications.
The application created in this project helps the user to keep track of his activity, on a daily, weekly and monthly basis. So it mainly interacts with sensors, collects data and then displays it in a user-friendly interface.

The aim of this project is twofold. First, to develop this monitoring cross-platform app using Dart and Flutter. Then to assess this new technology, its advantages and limits.

This study was carried out throughout the semester by a group of six students under observation and advice of Mr. Consel, the supervisor. Tasks to be performed were presented before defining other tasks for the following week at the weekly meetings. The work was tested on Web platform and Android platform due to material constraints, IOS app development required an IOS environment (MAC-OS) which is not available.

The report is divided into three main parts. In the first part, the functional description of the application is exposed in order to clarify the operation of the app for users and the different services provided by the app. In the second part, means used to develop the app are described. Each service is detailed and adopted answers are explained for each problem that was faced. Lastly, a review of used technologies is given explaining the benefits of Flutter and Dart, their limits and some possible upgrades.

2 Functional Description of the Application

In this part, a functional description of the app will be given in order to make the use of the app clearer for users.

2.1 Presentation

SuperFlutt is a mobile and web application of activity supervision. This app allows users to see daily, weekly or monthly data about their activity throughout five different services. Its use is quite intuitive, the user can choose in the settings which services are enabled or not and benefit from a friendly-user graphical interface for each service and a global view of data in the homepage (Figure 1). Each functional aspect of the application is then developed during this part.

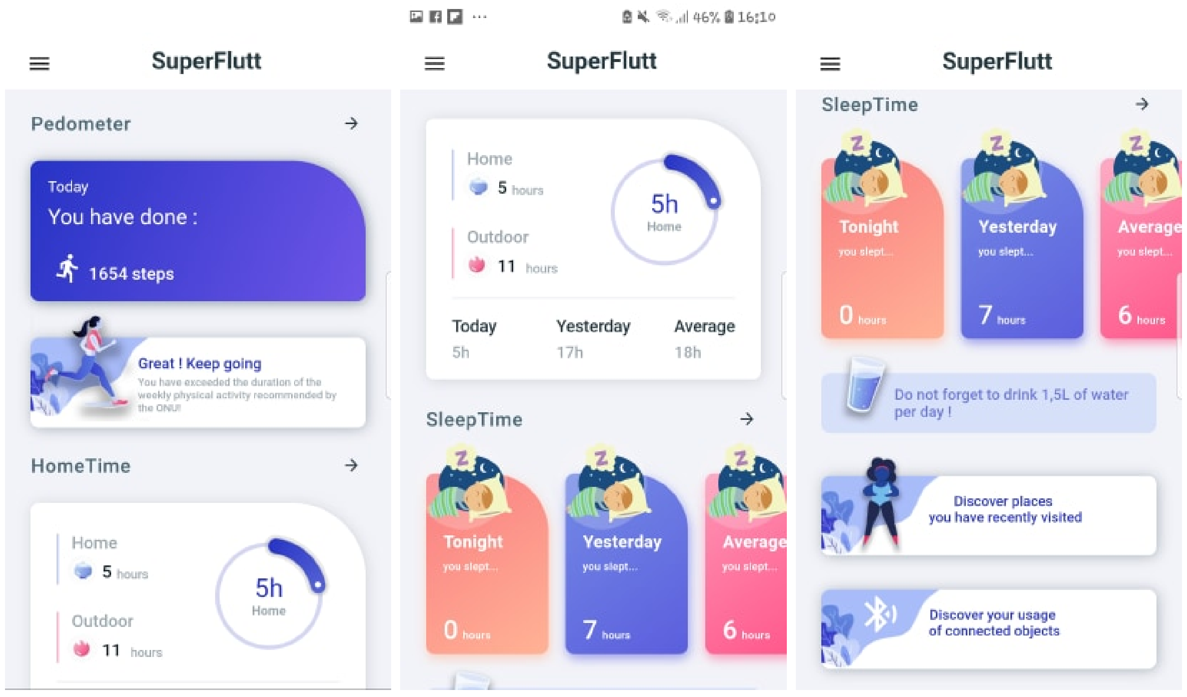

SuperFlutt
Pedometer
Today
You have done :
1654 steps
Great ! Keep going
You have exceeded the duration of the weekly physical activity recommended by the ONU!
HomeTime
Home
5 hours
Outdoor
11 hours
5h
Home
SuperFlutt
Today 5h
Yesterday 17h
Average 18h
SleepTime
Tonight you slept... 0 hours
Yesterday you slept... 7 hours
Average you slept... 6 hours
SuperFlutt
Do not forget to drink 1,5L of water per day !
Discover places you have recently visited
Discover your usage of connected objects

Figure 1: Homepage

2.2 Architecture

First and foremost, the architecture of the app is described in the schema shown in Figure 2. The app has two distinct parts, the Homepage module and the Settings module, which are presented in a navigation bar menu. Settings allow to enable/disable service functions and some particular settings for other services. Homepage module gives an overview of the data collected and allows to navigate to the desired services.

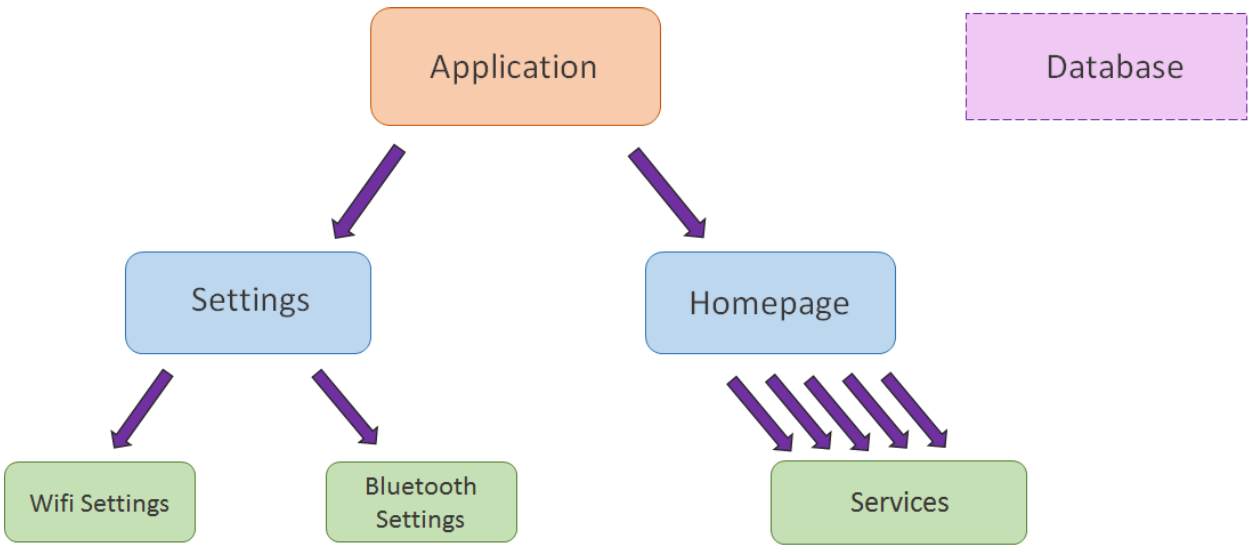


Figure 2: Architecture

2.3 Services and settings

In order to view the data collected, the application offers a daily, weekly and monthly display.

The user can choose whether or not to activate services in settings and take advantage of these different services presented in the following.

Settings

Before displaying the results, users can choose what they want to activate, and set parameters necessary for particular services such as the Hometime service (estimation of time spent by the user at home) that needs a WIFI configuration in WIFI settings or Bluetooth service that needs a special configuration too. These pages are illustrated below.

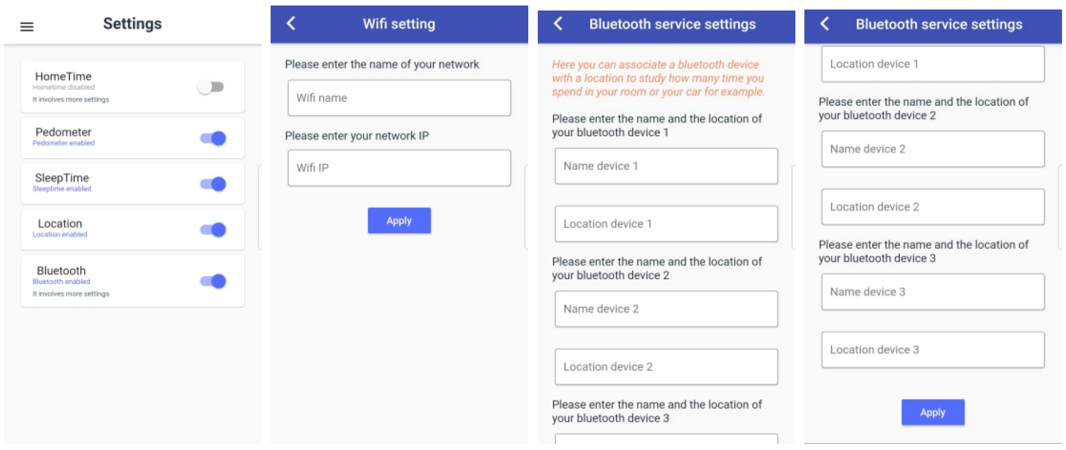

Settings
HomeTime
Hometime disabled
It involves more settings
Pedometer
Pedometer enabled
SleepTime
Sleeptime enabled
Location
Location enabled
Bluetooth
Bluetooth enabled
It involves more settings
Wifi setting
Please enter the name of your network
Wifi name
Please enter your network IP
Wifi IP
Apply
Bluetooth service settings
Here you can associate a bluetooth device with a location to study how many time you spend in your room or your car for example.
Please enter the name and the location of your bluetooth device 1
Name device 1
Location device 1
Please enter the name and the location of your bluetooth device 2
Name device 2
Location device 2
Please enter the name and the location of your bluetooth device 3
Bluetooth service settings
Location device 1
Please enter the name and the location of your bluetooth device 2
Name device 2
Location device 2
Please enter the name and the location of your bluetooth device 3
Name device 3
Location device 3
Apply

Figure 3: Settings

Pedometer

The pedometer service is used to track user's steps on a daily, weekly and monthly basis.

Device sensors count the steps and store values in the memory. It is then possible to access a graph that illustrates the final number of steps for one hour or one day over different periods (Figure 4).

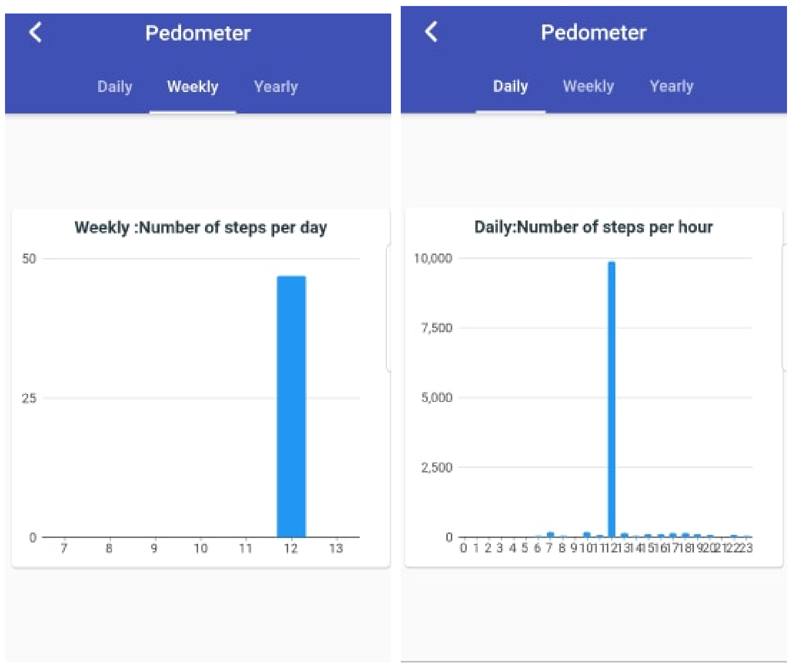


Figure 4: Pedometer

Hometime

Another service provided by the app is Hometime. It allows the user of the application to measure the time spent at home in one day over different periods as is done by the pedometer service during one day, one week or one month for example. The display depends on the period chosen as it is shown below.

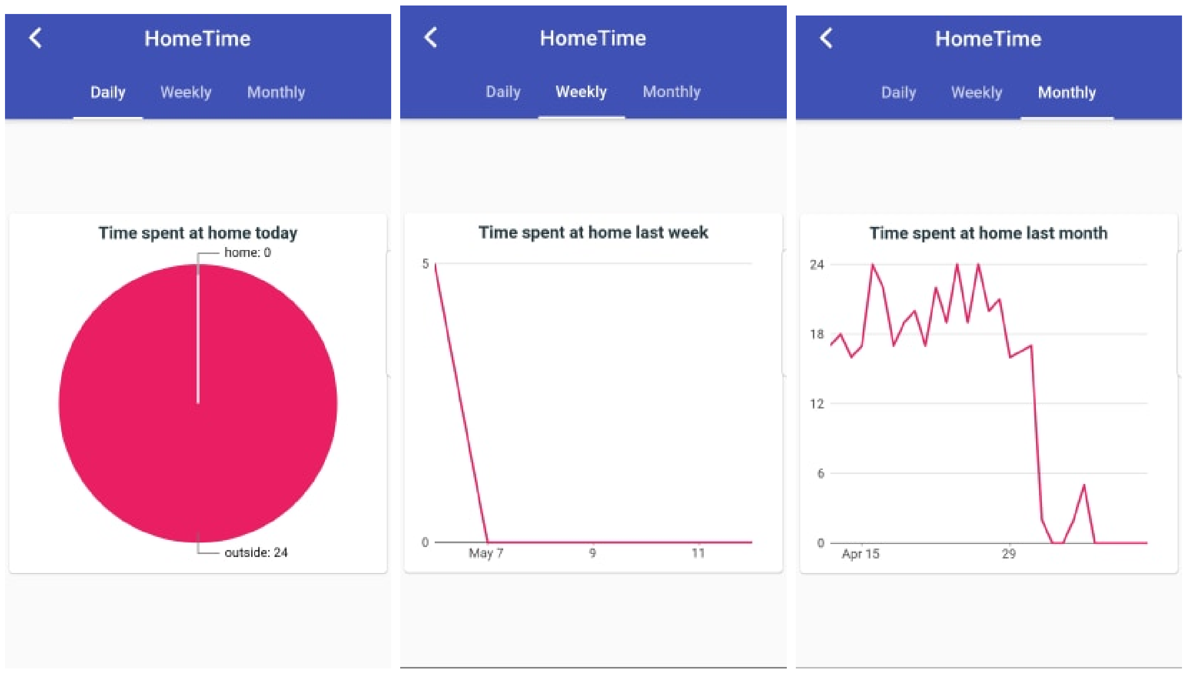


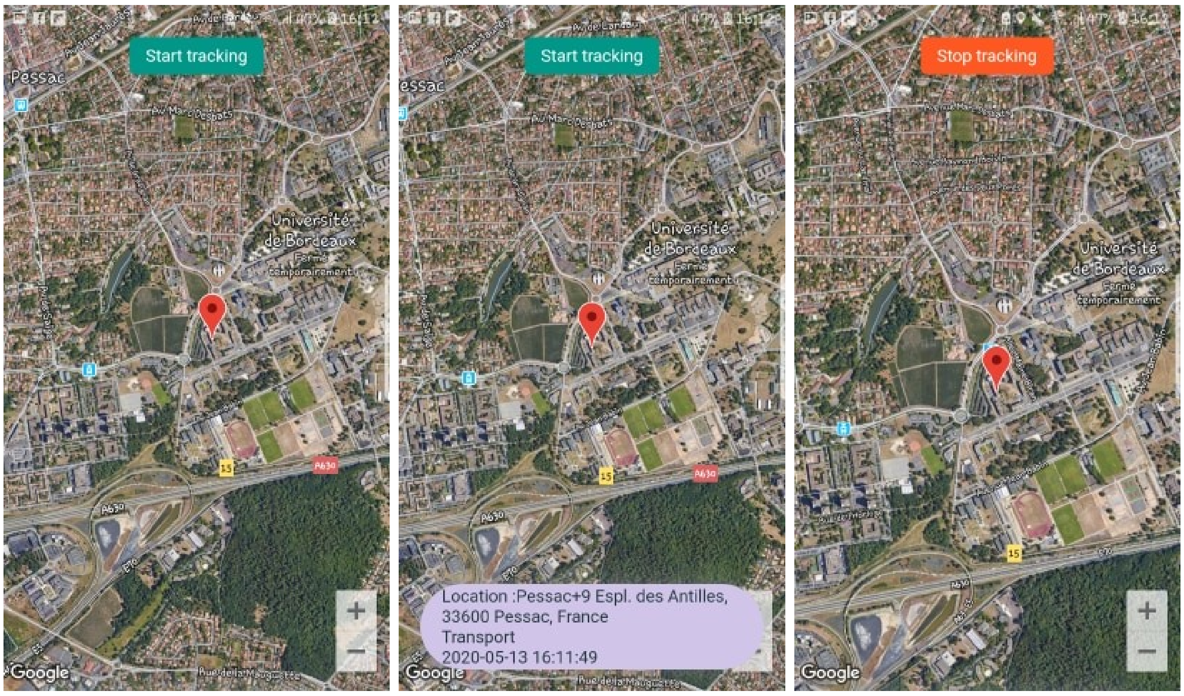
Figure 5: Hometime

Sleeptime

Sleeptime service is very similar to Hometime, it allows the user to measure the duration of sleeping. Sleep duration is approximated to the time spent at home in a dark room with the phone motionless. The same type of graphs seen in Hometime is used to visualize these data.

Geolocation

With the geolocation service, the user can view different locations he visited on a Google map as illustrated in Figure 6. He gets the exact address, the time, and how he got there, either walking, jogging, running or using other means of transport.



Location :Pessac+9 Espl. des Antilles,
33600 Pessac, France
Transport
2020-05-13 16:11:49

Figure 6: Locations

BluetoothTime

The BluetoothTime service measures the duration of its connections to 3 bluetooth devices which are associated to a specific location. First of all, the user has to fill Bluetooth service settings and choose devices associated to a location such as bedroom, bathroom or the car .

All these services allow the supervision of different daily activities of the user for information purposes. However to make this possible and convenient, an intuitive and simple navigation is necessary within the application.

2.4 Navigation

Navigation is one of the main issues faced while developing a mobile application. The complete schema of this navigation is given in the Figure 7.

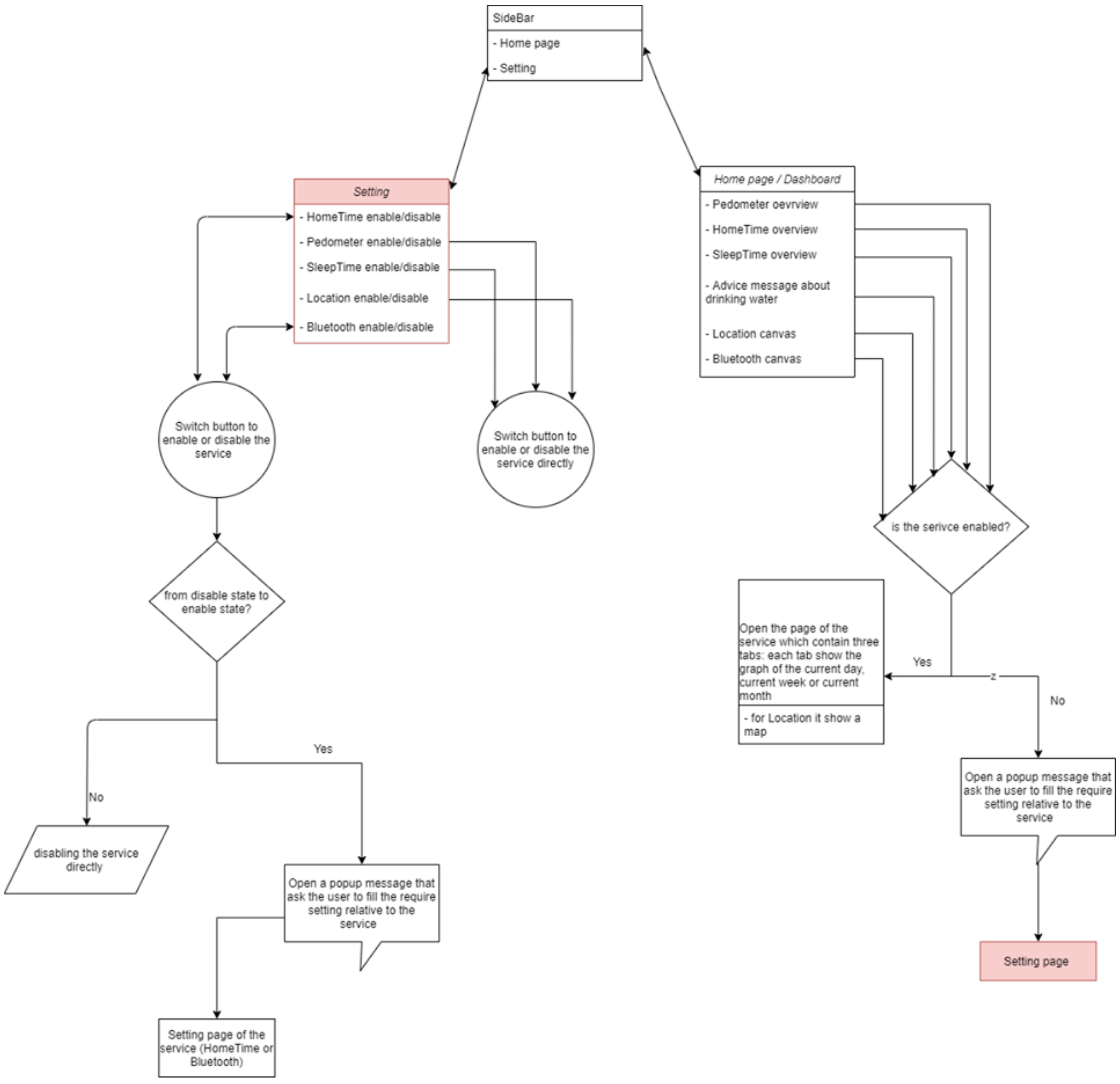

SideBar
- Home page
- Setting
Setting
- HomeTime enable/disable
- Pedometer enable/disable
- SleepTime enable/disable
- Location enable/disable
- Bluetooth enable/disable
Home page / Dashboard
- Pedometer oevrview
- HomeTime overview
- SleepTime overview
- Advice message about drinking water
- Location canvas
- Bluetooth canvas
Switch button to enable or disable the service
Switch button to enable or disable the service directly
from disable state to enable state?
No
Yes
disabling the service directly
Open a popup message that ask the user to fill the require setting relative to the service
Setting page of the service (HomeTime or Bluetooth)
is the serivce enabled?
Yes
No
Open the page of the service which contain three tabs: each tab show the graph of the current day, current week or current month
- for Location it show a map
Open a popup message that ask the user to fill the require setting relative to the service
Setting page

Figure 7: Navigation schema

In particular, it is possible to alternate between the homepage and the configuration page thanks to a side bar in the top left corner such as Figure 8 shows.

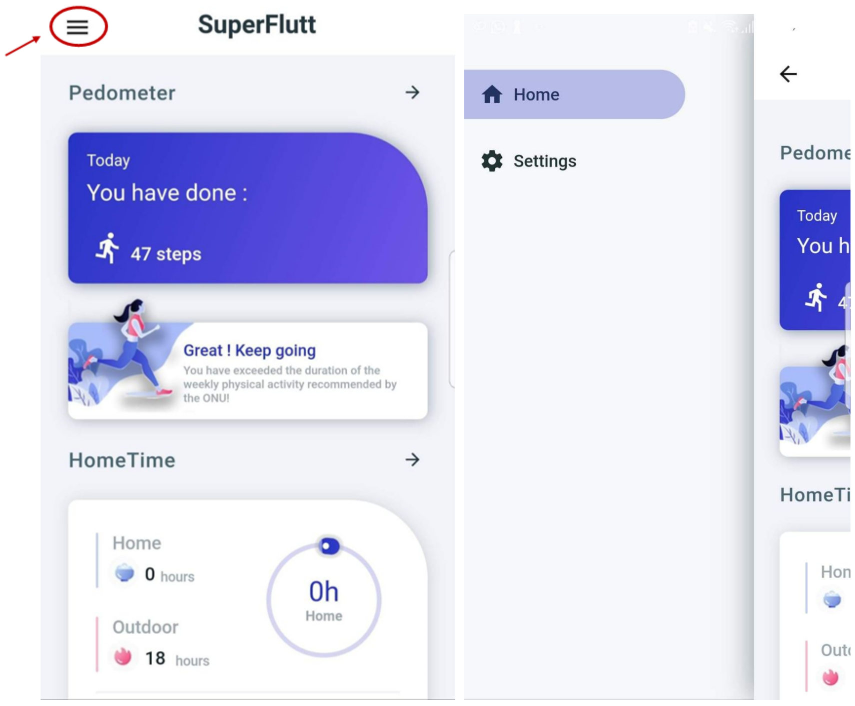


Figure 8: Navigation between home and settings

Subsequently, to access a service in detail, the user can simply touch a field related to the desired service as it is shown below.

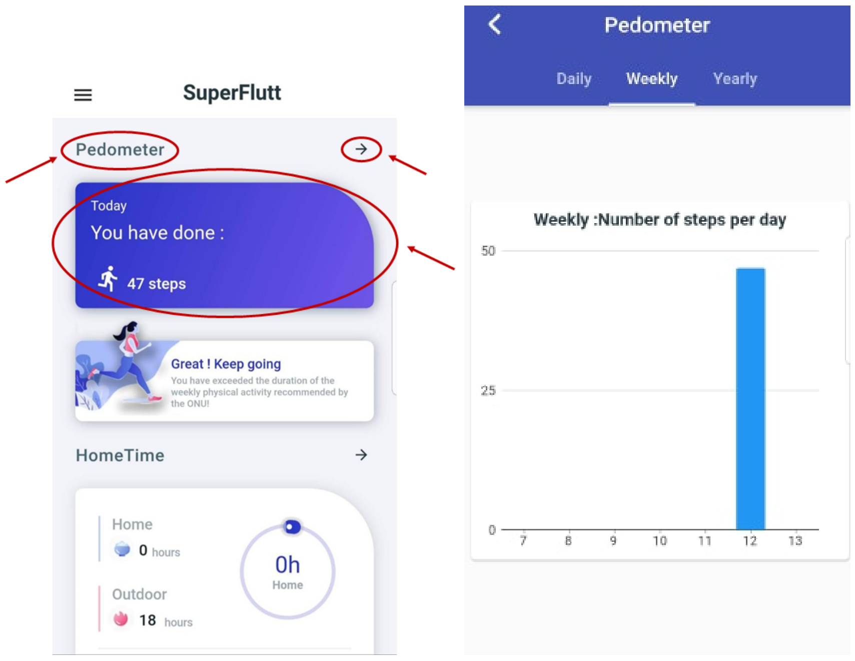
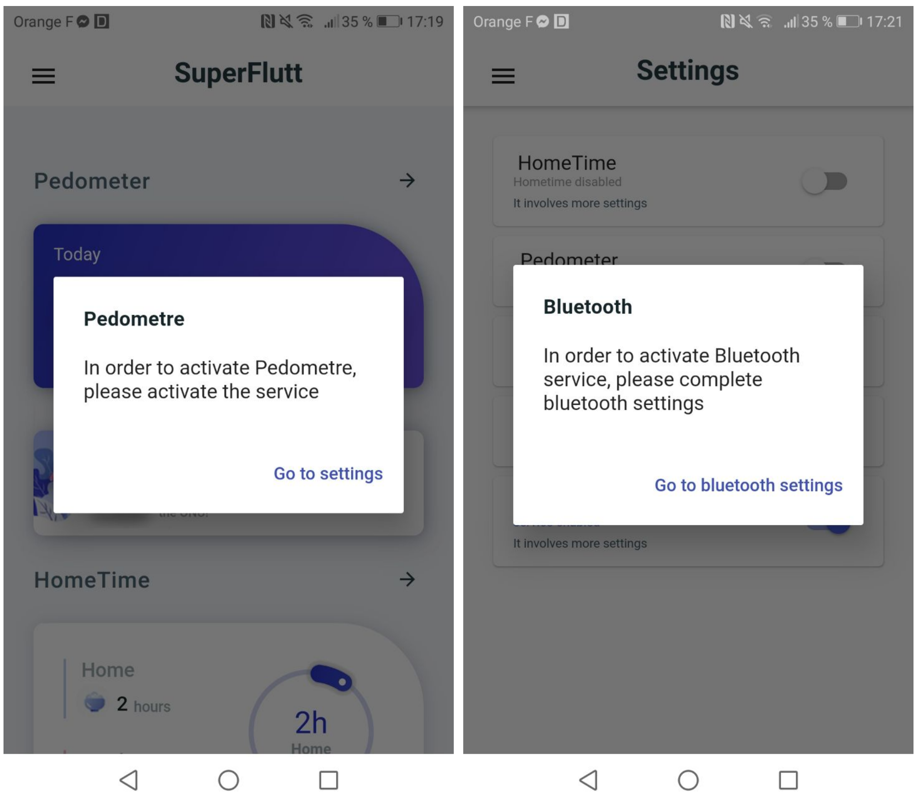


Figure 9: Access to a service

Another tool was developed to simplify the navigation in the app. When a service is disabled and the user clicks on it, a popup appears to direct him to settings. Similarly, when a service that requires specific configurations is activated, a popup appears to take the user to the right configuration page. Figure 10 is an example of these popups.



Pedometre

In order to activate Pedometre, please activate the service

Go to settings

Bluetooth

In order to activate Bluetooth service, please complete bluetooth settings

Go to bluetooth settings

Figure 10: Popup

To achieve a functioning as it is explained before, the difficulty was to take control of the imposed technologies and to find suitable solutions during the development.

3 Development

In the development phase, many problems were faced. It can be broken down into several parts which are data recovery from the phone's sensors, storage of these data, displaying data and routing for the application.

3.1 Storage and databases

Before discussing the different services, it is appropriate to first examine the databases, the solution used to store the collected data. It was necessary to keep the value produced by the sensors, in order to analyze the user's data on a given period.

Consequently, databases were used to store the data of each service. Whenever a sensor detects a particular activity, relevant figures are stored in the right database. This enables two things: the first one is organizing the data received from the sensors. As each sensor has its own database, it makes it easier to access those data in order to analyze them. And The second one is keeping track of the user's activity on a long period. Databases are useful and powerful tools to store a great quantity of data. As all sensors' data are stored in these databases, they make a great history of the user's activity, which can be displayed and analyzed over long periods of time.

The architecture of the database is available in Appendix 1.

Database is a key notion because it is used all over the application (Configuration, services and the design).

3.2 Services and settings development

Settings

Setting pages required an implementation of new classes for the general configuration and for the special Bluetooth service. Then for these pages of settings, other classes that extend StatefulWidget were created, using flutter tools and especially the class form from package: flutter/material.dart. To save the settings, a database was used like the annex 1 illustrates it. One table for general settings with only one line which contains the current general settings and which is editable with an update request. And one table for special Bluetooth service with the number of lines equals to the number of devices. In this case, there are at most three devices editable with an equivalent method.

Pedometer

Implementation Flutter has a pedometer plugin that allows a continuous step counting using the built-in pedometer sensor API of iOS and Android devices. So when the application is opened and the pedometer service is enabled, every step is registrered and stored in order to be visualized.

Bibliography Package pedometer : https ://pub.dev/packages/pedometer

Sleeptime

Implementation As soon as the configuration is complete, the service checks the conditions on the brightness obtained using the light library, on the WIFI connection obtained using the connectivity library and on the forces exerted on the phone obtained using the sensors library .
Each time the phone's WIFI module is changed, the service checks the SSID and IP address of the router for the current WIFI if:
- at least one of the 2 pieces of information coincides with the home WIFI
- the brightness in the room is less than 5
- the sum of the linear acceleration along the 3 axes is close to the terrestrial acceleration
- the sum of the rotational acceleration along the 3 axes is around 0

then a timer is started

Otherwise the service checks the value of the stopwatch, if this value is greater than 0, the service saves it in the database
There is an update every 5 minutes.

Each time a sleep duration is recorded in the database, the service adds the associated date and time which allows it to create several types of visualization.
To view the data collected, the user can access:
- a pie chart for daily display
- a line graph for weekly or annual display

Limitations For this service, there may be some concerns about the connectivity package and the Android version of the phone.
On some versions of Android, it is possible to obtain the SSID directly but for others only the IP

address of the router is available.
This is the reason why the service tests both the SSID and the IP address of the router of the current WIFI.
In addition, the conditions for switching on the chronometer can be met without it being a real sleep duration: the user can be at home with the mobile phone screen on the desk, it may be an incorrect duration.

Bibliography Package connectivity : https://pub.dev/packages/connectivity

Package light : https://pub.dev/packages/light

Package sensors (acceleromètre +gyroscope) : https://pub.dev/packages/sensors

Hometime

Implementation At the implementation level, this service is a clone of the sleepTime service. The conditions on the brightness and on the forces exerted on the telephone are omitted. So only the condition of the WIFI connection is kept.

Limitations Likewise, limitations similar to the sleepTime service.

Bibliography Connectivity package: https://pub.dev/packages/connectivity

Geolocation

Implementation To use this service, the application needs the permission to access the device's location, and the user has to click on a 'start' button to allow the location tracking.
In order to collect data, flutter plugins, such as geolocator, are used. It allows to get the current location of the device with continuous location updates and translates geocoordinates to an address. Every time the location is updated (on a minimum distance of 1Km), the distance traveled, the number of steps, and the duration are registrered and stored in the database. Number of steps divided by the distance enables the application to detect the user's activity, relying on a research study that has

found average steps per mile at walking and running speeds. And when it is combined with the speed (distance divided by duration), it detects whether the user used a mode of transport to reach the location.

Limitations Calculating the distance via geolocation can be inacurate sometimes, especially if the service provider is not very good. In the geolocator package, there is the LocationAccuracy enum, whose best accuracy is supposedly from 0 to 100m for Android, which does not sound good. In addition, detecting user's activity is much more complex since it depends on other factors such as the age and the gender of the user.

Bibliography Package geolocator : https://pub.dev/packages/geolocator
Package geolocation : https://pub.dev/packages/geolocation
google maps : https://pub.dev/packages/googlemapsflutter
Typical steps per Mile walking and running: https://www.verywellfit.com/how-many-walking-steps-are-in-a-mile-3435916

BluetoothTime

Implementation This service uses the bluetooth serial library to find out the status of bluetooth and to obtain the list of devices available for connection.

Once the configuration is complete, the service builds, at regular time intervals, the list of available devices and checks their connection status to the phone.
For each device, the service checks whether:

- the status is connected
- the device corresponds to a device configured
- no other configured device is on registration

If every condition is met, the service saves the name of the device concerned and starts a timer. There is an update every 5 minutes.
When the connection is complete, the service saves the value of the stopwatch, the date, the time and the name of the device in question.
In order to renew another connection, the value of the stopwatch is set to 0, and the backup variable, with the name of the device, is initialized.

Limitations Some limitations exist for this service.
The service can only track the connection time for one device at a time.
When 2 configured devices are connected in parallel, registration is performed for the first available device.
In addition, the application only offers the registration of 3 devices.

Bibliography bluetooth serial package: https: //pub.dev/packages/flutter$_{b}luetooth_{s}erial$

An example of using bluetooth: https://github.com/edufolly/flutter$_{b}luetooth_{s}erial/blob/master/example/lib/$

package shared preferences (backup value): https: //pub.dev/packages/shared$_{p}references$

3.3 Data visualization and graphics

Description In order to use the data retrieved by the different sensors and available in the database, the application allows the user to visualize those data thanks to charts, that can be displayed according to different time scales.

Data retrieving To extract data from the database, some functions were implemented. They allow the user to visualize the data hour by hour, day by day. Or to view it as a mean per day over a given duration.
In order not to be limited in term of data, for a specific duration, for instance a month, the data of the past 30 days are used, instead of using only the data of the current month.

Data visualization The package "charts_flutter" is chosen in order to view the graphs. It provides a wide panel of graphical representations. In order to use that package, it was necessary to know the parameters types that the functions take as arguments, to create additional functions to format the data. Then, to display a graph, the print function had to be in a "widget".

Encountered difficulties The main difficulty is within the type of the data. In fact, the data recorded at unpredictable moment, so they had to be "Future" type, which is an asynchronous type. Furthermore, this type is communicable, if a function uses a "Future", that function returns a "Future" too. However, it is not allowed to use an asynchronous calling in a widget. Actually, the widget is responsible for printing the data, and is unable to print something that may not exist yet.

Usually, the "await" function is used in order to wait for the data to exist. However, it can not be used with a "widget" because of the problem previously explained.
To solve that problem, the "Future builder" class has to be used. It allows to use asynchronous data in a widget by confirming the existence, or nonexistence, of the data, and give appropriate instructions. In the case where the data exists, a graphic is properly printed, if the data does not exists yet, a warning quote explaining that there are no data available is printed. Finally, if the data are incorrect, an error message appears.

Bibliography Package charts_flutter : https://pub.dev/packages/charts_flutter
Chart gallery : https://google.github.io/charts/flutter/gallery
Utilisation of Future Builder : https://api.flutter.dev/flutter/widgets/FutureBuilder-class.html

3.4 Design and routing

Routing

Description Routing is an important part of modern web applications. It is a system for resource navigation and in this case, it meant matching components (pages and tabs).

Implementation Mobile apps typically display full-screen elements called "screens" or "pages". In Flutter, these elements are called routes and they are managed by a Navigator widget. The navigator manages a stack of Route objects and provides methods for managing the stack, like Navigator.push and Navigator.pop.

Navigation in this case was done by two ways : Named routes and pushing Routes explicitly.

Pushing Routes explicitly is done by instantiating a PageRoute and passing it to the Navigator, using Navigator.push() to navigate to the second route and Navigator.pop() to return to the first route.

However, using this approach to navigate to the same screen in many parts of the app, resulted a code duplication. So the solution was to define a named route, and use the named route for navigation.

To work with named routes, the Navigator.pushNamed() function was used instead of Navigator.push(). But first, the routes were defined by providing additional properties to the MaterialApp constructor: the initialRoute and the routes themselves. The initialRoute property defines which route the app should start with. The routes property defines the available named routes and the widgets to build when navigating to those routes.

Bibliography https://flutter.dev/docs/cookbook/navigation/navigation-basics

https://flutter.dev/docs/cookbook/navigation/named-routes

Design

A soft design with smooth animation was used in the UI of the app. Some elements of the home page were picked up from an open project in the internet, and their integration into the app, required the full comprehension of each component. The rest of UI was based on Material design, which is a design language provided by Google inspired by the physical world and its textures, including how they reflect light and cast shadows. Material surfaces reimagine the mediums of paper and ink. Flutter provides a bunch of Material Components widgets ready to use, which makes developing UI in flutter significantly easy comparing to other mobile UI framework.

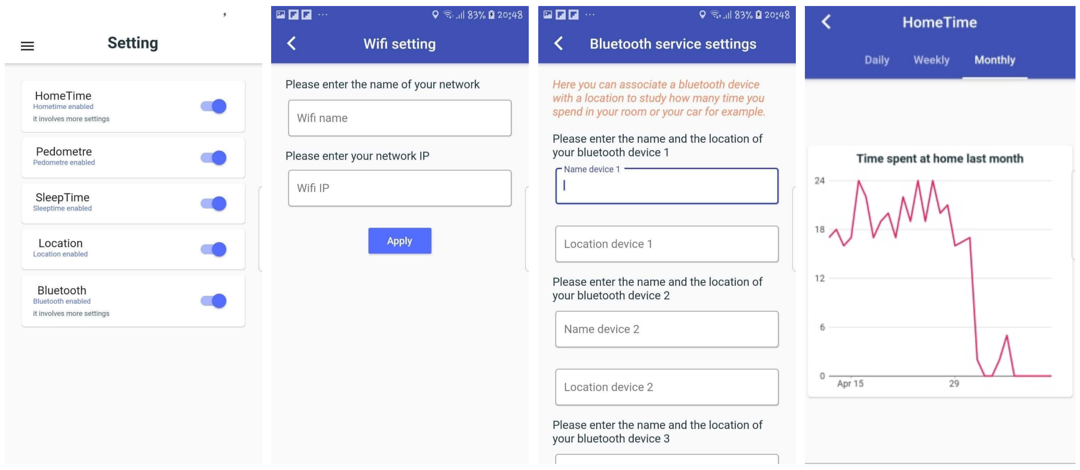
The app provides also a good user experience by giving an overview of each service at the home page.

Switching between home page and settings is available via the sidebar, which can be shown by pushing the icon at the fixed topbar.

At the setting page, enabling or disabling a service is explicit thanks to the switch buttons and the little text under the name of the service. If the user tries to enable a service that requires a specific configuration, the app launchs a popup that asks him to fill required information for the service. The same popup appears if the user tries to access a disabled service from home page.

In the configuration page of WiFi and Bluetooth, when the user fills the forms, the app checks whether the information provided is valid. If the user submits incorrect information, a friendly error message is displayed letting him know what went wrong. If the user has correctly filled out the form, the information is processed and a snackbar provides a brief message at the bottom of the screen.

Inside the detailed page of each service, switching between daily, weekly or monthly data is possible thanks to a list of tabs.



Please enter the name of your network

Please enter your network IP

Here you can associate a bluetooth device with a location to study how many time you spend in your room or your car for example.

Please enter the name and the location of your bluetooth device 1

Please enter the name and the location of your bluetooth device 2

Please enter the name and the location of your bluetooth device 3

Figure 11: Design

Limitations The UI development was based and tested only on some android devices, which does not guarantee the compatibility of the UI with all the devices, and risks to show some graphic errors.

4 Retrospective on used technologies

During this project, Flutter framework was the main tool used to develop our application. What exactly is this tool ? What are its advantages and disadvantages ? The answers to these questions will be given in this part as well as a criticism of this technology.

4.1 Description of Flutter and Dart

First of all, Flutter is an opensource framework for developing native mobile applications created by Google in 2017. It allows to write Android and iOS apps in a single shared code. Flutter uses Android and iOS Sdks to get a native look, whether using Material Design or Cupertino. It is possible to have an application rendering using the material design on iOS, as well as a cupertino application on Android. As with any iOS app development, a macOS machine is required to compile the iOS app. Without disposing any macOS machine, the decision to focus on Android development was taken.

From a programming point of view, Flutter uses the Dart programming language, also invented by Google. Dart is very close to languages such as C++ or java allowing to get a grip on it fast enough. For the editor, Visual Studio Code was privileged with its Flutter plugin it seemed to be a relevant choice. Flutter apps, thanks to Dart, are compiled AOT (Ahead of Time). This makes it possible to generate a native app for Android and iOS. Thus the code is optimized for the architecture of each platform. This feature allows it to stand out from other tools such as React Native.
One of the main advantages is that the user interface will be identical on Android and iOS. Flutter applications are designed pixel by pixel, so the design of the application is not dependent on the OS. But this is also a drawback because the rendering of widgets will not be native to the OS, these are not native components.

In Flutter, programming is reactive, the interface reacts to user inputs by changing properties and/or variables. When these properties are changed, the interface is redrawn according to the new state. It is therefore a modification of the properties that triggers a modification of the UI and not the functions that modify it.

4.2 Widgets

In Flutter, the interface is built using widgets. Each widget describes what their view should look like based on their configuration and status. When the state of a widget changes, the framework calculates the difference with the previous state to determine the minimum changes needed in the rendering tree to move from one to the other. Each widget is a subclass of Statelesswidget or Statefulwidget and must implement the build function which is intended to describe the widget's user interface. it is possible to find the complete catalogue of Flutter widgets on https://flutter.dev/docs/development/ui/widgets.

The so-called stateless widget are widgets that do not support states. It does not depend on anything outside of properties passed to it in its builder. Thus, if a widget property changes, the widget interface will not be redesigned.

The so-called stateful widget, on the other hand, is dynamic. It can be modified depending on inputs or an asynchronous request. it listens to changes and updates the interface to reflect those changes. A setState function is then called and triggers the build call.

4.3 Summary of the advantages and drawbacks of Flutter

Advantages	**Drawbacks**
Easy handling thanks to opensource and proximity to other langages	Widgets are not native to the OS
Flutter apps are compiled AOT (Ahead of Time)	Libraries and support not so rich as for the native development
Identical user interfaces for iOS and Android	Problems with iOS
Reactive programming	Massive File Size
Varied Widget	

Table 1: Advantages and drawbacks of Flutter

To sum up, the advantages of Flutter can also be its flaws, depending on how it is used. Its easy handling makes it possible to make this tool accessible and to develop cross-platform applications using a single language.

5 Conclusion

The main goal of SuperFlutt was to design a cross-platform application that tracks user's activity. The major services implemented are the pedometer, locations, average time spent at home, and time spent sleeping, in addition to data related to connected devices via Bluetooth. Results are displayed in different kinds of charts and forms, on a daily, weekly and monthly basis.

The project was quite ambitious and challenging, especially in the beginning, as it was our first time being introduced to Flutter-Dart environment. However, since both Flutter and Dart are open-source, extensive documentation is provided, and a large community support is available to help out with any issues we could encounter. Moreover, Flutter offers high productivity, with only one codebase that covers both Android and iOS platforms, and also a simple and fast development.

There are some possible improvements to consider as well. It is possible to implement a background process, in order to keep the services running while the application is not open. Some further optimisations can be integrated to enhance the performance and the accuracy of displayed data . Finally, the application can always be improved in order to offer new services and track other user's activities.

6 Appendix

Appendix 1 : Database

Id	adress	elapsedTime	elapsedDuration	diffDuration	distance	coordinates	lat	long	vitesse	pas
1	...	...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...	...

Table 2: Geolocation

Id	duration	theDay	theMonth	theYear	theHour	theMin	thePart
1	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...

Table 3: Sleep

Id	numberSteps	theDay	...	thePart
1	...	...	...	...
...	...	...	...	...

Table 4: Steps

Id	theTime	theDay	...	thePart
1	...	...	...	...
...	...	...	...	...

Table 5: Hometime

Id	name	theTime	theDay	...	thePart
1	...	...	...	...	...
...	...	...	...	...	...

Table 6: Blue

Id	wifiname	wifiIP	hometime	sleeptime	pedometre	location	bluetooth
1	exemple : ”box”	exemple : ”IP”	0/1	0/1	0/1	0/1	0/1

Table 7: Config

Id	name	location
1	device 1	location 1
2	device 2	location 2
3	device 3	location 3

Table 8: ConfigBlue