

Design Document (DD)

MyAir project

Alessandro Mastropasqua / Aristide Bordoli

AA 2020-2021

Table of Contents

1	INTRODUCTION	3
A.	<i>Purpose.....</i>	3
B.	<i>Scope.....</i>	3
C.	<i>Definitions, Acronyms, Abbreviations.....</i>	3
D.	<i>Revision history.....</i>	3
E.	<i>References</i>	4
2	ARCHITECTURAL DESIGN	5
A.	<i>Overview.....</i>	5
C.	<i>Component view.....</i>	6
D.	<i>Deployment view.....</i>	12
E.	<i>Runtime view.....</i>	17
F.	<i>Selected architectural styles and patterns.....</i>	19
G.	<i>User interface design</i>	21
3	UNIT, WIDGET and SYSTEM TEST PLAN.....	29
A.	<i>Unit and Widget test plan.....</i>	29
B.	<i>System test plan.....</i>	30
D.	<i>Unit & Widget Tests.....</i>	31
E.	<i>System Tests</i>	35

1 INTRODUCTION

A. Purpose

1. The Design document defines the technical details related to the design and development of the application software for MyAir.
2. The purpose of the Design document is to document how the functionality will be provided by the new system. It contains development items on a technical level of description.
3. The Design document covers the hardware and software design, as appropriate:
 - i. The software design is the planning and allocation of the software modules and data storage.
 - ii. The hardware design is the definition of the hardware requirement to enable the hardware to be procured / manufactured.

B. Scope

The Design document contains a technical level of description of the project application software / configuration, it contains the following items:

1. Conversion of requirements into a design specification with descriptions of the software modules.
2. Technical details of the software application to be implemented or configured.
3. It provides enough information to implement, install and perform unit and module testing.
4. The Design document builds the committed base for:
 - a) Software Development and configuration
 - b) Software Module development and testing
 - c) System installation

C. Definitions, Acronyms, Abbreviations

Term, Acronym	Definition
DD	Design document
GPS	Global Position System
GUI	Graphical User Interface
AQI	Air Quality Index
Web-API	Application Program Interface over HTTP

D. Revision history

Document Version Number	Document Revision Date	Change Summary (Reference section[s] changed)
1	02-Feb-2021	First version of this document

E. References

References		
#	Document	Version
[1]	https://www.arpalombardia.it/Pages/Aria/qualita-aria.aspx	N/A
[2]	https://pub.dev/	N/A
[3]	https://flutter.dev/?gclid=Cj0KCQiAx9mABhD0ARIsAEfpavSJSi-D_SHc_5MRVutLZMQ_cwu7rwwYFq-GAGq0DIeWwmZYymWrP2EaAms4EALw_wcB&gclsrc=aw.ds	N/A

2 ARCHITECTURAL DESIGN

A. Overview

In this section we highlight the overall structure of the computerized system using the logical-functional view.

The key objectives of the system architecture are as follows:

Definition of the system in terms of computational components and interactions among those components

Analyze the externally visible (observable) properties of components.

Definition of earliest set of design decision such as constraints on implementation, dictate organizational structure and inhibition/enabling quality attributes.

The application related to myAir is based on a client-server architecture.

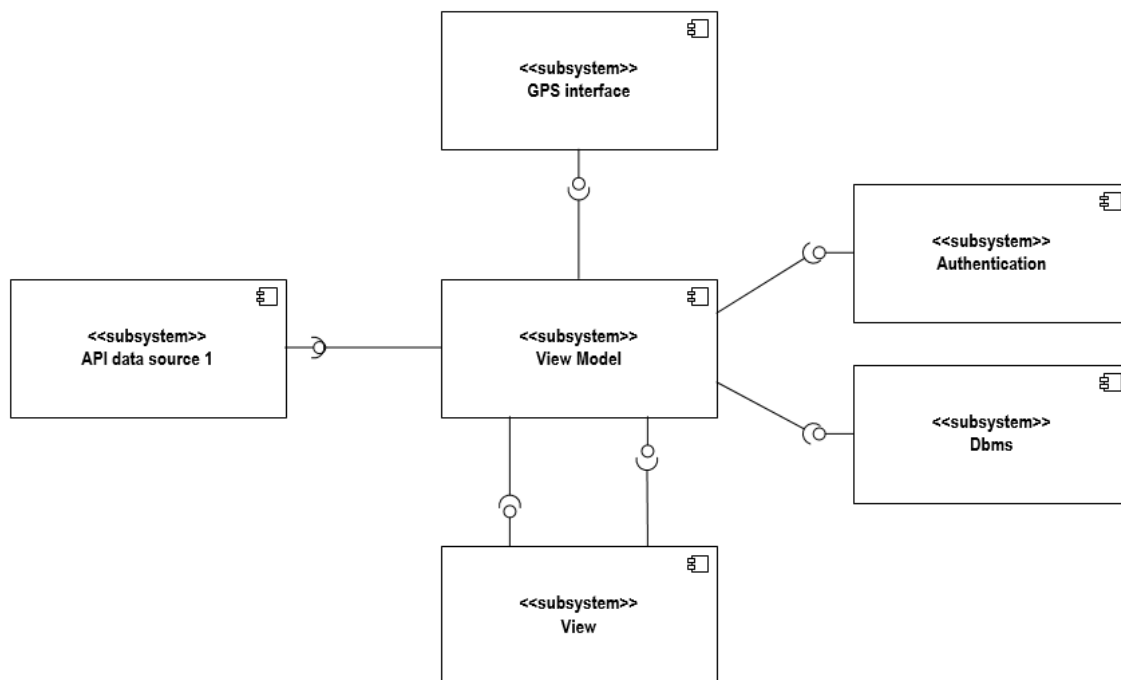
The Model-View-ViewModel is the design pattern selected:

Model: Represents data model, manages data states as business logics

View: Is the way we represent data, renders the UI

ViewModel: It serves as intermediary between View and Model, it provides data to the View via bindings, notifies View of updates, handles View logic, and calls methods on the Model

The client is connected with the data sources using dedicated Web API over HTTP protocol.

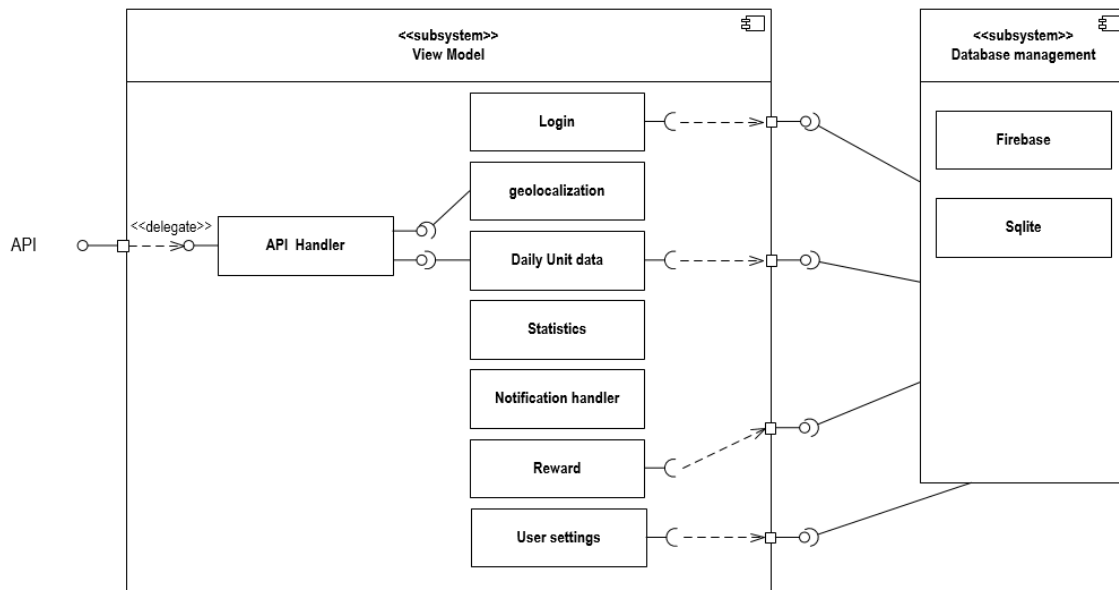


C. Component view

In the component diagram each element is not reported only by its name.

The information related to the main methods and interfaces are included in the analysis in order to capture the physical structure of the implementation.

The component view is usually developed by architects and programmers.



API Handler:

The application needs to retrieve the list the sensor and for each sensor needs to retrieve the related data.

The below APIs have been included in the application:

fetchSensorsFromAPI() includes the Web Api '<https://www.dati.lombardia.it/resource/ib47-atvt.json>' to retrieve the list of the sensors.

fetchSensorsFromAPI() includes the Web Api '<https://www.dati.lombardia.it/resource/nicp-bhqi.json>' to retrieve the last 24h data related to a selected sensor.

Login:

This module manages the login on the user.

The user can create a local account or can use the google service.

Geolocalization:

This module manages the geolocalization of the user in real time.

As soon as the position of the user is defined, the related sensors close to the user are retrieved and the data for each sensor are selected according to the closer sensor.

Daily Unit Data:

The daily average is calculated according to the data retrieved from the selected sensors.

Statistics:

The actual data are represented using an animated chart.

The bar chart is used to display the average data calculated by the daily unit data.

Notification Handler:

The system automatically notifies the user in case of the data sensor reach a predefined limit.

Another kind of notification is related to the completion of a reward configured in the system.
Both of the notifications can be disabled.

Reward:

Different rewards have been configured according to the value of the sensor data and related to the behaviour of the user.

User settings:

A dedicated section is foreseen in order to perform different settings:

- Modification of the name displayed in the App
- Change of the password
- Enable/Disable the notifications
- Set the values for the rewarding

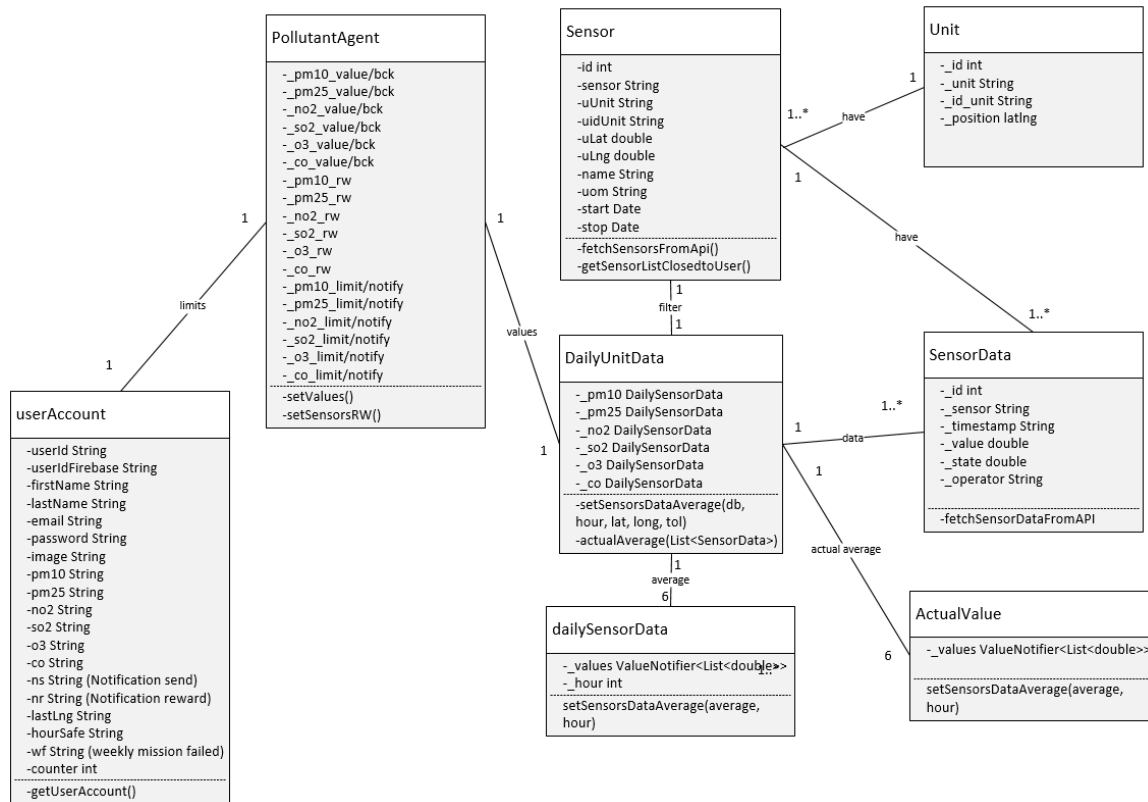
Database Management:

Two different data repository have been configured: Firebase and SQLite.

Firebase is used for user account repository of data.

SQLite records the user account local data and the data related to the sensors.

A class diagram has been included in order to map out the structure of the system by modeling its classes, attributes, operations and relationship between objects.



Sensor:

The class contains the list of the sensor.

fetchSensorsFromApi() retrieves the list of the sensor from the dedicated API.

getSensorListClosedToUser() selects the sensor closed to the position of the user.

SensorData:

It is the class containing the time-series data related to each sensor selected based on the position of the user.

fetchSensorDataFromApi() retrieved the last 24 values for the selected sensor.

Unit:

It is the class containing the list of the stations to be displayed on the map widget.

The station is the logical name that represent the list of sensors with the same location.

DailySensorData:

The class contains the average calculated at each hour for the type of sensor it represents.

setSensorDataAverage() set the actual average of the sensor every time a new value is retrieved through the dedicated Api.

DailyUnitData:

It is the collection of the sensors PM10, PM2.5, NO2, SO2, O3 and CO of type DailySensorData.

setSensorsDataAverage() calls actualAverage() in order to calculate the average for each sensor. It also stores the actual data for each sensor in the ActualValue class in order to be displayed on the dedicated widget.

ActualValue:

The actual average value for each sensor is stored in the Actual Value class in order to be retrieved by the dedicated widgets which display the related information.

Set SensorsDataAverage set the average values in the data structure.

PollutantAgent:

It is the class used for the reward management.

The actual data for each sensor is normalized and used in the reward calculation.

setValues calculates the actual based on the hour parameters.

As soon as the day changes a backup of the data is performed.

setSensorsRW updates the value of the reward for each type of sensor and manage the notification as soon as the calculated value reach the related limit.

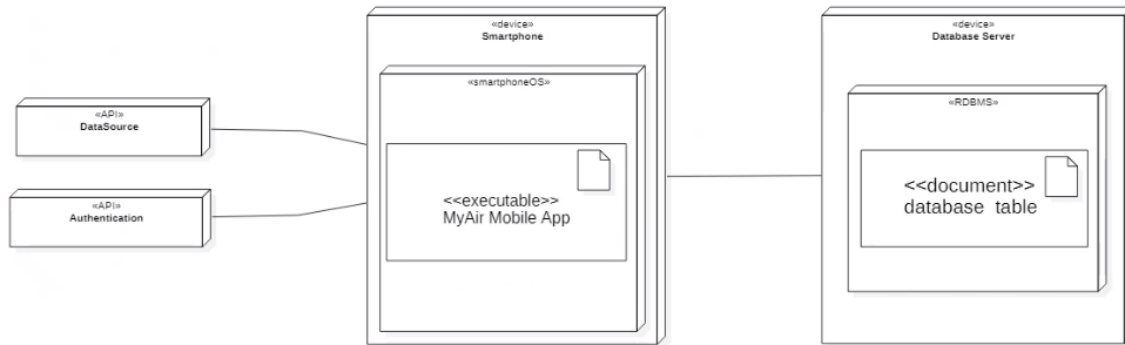
UserAccount:

All the information related to the user are stored in this class. The User table has been created in the Firebase and SQLite databases.

Widget list and structure	
Page Name	Page Composition
Main()	<ul style="list-style-type: none"> - MyApp - StatefulWidget - Initialization() - PermissionPage() only if !permissions - ProfilePage() only if actualUser=null - HomePage() only if actualUser!=null - SplashScreen() in the waiting time
Permissions:	View/PermissionPage <ul style="list-style-type: none"> - PermissionPageWidget – StatefulWidget
Profile Page:	<ul style="list-style-type: none"> - ProfilePage – StatelessWidget - HomePage – StatefulWidget if the user logged in and it has been created in the local db - SignUpWidget – StatelessWidget only if the local user is not yet log in <ul style="list-style-type: none"> - GoogleSignUpButtonWidget – StatelessWidget - SignUp Button - Login TextButton
Registration Page: View	View/RegistrationPage – StatelessWidget <ul style="list-style-type: none"> - RegistrationPageWidget – StatelessWidget: <ul style="list-style-type: none"> - LogoImport – StatelessWidget - RegistrationForm – StatefulWidget
Login Page: View	View/LoginPage – StatelessWidget <ul style="list-style-type: none"> - LoginPageWidget – StatelessWidget <ul style="list-style-type: none"> - LogoImport – StatelessWidget - LoginForm – StatefulWidget
Home Page: TabBar	<ul style="list-style-type: none"> - HomePage – StatefulWidget - ThemeSwitchingArea <ul style="list-style-type: none"> • ChartPage() – Index 0 • MapPage() – Index 1 • UserPage() – Index 2 • RewardPage() – Index 3 • HomeStatisticsPage() – Index 4 (default) - FloatingActionButton → HomePage()
Home Statistics Page: View	<ul style="list-style-type: none"> - HomeStatisticalPage – StatelessWidget - GraphWidget() - StatelessWidget <ul style="list-style-type: none"> - Card 1 <ul style="list-style-type: none"> - Title + Info - AgentListWidget - PieChart <ul style="list-style-type: none"> - PieChart.info: ActualValue.value - Container.DecorationImage: _aqi = PollutandAgent.getAqi() - Card 2 <ul style="list-style-type: none"> - GridAgentWidget – StatelessWidget <ul style="list-style-type: none"> - AgentInfoWidget(6) – StatelessWidget - Flexible -> child: value.name

	<ul style="list-style-type: none"> - Flexible -> AgentPieChart, FractionallySizeBox (value-amount) <ul style="list-style-type: none"> - AgentPieChart extends CustomPainter - valueListenable: ActualValue.value.elementAt(index)
Charts Page: View	ChartPageWidget – Stateless Widget ListView() <ul style="list-style-type: none"> - AnimatedChart() <ul style="list-style-type: none"> - AnimatedChart – StatefulWidget - ‘Overview of your day’ : Text - ScrollableTabBar() <ul style="list-style-type: none"> - ChartCardWidget <ul style="list-style-type: none"> - MinimizePreview <ul style="list-style-type: none"> - BarChartPreview - ExpandedPreview <ul style="list-style-type: none"> - BarChart
Map Page: View	MapPageWidget – StatefulWidget <ul style="list-style-type: none"> - SearchWidget – StatelessWidget - SearchBackWidget – StatelessWidget - SearchableDropDownWidget – StatefulWidget - StationInfoWidget - StatelessWidget
Rewarding Page: View	RewardPageWidget – StatelessWidget <ul style="list-style-type: none"> - ActiveReward – StatelessWidget: one per each pollutant - Text(“Weekly Challenges”) - ActiveReward – StatelessWidget: <i>My Air is better</i> - ActiveReward – StatelessWidget: <i>Daily Access</i> - ActiveReward – StatelessWidget: <i>Don’t touch the bottom</i>
Account Page: View	AccountPageWidget – StatefulWidget <ul style="list-style-type: none"> - ProfileInfo <ul style="list-style-type: none"> - ChangeImage StatefulWidget - Text for personal data and email - Theme Switcher - ProfileListItem – Stateless Widget(Help&Support, Settings, Logout) – StatelessWidget
Help & Support Page : View	HelpSupportView – StatelessWidget <ul style="list-style-type: none"> - HelpSupportWidget – StatelessWidget - SendMailFromLocalHost – StatefulWidget
Settings Page: View	SettingsPage View - StatelessWidget <ul style="list-style-type: none"> - SettingsPageWidget – StatefulWidget <ul style="list-style-type: none"> - NameChanger - StatefulWidget - PasswordChanger – StatefulWidget - OpenPassword - StatefulWidget - ClosePassword - StatelessWidget - SliderAgent - StatefulWidget

D. Deployment view



We have developed the application using Flutter framework and Dart language.

The deployment diagram shows the architecture of the system.

A brief view of the system:

The services used by the application in order to retrieve the external data:

- fetchSensorFromAPI: retrieve the list of the sensors from the provider
- ferchSensorDataFromAPI: retrieve the last 24 hours data for a selected sensor

The below databases has been provided:

- Firebase realtime database in order to store the user information
- SQLite database to store the list of the active sensors, the user table and the 24 daily data for each sensor

Geolocalizator service:

- To retrieve the current position of the user

Authentication service:

- To sign in with google
- To create a new user
- To login as existing user

User interface service:

- Home statistics page with realtime values for each sensor. The cumulative information grouping the data for each sensor is provided.
- Charts page with graphically representation of the average of the sensors for the last 24h
- Map page with the map centered using the position of the user. The pins represent the sensors retrieved from the API. Selecting the pin the related geo-coordinates are displayed. The user can zoom and move around the map. A button is present in order to go back to the current position. A searchable dropdown list is used to find the sensors configured in the system.
- Rewarding page with the information related the reward for each sensor and related to the gesture of the user
- User setting page for the settings of the user related to notifications and limits for the rewarding management. An help&support and the logout from the application features can be selected in this page

All the plugin used in the project with a summary of their utility are listed below:




Plugin Name	Use	Reference Link
url_launcher	It is used in the Help&Support page in order to redirect the user on our social and in the official ARPA page.	https://pub.dev/packages/url_launcher
mailer	It is used by the SendMailFromLocalHost with the purpose of send an email with the google mail service.	https://pub.dev/packages/mailer
Toast	It is used to notify the user the status of its mail in the SendMailFromLocalHost.	https://pub.dev/packages/toast
Image_picker	It is used in the changeImage Widget in order to pick the image from the gallery of the user or taking a photo with the camera and convert them in strings of bytes.	https://pub.dev/packages/image_picker
Path_provider	It is used to manipulate the file that contains the image of the user in the ChangeImageWidget	https://pub.dev/packages/path_provider
Animated_Theme_switcher	It is used in the Account Page by the ThemeSwitcher Widget in order to change the theme of the application with one icon .	https://pub.dev/packages/animated_theme_switcher
Syncfusion_flutter_charts	It is used in the ChartPage View by the baChart and barChartPreview widget to display a customize and dynamic Bar Chart .	https://pub.dev/packages/syncfusion_flutter_charts
Firebase_auth	It is used by the the GoogleSignIn Service in order to request, fetch and check the user google account.	https://pub.dev/packages/firebase_auth
Google_sign_in	It is used to check that the user is connected with his google account in the GoogleSignIn Service.	https://pub.dev/packages/google_sign_in
Firebase_database	It is used by the FirebaseDatabaseHelper Service in order to manage the remote database containing user accounts.	https://pub.dev/packages/firebase_database













Flutter_map	It is used by the MapPageWidget in order to display and manage map and pins placed on it.	https://pub.dev/packages/flutter_map
User_location	It is used by the MapPageWidget to manage the user location in the map and handle the zoom and positioning options. (Flutter_map and user_location plugins are often used simultaneously).	https://pub.dev/packages/user_location
Geolocator	It is the main plugin in the application and it is used by the GeolocationService in order to retrieve the user position every 30 seconds.	https://pub.dev/packages/geolocator
Searchable_dropdown	It is used in the SearchWidget in the Map with the purpose to provide a dynamic search tab list.	https://pub.dev/packages/searchable_dropdown
Permission_handler	It is used by the PermissionPage and the GeolocationService in order to know the current status of permissions regarding the location of the device	https://pub.dev/packages/permission_handler
Flutter_spinkit	It is used by the SplashScreen in the main to provide an animated loading screen.	https://pub.dev/packages/flutter_spinkit
Flutter_local_notifications	It is used by the PopUpNotification Widget in order to send local notification to the device of the user.	https://pub.dev/packages/flutter_local_notifications
Bordered_text	It is used in the LogoWidget	https://pub.dev/packages/bordered_text
Percent_indicator	It is used in the ActiveReward Widget to provide a circular percentage indicator widget	https://pub.dev/packages/percent_indicator
Sqflite	It is used by the DatabaseHelper Service in order to store all the information of the sensors and users. (For what concern the	https://pub.dev/packages/sqflite

	presentation we have create an additional tabel in order to have real values to show)	
Google_font	It is used in the HomeStatisticsPage to have a wide choice of fonts	https://pub.dev/packages/google_fonts
Intl	It is used in the entire application to Format the dateTime	https://pub.dev/packages/intl
http	It is used because has a set of high-level functions and classes that make it easy to consume HTTP resources	https://pub.dev/packages/http
Provider	It is used to have a wrapper around InheritedWidget to make them easier to use and more reusable.	https://pub.dev/packages/provider
Line_awesome_flutter	It provides a wide choice of icons	https://pub.dev/packages/line_awesome_flutter
Auto_size_text	It is used in order to resize the font size in text on multiple lines	https://pub.dev/packages/auto_size_text
Fl_chart	It is used by the AnimatedChart in the Graph Page	https://pub.dev/packages/fl_chart
Font_awesome_flutter	It is used to have a wide choice of fonts	https://pub.dev/packages/font_awesome_flutter

Note: This application was developed using the Android studio development environment and testing it only on devices with the Android operating system. As for the iOS operating systems, we have done everything possible to provide the same features, both in terms of plugins and graphics, but we do not guarantee their operation as we are unable to test them given the development operating system used (Windows).

For what concern the Android operating system, the application works on:

Android Release Name	API Level	Targer	Run
 R	30	Android 11.0 (Google APIs)	✓ It works perfectly
 Q	29	Android 10.0 (Google APIs)	✓ It works perfectly
 Pie	28	Android 9.0 (Google APIs)	✓ It works perfectly

 Oreo	27	Android 8.1 (Google APIs)	 It works perfectly
 Oreo	26	Android 8.0 (Google APIs)	 It works perfectly
 Nougat	25	Android 7.1.1 (Google APIs)	 It works perfectly
 Nougat	24	Android 7.0 (Google APIs)	 It works perfectly
 Marshmallow	23	Android 6.0 (Google APIs)	 It works perfectly
 Lollipop	22	Android 5.1 (Google APIs)	 Application doesn't work. Geolocation plugin doesn't take the user position

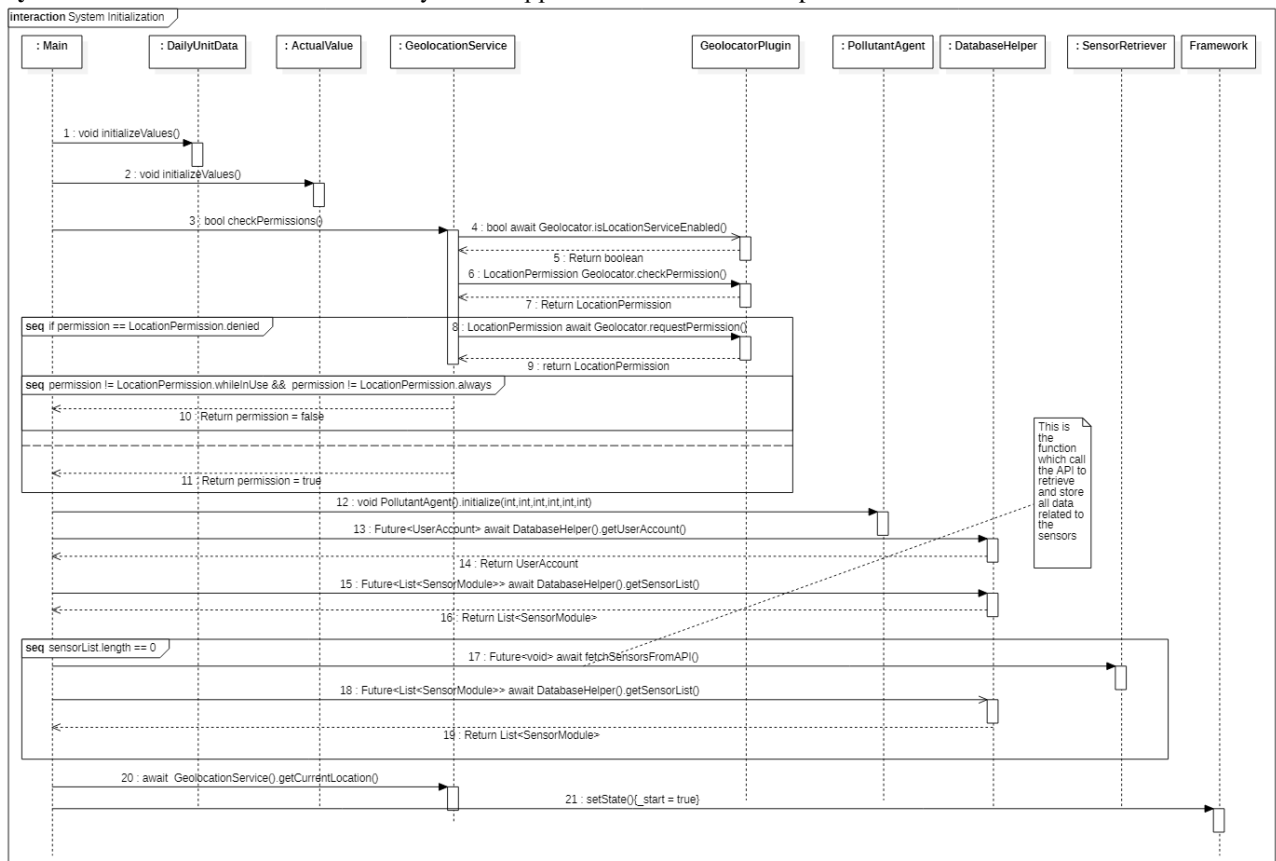
E. Runtime view

A runtime view has been inserted in order to understand how the customize code operates and a reference for troubleshooting.

The main criterion for the choice of possible scenarios is their architectural relevance.

A representative selection of them has been documented in order to verify the system's behavior on its most important external interfaces and the system's behavior in the most important critical situations.

System initialization: this is a summary of the application's initialization process.

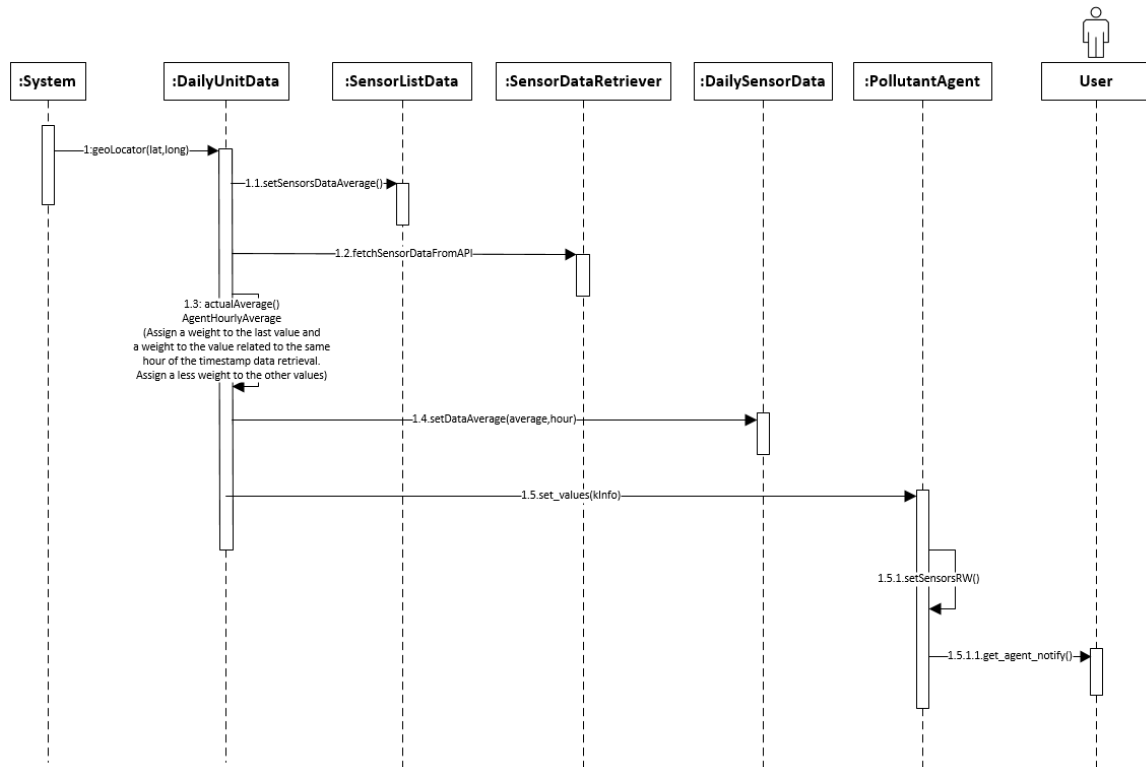


Initialization: explanation of the method and their interaction

- DailyUnitData initialization setting to zero the 24h values in DailySensorData
- ActualValue data initialization with the default values
- An instance of the GeolocationView is created
- userPosition actual user position
- DailyUnitData reference of the daily unit data
- Notifications reference to the notification class
- Check geolocator permissions (deniedForever, denied, whileInUse, always)
- Get shadow user account from Firebase for mailing
- Set the default reward limits in the PollutantAgent class instance
- getUserAccount from the local database SQLite
- getSensorList from the local database SQLite
- fetchSensorsFromApi if not found in the local database

- getLocation from GeolocationService

The scenario below is related to the sensor and sensor data management.



F. Selected architectural styles and patterns

For the MyAir project we have selected the Client-Server architecture where the client is based on the design-pattern MVVM(Model-View-ViewModel):

Model: Represents data mode, manages data states has business logics

View: Is the way we represent data, renders the UI.

ViewModel: It serves as intermediary between View and Model, it provides data to the View via bindings, notifies View of updates, handles View logic, and calls methods on the Model

Model

- Arpa: the arpa service are divided in: SensorDataRetriever and SensorRetriever. Their purpose is the connection to the ARPA databases and, first, get all the active sensors in the region, and after query the single sensor in order to retrieve the information about the AQI.
- Databases: for what concern the database services used in MyAir, we can separate data in two parts: the data residing in the local database, used by the individual user, and the relevant data for MyAir such as the user accounts that allow profiling of the latter. As for the data used by the application such as the list of sensors, they are managed by our Databasehelper Service while the data on the users' accounts reside on a Real-Time database provided by firebase which also contains the access credentials to the ShadowUser's google account.
- Geolocator: The geolocation service manages and directs most of the work. Its main task is to request the user's location every 30 seconds and subsequently initiate the data analysis and modification routines. This thread starts with the application opening and restarts every 30 seconds thanks to this function:

```
GeolocationService._internal(){
  this._notifications.initNotifications();
  timer = Timer.periodic(Duration(seconds: 30), (Timer t) =>
    getCurrentLocation());
}
```

- Sensor: is the class used to enclose the data concerning a single sensor and operate on them(getter, setter, toMap).
- SensorData: is the class used to enclose the data retrieved by the sensors.
- UserAccount: is the model of the application's user. It contains all the method to retrieve/update it from the FirebaseDatabase and Local database.
- ActualValue: It contains the actual values calculated by the DailySensorData

View

- Login_view
- AccountPage
- Registration_Page
- ProfilePage
- Permission_view
- HomePage
- Graph_view
- MapPage
- Reward_Page
- Setting_view
- Help&Support_view

View-Model

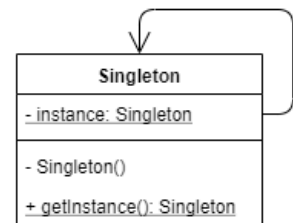
- DailyUnitData: this class is in charge of collecting the data extracted from the sensors, thanks to retrieveSensorData, and modeling them in a way that gives a reliable value. In fact, thanks to the actualAverage function present in it, it calculates the current value as the average of the last 24 measurements giving greater importance to the measurements that took place at the same time as the current one (as it could be a traffic schedule that increases air pollution) and the last measurement made by the sensor as if we were making a weighted average. this value is then sent to the dailysensordata class which will make an average with all the other values measured in the same hour.
- DailySensorData: contains hourly averages for the last 24 hours.
- PollutantAgent: this class deals with analyzing the values to convert them into user earnings that will be used by the Reward Page view. It also takes into account when to send notifications regarding the completion of a reward.
- SensorListData: this class is fundamental for the application and above all for what concerns the api calls to the sensors. Given a list of sensors, it takes care of creating an ordered list in terms of distance between the sensor and the current position of the user, allowing us to always have reliable data as those of the sensor closest to the user.

Design Pattern: Singleton

We decided to use singleton classes as we needed them not to be replicated. Singleton is a creational design pattern which ensures that a class has only one instance and also provides a global point of access to it. Among the various classes we can find:

- DailySensorData
- DailyUnitData
- ActualValue
- GeolocatorService
- DataBaseHelper
- FirebaseDatabaseHelper

Singleton implementation:



Class Diagram — Singleton

```

class GeolocationService{
    static final GeolocationService _geolocationView =
    GeolocationService._internal();

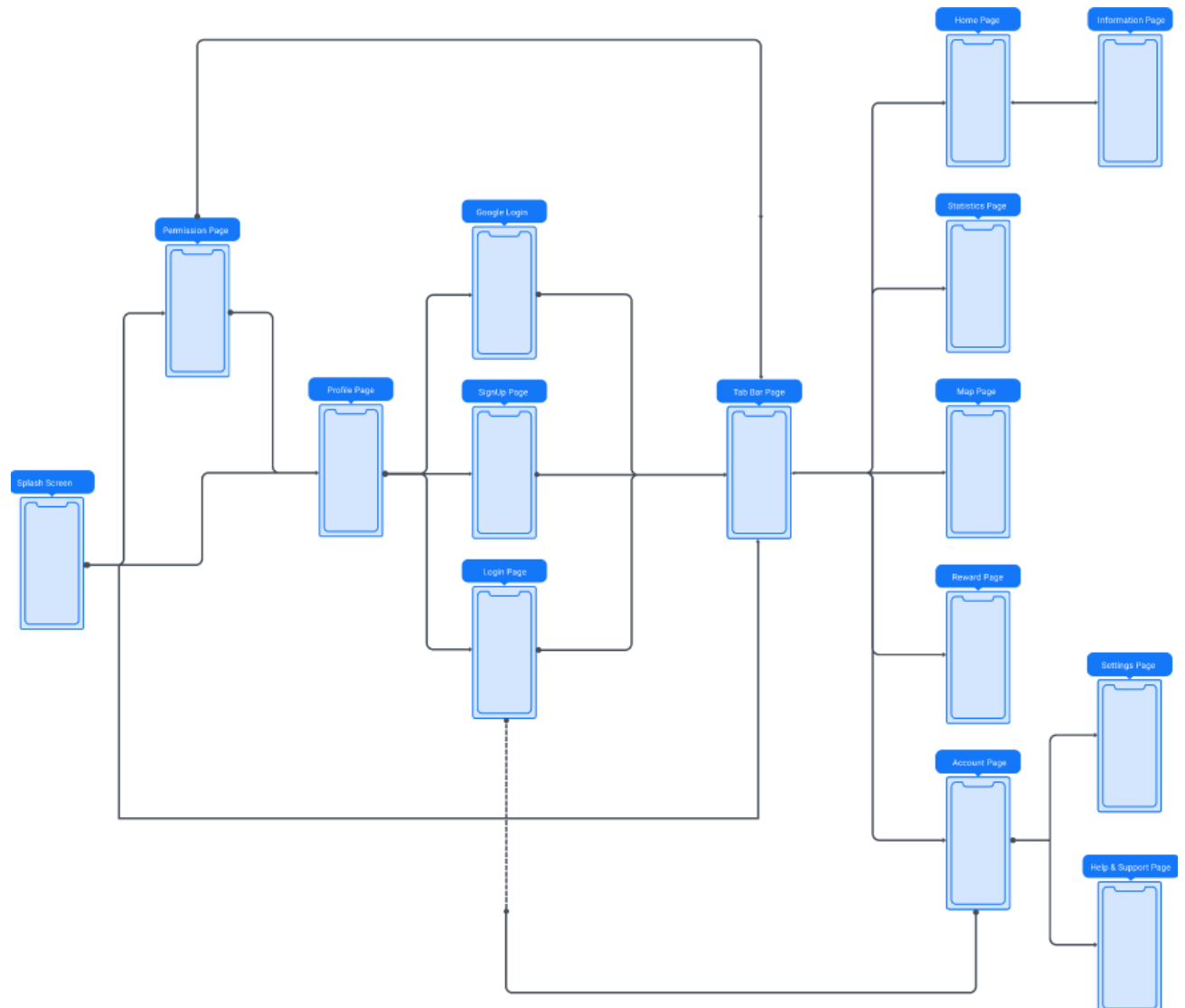
    //We handle it as a SINGLETON
    factory GeolocationService() {return _geolocationView;}

    //Starts the time that call the function every 30 seconds
    GeolocationService._internal(){
        this._notifications.initNotifications();
        timer = Timer.periodic(Duration(seconds: 30), (Timer t) =>
    getCurrentLocation());
    }
  
```

G. User interface design

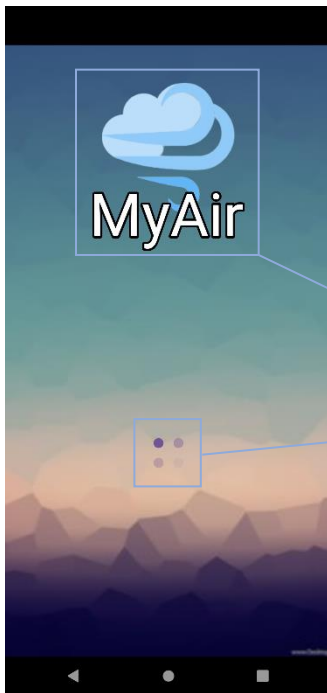
User Interface (UI) Design focuses on anticipating what users might need to do and ensuring that the interface has elements that are easy to access, understand, and use to facilitate those actions.

UI brings together concepts from interaction design, visual design, and information architecture. Below we propose an overview of the application flow, after which we will analyze the interfaces individually.



User Interfaces Description:

Splash screen:

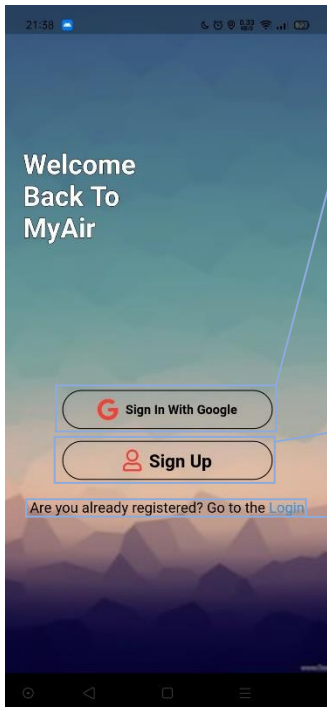


This is the Splash Screen of the application. Every time that the app is opened, the splash screen works until all the initialization of the app data are not done.

LogoWidget: it is a Stateless widget which provide the logo of the application stacked with the name, encapsuled in a BorderedText provided by the bordered_text plugin.

SpinKitFadingFour: a Stateful widget which allows us to simulate a sort of loading screen. It is provided by the flutter_spinkit plugin.

Profile screen:



The profile screen the page allows us to profile users, who have three access methods:

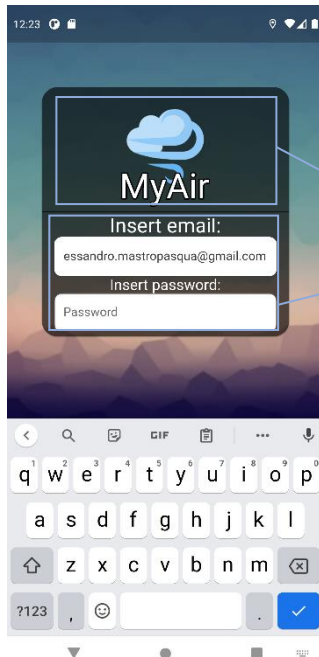
GoogleSignupButton Widget: It allows the user who selects this option to log in through his/her google account. In the event that the user has not registered any google account on his/her device, he will be offered to create it (as the picture on the right shows).



SignUp Button: allows the user to open the Registration Page in which he/she can insert his/her personal data in order to be registered in our databases.

Login TextButton: allows the user to open the Login Page in which he/she can insert his/her credentials in order to access in the application.

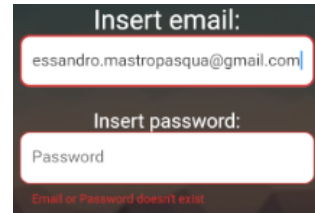
Login screen:



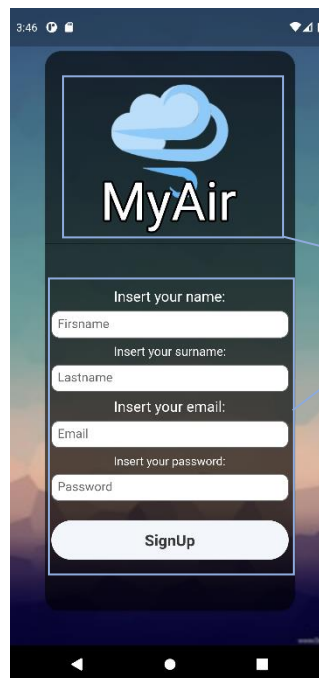
In this page the user has the possibility to make the login in order to access to the home page. As showed in the picture the user can insert his/her email and password in the form adapted as a listView.

LogoWidget: it is a Stateless widget which provide the logo of the application stacked with the name, encapsuled in a BorderedText provided by the bordered_text plugin.

LoginForm: login form used by the user, when the user tab the login button, a function makes the check if the account exists on our Firebase Database. If the user exists, he/she will be redirect to the Home Page, otherwise an error message it is displayed.



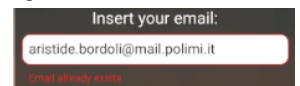
Registration screen:

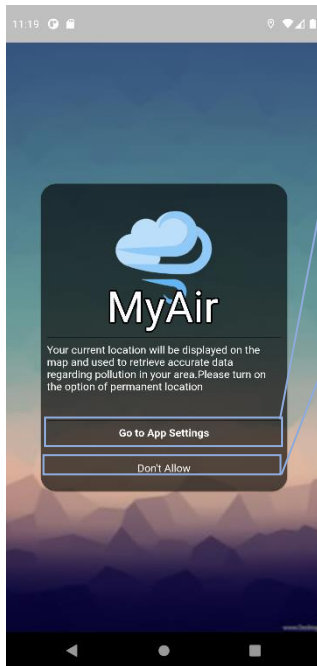


The registration page allows you to fill out a form to create your user account, in case you do not want to register via google account. Once the form has been completely filled out, just press the sign up button to be automatically registered in our firebase database which will check through queries that the email used to fill in the form has not already been used in advance.

LogoWidget: it is a Stateless widget which provide the logo of the application stacked with the name, encapsuled in a BorderedText provided by the bordered_text plugin.

RegistrationForm: registration form is a stateful widget which is used in order to authenticate and profile new users. It has the task of taking the information entered by the user, analyzing them and, if there are no errors, converting them into a format suitable for saving in the database. If, on the other hand, the email used by the user is already taken, the error message is displayed over the TextView of the email in the form



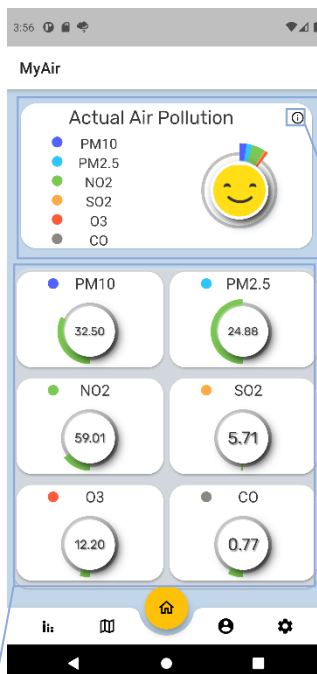
Permission screen:

The user accesses this page when changing the permissions settings for the app's geolocation. It will remain on this page until two events occur:

- Click the "Go to App settings" button which will redirect it to the system settings of the application and you will have to manually set the permissions for geolocation. When the permissions become "Allow always" it will be redirected to the correct page.
- Click the "Don't Allow" to deny the permission on the geolocation. This action leads to the closing of the application

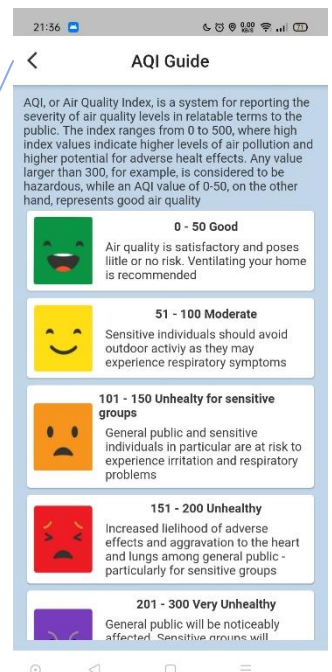
This page uses the Permission_handler plugin in order to open the app settings and check if the permissions are enabled.

Home Page screen: The home page is the opening page of our application. It is divided into various sections which contain the information on the data detected by the last measurement and displayed, first, in a global view that suggests how much the air is polluted as a whole and, only subsequently, focuses on the single pollutant.



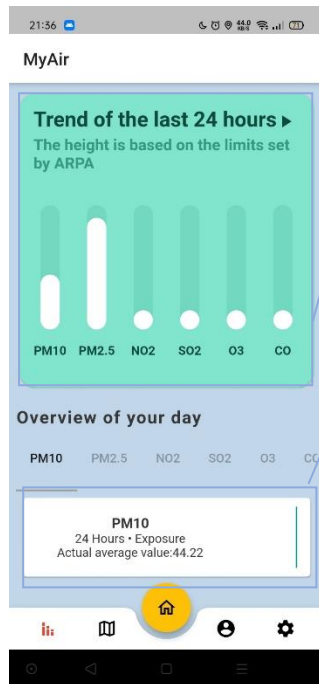
In this part are displayed a global view of the information taken from the last measurement. taken from the last measurement. As we can see, all the pollutants taken into consideration by the measure are listed and are put in the form of a pie diagram whose total perimeter is the sum of the permitted limits of the agents established by the ARPA.

Information Page: By clicking this icon you will be routed to a pop-up page (Shown on the left side) with all the reference values enclosed to understand the meaning of the image within the graph and the level of severity of air pollution. This AQI guide derives directly from the ARPA page.



AgentInfoWidget: this widget contains the last value detected by the sensor closest to the user. The green part is the value proportionate to the limit value established by the ARPA. (Values change in Real-Time thanks to ValueEventListener).

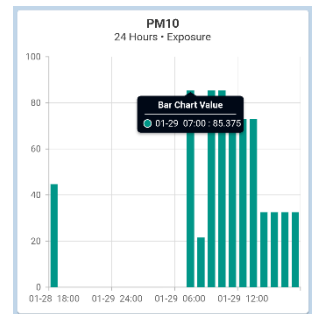
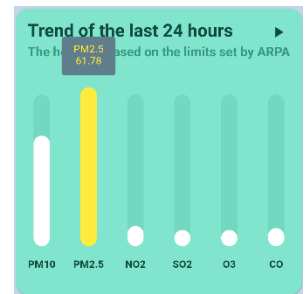
Chart Page screen:



This is the statistics page where the values collected in the last 24 hours are shown in the form of bar graphs and with an animated trend of values. Values are readable by pressing on the bar of the bar charts in order to make them more understandable by the user.

AnimatedChart: is a Stateful Widget which generates an animation with the values of the last 24h averages. User can either start or pause the animation and watch the intermediate values.

ChartCardWidget: is a Stateful Widget with two modes: the first one in which it gives a summarized view of the data, concerning a specific pollutant, collected in the last 24 hours. It also goes to show the current value of the agent. The graph on the right is a preview of the graph that will be displayed when you click on the container. When the user clicks on the box, the graph will expand into a real bar chart which will show the trend of the last 24 hours, compared to the current one, and each bar represents a specific time with the average of the values collected in this hour.



Map Page screen:



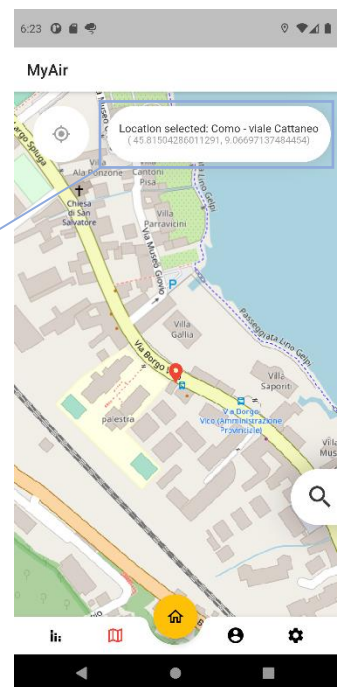
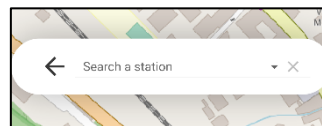
On this page, the user has the possibility to view all the sensors and their relative positions. Navigation can be done in free mode by selecting a pin on the map, or by searching for a specific sensor with the search widget

Center Position: the user set the position of the map in his/her actual position.

StationInfoWidget

This Stateless Widget starts with a visibility null, it is shown when the user taps a pin in the map or chooses a station in the search bar.

SearchWidget: is a Stateful Widget used by the user to find a specific station/sensor in his/her city. The user has to press and drag the icon in order to take out the search bar. (The layout of the search bar is shown below)



Reward Page screen:



This page has been designed to test the user with daily / weekly challenges aimed at keeping him in non-polluted areas and at the same time using the application constantly.

- **Daily challenges:** These challenges have been specifically designed to self-increase challenge earnings in proportion to the quality of each pollutant in the user's area.

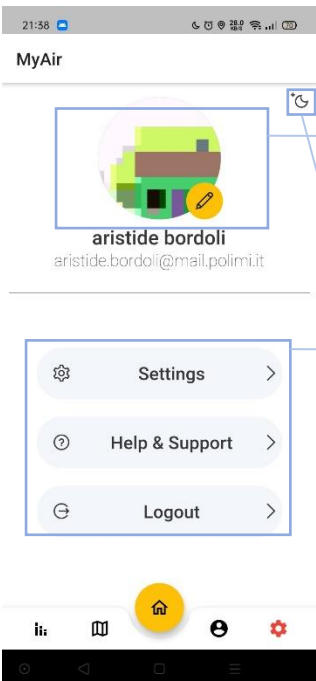
- **Weekly challenges:**

My Air is better: is studied in order to auto-increment based on the quality index gain of the area (the quality index is computed by the PollutantAgent Module which gives a reward between 0 and 1 based on the AQI value).

Daily Access: trivially is the login in the app in a row
Don't touch the bottom: is a progressive challenge that increments day by day and it is only completed if the user has never entered an area with AQI at the Hazardous level at the weekend (see the info page in the home page infos).



Account Page screen:

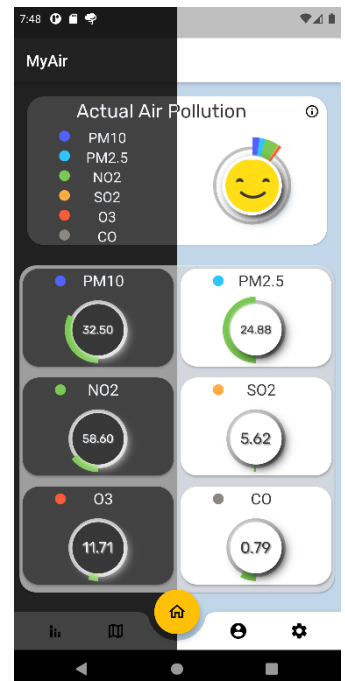
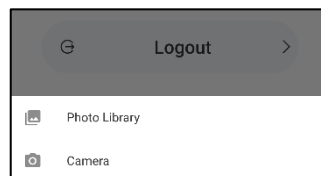


This page displays the user's personal data, his email and the default image that can be changed by pressing the icon below.

- **Change Image:** is a stateful Widget that allows the user to change the default image with one taken by the gallery or by taking a picture.

- **ThemeSwitcher:** is a stateful widget located at the top right and it is used to change the theme of the app from light to dark or viceversa (a demonstration is provided on the right).

- **ProfileListItem:** are Stateless widgets which can assume different behaviours. By clicking the first and second buttons they will be opened the settings and Help pages. By clicking the logout button the user will be redirected directly to the Profile screen.



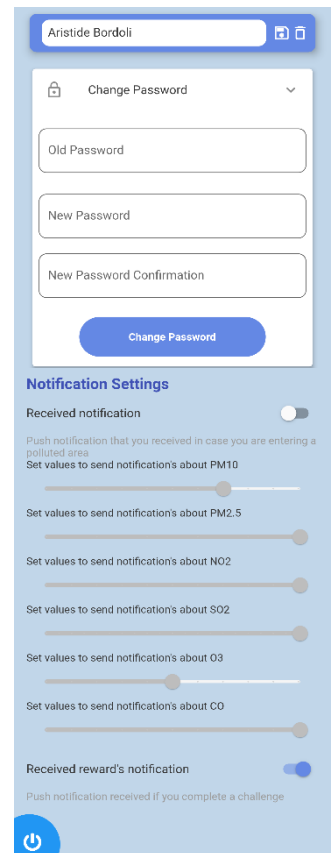
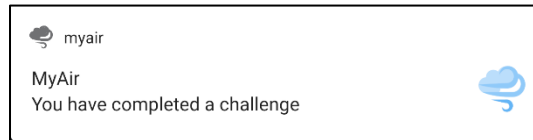
Settings Page screen:



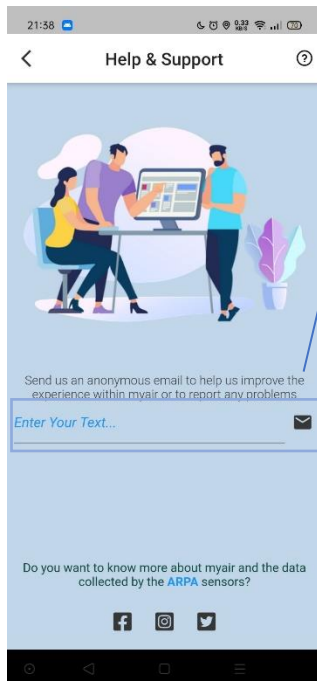
This page contains all the constructs for modifying user data. With the **nameChanger** Stateful Widget the user can change its name with a firstname lastname format and confirm with the safe icon or block the modification with the delete.

PasswordChanger: the expansion of this Stateful widget gives the possibility to the user to change the password by entering the old one first and then the new one twice.

The **notification settings** section gives the possibility to customize (with the **SlideAgent**) the limits in which the notification is throw by the application (Each time a user enters in an area with a PM10 over the limit setted by the user, a push notification is thrown) and also send a **notification** when a challenge is completed.



Help and Support screen:



This page was created with the intent to receive user feedback regarding the quality of the app and direct them to our official channels. Also, if interested, they can visit the ARPA page to learn more about the AQI.

SendMailFromLocalHost: it sends a mail with a pre-configured host to our official mail address. (Mail are sent thanks to the Mailer plugin)

- ☐ ☆ shadowmyairuser1 Help&Support Comment in 2021-01-21 20:04:06.648385 - Comment:
- ☐ ☆ shadowmyairuser1 Help&Support Comment in 2021-01-21 20:02:34.300008 - Comment:



RemoteView Widget:

An Android Home Widget that the user can place in order to know the actual values of the AQI and some of the pollutant agents.

Its update is managed by the RemoteViewUpdate service which has been developed in order to communicate between Flutter and Android.

3 UNIT, WIDGET AND SYSTEM TEST PLAN

A. Unit and Widget test plan

Unit and widget testing is applied to the program code, which is developed in accordance with the DD (Design document) as applicable.

Unit and widget test	
Tools and methods required for the implementation testing	<ul style="list-style-type: none"> - White box tests - Unit and widget testing
When to execute implementation testing	The implementation testing is executed as soon as the code programming of a software module is completed and before the software module is released for integration test.
Who execute implementation testing	<p>The implementation testing is executed and documented by qualified developers who verify the correctness and completeness, fix identified issues and document successful run.</p> <p>The review is performed by the project/quality manager.</p>
What is tested	<p>The implementation testing covers the modules as defined in the Design document.</p> <p>The following developed classes are tested:</p> <ul style="list-style-type: none"> - Sensor - SensorData - SensorListData - DailySensorData - PollutantAgent
Test of specified boundaries and invalid inputs	<ul style="list-style-type: none"> - Boundary tests are required for complex and critical calculations. - Boundary tests are required for interface data inputs. - Verify that the module is set to a defined and safe status if non-expected values/characters are inserted.
How to execute the test	<ul style="list-style-type: none"> - Collate the design specification and the developed code. - Verify all specified results by at least one test case. - Fix all identified deviations and re-execute the test until the module works as specified. - The last successful test run and code review must be documented.

B. System test plan

System testing should demonstrate that the system runs as required by the customer under conditions similar to production. The test follows the typical application process flow using pre-approved system test specifications.

System testing covers:

Verification of the system functionality to demonstrate fitness for purpose as defined by the Design document.

System test	
Tools and methods required for the system testing	<p>Test of functions under normal conditions.</p> <p>The system functions are tested in a test environment similar to the operational environment with near practical data or actual operating data. Simulation is used for important interfaces to external systems as applicable.</p> <p>Performance test in order to identify inefficiency related to algorithms, database queries, hardware/network issues.</p> <p>Load testing where applicable.</p> <p>Stress testing where applicable.</p>
When to execute system testing	<p>System test plan can be executed as soon as the system is installed in the testing environment as close as possible to the production environment. An installation qualification can be performed before the execution of the system test.</p> <p>The system test plan can be executed on a complete integrated system.</p>
Who execute system testing	<p>The system testing is executed and documented by an independent team who verify the correctness and completeness and document successful/unsuccessful run.</p> <p>The review is performed by the project/quality manager.</p>
What is tested	<p>The system testing covers the functions as defined in the Design document.</p>
Test of specified boundaries and invalid inputs	<p>Boundary tests and Negative test (abnormal conditions).</p> <p>Test of potential technical risks.</p>
How to execute the test	<p>System test plan shall include testing of functions as defined in the Design document.</p> <p>All tests are executed as specified in this test plan.</p>

D. Unit & Widget Tests

Step No.	Test Steps	Expected Result(s) / Acceptance Criteria	Result ok ?
1.1	<p>(DailyUnitData/actualAverage)</p> <p>Name: Test 01</p> <p>Insert of same value for the 24h-array</p> <p>This test is going to verify the average of the values related to the 24h-array of a sensor.</p> <p>The array is filled with the same value for all the 24 hours simulating the reading of the same value from the sensor</p>	<p>As soon as the array contains the same value, the result of the average must be the value inserted in the array.</p>	Tests passed: 1
1.2	<p>(DailyUnitData/actualAverage)</p> <p>Name: Test 02</p> <p>Insert of different value only for the position of the hour of the test.</p> <p>This test is going to verify the average of the values related to the 24h-array of a sensor.</p> <p>The array is filled with the same value for all the 23 hours simulating the reading of the same value from the sensor.</p> <p>In the cell of the array related to the same hour of the test a different value is inserted.</p> <p>Before the run the 'if' condition inside the 'for' and the timestamp of the sensor data must be changed with the hour of the test (0-23 h) as indicated in the two comments reported in the test</p>	<p>If the value in the 23h position is 0.84 and the value in the hour of the test position is 0.78 the result will be:</p> $(0.84 * 3 + 0.84 * 22 + 0.78 * 3) / (24 + 2 + 2) = 0,833571$	Tests passed: 1
1.3	<p>(DailySensorData/setDataAverage)</p> <p>Name: Test 03</p> <p>The same value is inserted in the same cell of the 24h position array in order to verify the calculation of the average.</p> <p>The cell selected is the one related to the hour of the running of the test.</p>	<p>The value 0.78 is inserted in the same cell 10 times and every time the average is calculated.</p> <p>The expected result will be 0.78</p>	Tests passed: 1
1.4	<p>(DailySensorData/getSum)</p> <p>Name: Test 04</p> <p>The same value is inserted in the 24 cells of the 24h position array in order to verify the sum of all of them.</p>	<p>The value 0.78 is inserted for ten times every time incrementing the value of 0.1 in 10 different positions of the array.</p> <p>The sum is calculated and the expected value is 39.6</p>	Tests passed: 1

1.5	<p>(SensorListData/getSensorListClosedToUsers)</p> <p>Name: Test 05</p> <p>The objective of this test is to verify that all the sensors within an area closed to the user are included in the list of the selected sensors used for the retrieving of the further data.</p>	<p>Six sensors have been inserted in the list of the installed sensors.</p> <p>The distance of all of them with the user is lower than 100Km.</p> <p>The function is called passing the coverage of an area within 100Km.</p> <p>The expected number of sensors selected is 6.</p>	Tests passed: 1
1.6	<p>(SensorListData/getSensorListClosedToUsers)</p> <p>Name: Test 06</p> <p>The objective of this test is to verify that all the sensors within an area closed to the user are included in the list of the selected sensors used for the retrieving of the further data.</p>	<p>Six sensors have been inserted in the list of the installed sensors.</p> <p>The distance of all of them with the user is lower than 100Km.</p> <p>The function is called passing the coverage of an area within 10Km.</p> <p>The expected number of sensors selected is 6.</p>	Tests passed: 1
1.7	<p>(PollutantAgent/setValues)</p> <p>Name: Test 07</p> <p>According to the values retrieved for each sensor a normalized value is calculated in order to manage the rewarding and related notifications.</p>	<p>For each sensor a selected value has been inserted in order to get the below expected data:</p> <p>pm10 = 1.00 pm25 = 0.8 no2 = 0.6 so2 = 0.4 o3 = 0.2 co = 0.2</p>	Tests passed: 1
1.8	<p>(PollutantAgent/setValues)</p> <p>Name: Test 08</p> <p>According to the values retrieved for each sensor a normalized value is calculated in order to manage the rewarding and related notifications.</p>	<p>For each sensor two selected values have been inserted in order to get the below expected average:</p> <p>pm10 = 1.00 pm25 = 0.8 no2 = 0.6 so2 = 0.4 o3 = 0.2 co = 0.2</p>	Tests passed: 1
1.9	<p>(PollutantAgent/setValues)</p> <p>Name: Test 09</p> <p>According to the values retrieved for each sensor a normalized value is calculated in order to manage the rewarding and related notifications.</p>	<p>For each sensor two selected values have been inserted in order to get the below expected average:</p> <p>pm10 = 0.6 pm25 = 0.8 no2 = 0.7 so2 = 0.4 o3 = 0.5 co = 0.2</p>	Tests passed: 1

1.10	<p>(PollutantAgent/setValues)</p> <p>Name: Test 10</p> <p>According to the values retrieved for each sensor a normalized value is calculated in order to manage the rewarding and related notifications. As soon as the day changes, a backup of the previous values is performed</p>	<p>For each sensor two selected values have been inserted and in the second set of data the day is different in order to get the below expected average:</p> <p>pm10 = 0.2 pm25 = 0.8 no2 = 0.8 so2 = 0.4 o3 = 0.8 co = 0.2 pm10_bck = 0.2 pm25_bck = 0.8 no2_bck = 0.8 so2_bck = 0.4 o3_bck = 0.8 co_bck = 0.2</p>	Tests passed: 1
1.11	<p>(PollutantAgent/setSensorsRW)</p> <p>Name: Test 11</p> <p>Reward Each time the system receive the values coming from the sensors, the correspondent normalized value is added to the previous one.</p>	<p>For each sensor two selected values have been inserted and in the second set of data the day is different in order to get the below expected average:</p> <p>pm10_rw = 1.00 pm25_rw = 1.6 no2_rw = 1.2 so2_rw = 0.8 o3_rw = 0.4 co_rw = 0.4</p>	Tests passed: 1
1.12	<p>(PollutantAgent/setSensorsRW)</p> <p>Name: Test 12</p> <p>Reward Each time the system receive the values coming from the sensors, the correspondent normalized value is added to the previous one even if the hour has changed.</p>	<p>For each sensor two selected values have been inserted and in the second set of data the day is different in order to get the below expected average:</p> <p>pm10_rw = 1.00 pm25_rw = 1.60 no2_rw = 1.30 so2_rw = 0.80 o3_rw = 0.70 co_rw = 0.40</p>	Tests passed: 1
1.13	<p>(PollutantAgent/get_pm10_notify)</p> <p>Name: Test 13</p> <p>Reward As soon as a cycle related a sensor is completed a notification is sent to the user as an alert</p>	<p>The limit for the pm10 has been set to 5. For each sensor 5 selected values have been inserted in order to reach the limit for the reward notification:</p> <p>pm10_rw = 5.00 pm10_notify = true</p>	Tests passed: 1
1.14	<p>(PollutantAgent/get_pm10_notify)</p> <p>Name: Test 14</p> <p>Reward As soon as a cycle related a sensor is completed a notification is sent to the user as an alert</p>	<p>The limit for the pm10 has been set to 5. For each sensor 5 selected values have been inserted in order to reach the limit for the reward notification:</p> <p>pm10_rw = 4.00 pm10_notify = false</p>	Tests passed: 1

1.15	(Reward page) Name: Test 15 Widget test The presence of 9 widgets of type ActiveReward is checked	9 ActiveReward widget is present in the Reward page	Tests passed: 1
1.16	(Settings page) Name: Test 16 Widget test The presence of 6 widgets of type SliderAgent is checked	6 SliderAgent widget is present in the Settings page	Tests passed: 1
1.17	(Chart page) Name: Test 17 Widget test The presence of 1 widget of type AnimatedChart is checked	1 AnimatedChart widget is present in the Chart page	Tests passed: 1
1.18	(Home statistics page) Name: Test 18 Widget test The presence of 1 widget of type GridAgentWidget is checked	1 GridAgentWidget widget is present in the Home statistics page	Tests passed: 1
1.19	(Home statistics page) Name: Test 19 Widget test The presence of 6 widget of type AgentInfoWidget is checked	6 AgentInfoWidget widget is present in the Home statistics page	Tests passed: 1

E. System Tests

Step No.	Test Steps	Expected Result(s) / Acceptance Criteria	Result ok ?
1.1	<p>Perform the login in the application creating a new user.</p> <p>Logout and logon to the application using the user previously created</p>	<p>The login is successfully done and a new user aristide.bordoli@mail.polimi.it has been created with the password 'test01'</p> <p>The user is visible in the user screen.</p>	OK
1.2	Select the Statistics in the Home page and verify the actual data coming from the six sensors configured are correctly displayed according the position of the user	<p>The data are correctly displayed in the statistics page and the color of the shadow is according the limits set in the application.</p> <p>The smile in the Actual Air Pollution area is display according the quality of the parameter retrieved.</p>	OK
1.3	In the statistics page click on the Information widget in order to have the description of the levels of the quality for each type of sensor configured	The Information page is correctly displayed	OK
1.4	<p>Select the chart page.</p> <p>In the upper part click on the run button and verify each sensor shows the 24h values for each of them.</p> <p>In the lower part the user can select each sensor and show the related 24h values using an histogram</p>	The sensor data are displayed according to the rules described aside	OK
1.5	<p>Select the map page.</p> <p>Verify the map is centered in the position of the user.</p> <p>Eventually the sensors are displayed thought pins. If you select a pin the related coordinates are shown.</p> <p>You can move around the map and zoom.</p> <p>You can press the symbol in the left top corner in order to go back to the position of the user.</p>	The gesture are according the description aside	OK

1.6	Open the search button and insert a location.	The pin of the related location is displayed on the map and the latitude and longitude are correctly displayed	OK
1.7	Click on the position object in order to come back to the current location	The current location is displayed on the map	OK
1.8	Selecting the reward page from the home the user is able to display the status of the rewards related to each sensor, the reward related to be almost in a clean area, the reward related to the number of daily access and the reward to avoid highly polluted areas	The information listed aside are all correctly displayed	OK
1.9	Select the User Profile from the Home page and verify the current user is logged in. Click on the theme switcher in order to change the background/foreground colors of the application	The current user is displayed and the theme switcher works correctly changing the default colors.	OK
1.10	From the user profile page select the settings page. Verify the possibility to change the own password, change the limits for the reward displaying and notification, the possibility to enable/disable notification related to reaching the sensor values limits and the possibility to enable/disable the notifications related to the rewarding	The settings described aside are working correctly	OK
1.11	From the user profile page select the Help & Support page. Verify the possibility to send a mail and to access to the ARPA web site	The user can enter a text sending a mail and can access to the ARPA web site	OK
1.12	From the use profile page the user can logout from the application	The user logout from the application and lands in the log in page	OK