Technical Project Report | Flutter Module

G03 | “WhatzUp”

Subject:	Mobile Computing
Date:	January 8th, 2024
Students:	João Simões (88930) Pedro Duarte (97673)
Project abstract:	WhatzUp is a social media app inspired on WhatsApp but specific to connect people at social events.

Report contents:

1 | Concept and motivation

WhatzUp is the platform where one will find information about events and people taking part in them, guiding you from discovering what’s happening near you to practical networking through real-time chats, status/stories updates and voice/video calls, available during the event.

One will be able to connect with all people around without relying on an internet connection, since the app takes advantage of the peer-to-peer architecture of the Nearby APIs exposed by modern mobile devices, abstracting multiple technologies such as Bluetooth, BLE, and Wi-Fi.

2 | Features and requirements

In order for the application’s concept to be validated, some of the identified user stories were implemented, which will be described in this section.

First of all, the app has a page listing all the events happening. On this “Events” tab, the user is able to see an image, the name and a short description of each event and is given the ability to do a simple search filtering using these last two fields.

For each event, two actions are made available - adding the event to the calendar; and seeing more details on a new page, opening from a hero animation. The extra information includes the date, ticket price, and a map where the user can see his and the event’s position.

On the “Chats” tab, the user is able to see his current nearby chats and his chat history, in an interface resembling the commonly used WhatsApp. The difference in this page is that, instead of having the ability to start new chats from phonebook contacts, users are given the ability to find and directly connect to nearby devices. The user can wait for another user to connect with him or the user himself can take the initiative and look for nearby devices. After connecting to other users/devices, they now have access to the real-time chat functionality if taping on that person. The messages received and sent are displayed on the “Chat” page and every time a new message is detected, a popup notification is shown to warn the user to reply. There are text messages but also image messages. We tried to implement audio messages too, but there was a problem with the audioplayers package conflicting with others. It’s possible to delete old chats by long pressing on the ListView widget, which opens an alert dialog to confirm (or cancel) the action, as a precaution. Likewise, in this tab, user can search for usernames and messages to filter the chat entries.

On the “Status” tab, the user can find status updates with photos, videos or only text from their contacts in a fashion made popular by Instagram’s “stories”. They can also create a new status update (aka story) by pressing the FAB (floating action button) with a camera icon. Once again, the user can also search by username to filter the status updates.

Moving on to the “Calls” tab, the user can see a history of calls (video and voice calls) made or received and can even start another one (demo only), by pressing on that person. And after that, the user is redirected to the “Call” page, in which appears the person name, phone number, call duration (implemented with a timer), and action buttons to end and control the call. A silent notification also appears informing the user that he is on an ongoing call with the respective other person and displaying the call duration. Still on the “Calls” page, there is a FAB to quickly open the phone dialer in case of emergency to call 911 or an emergency contact. Once more, the user can also search by username to filter the history of calls.

On the top right corner, an icon with three vertical dots hides two more pages. The first one, “Profile”, lets the user edit his profile picture (by picking images from the image library, taking new pictures with the camera or simply choosing the default avatar), his display name, and his short biography or mood. It also displays the phone number to make sure that the account is unique, and because of that, this field cannot be edited, since it should always be the device’s phone number.

The other one is the “Settings” page, where the user is able to:

- change the language of the app to English, Portuguese, Spanish, French, Italian, German, Russian, Chinese and Japanese;
- choose a different wallpaper for chat and call page background;
- enable or disable popup notifications;
- alternate between dark and light mode;
- ask for help and create support tickets via email;
- and, finally, see app and app developers details.

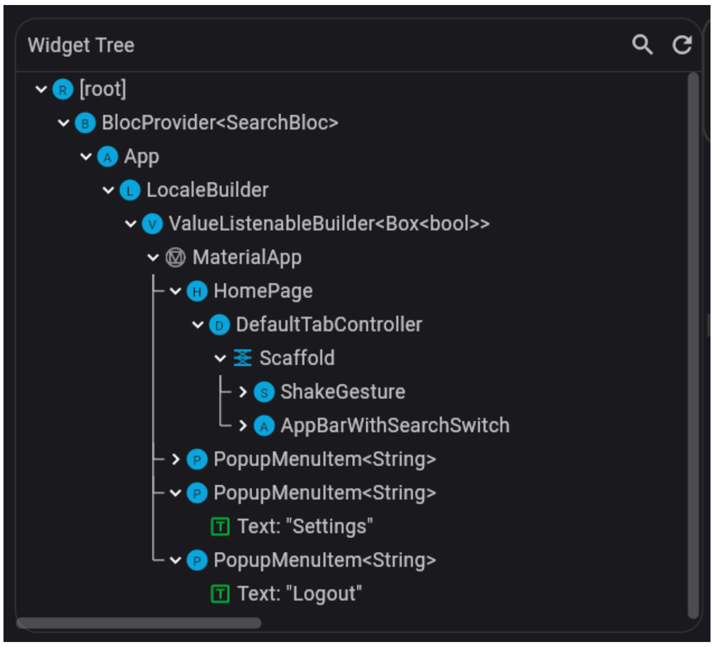
The last functionality made available to the user is a little bit secret: reporting bugs. This is a feature taken from Instagram and it’s a modal bottom sheet that pops up after the user physically shakes his device. The goal of this modal is a shortcut for users to easily report problems in the app, as they can make more abrupt movements with the phone when they’re stressed because something isn’t working properly in the app.

In summary, the specific technical features we implemented were:

1) **Camera** (take pictures for status/stories and for changing the profile avatar);
2) **File system** (upload images from mobile native gallery);
3) **External resources** (fetch dynamic list of status updates from external JSON APIs);
4) **Nearby connections** (find and chat with friends nearby);
5) **GPS** (get user location to show on map);
6) **Maps** (show the location of events and users);
7) **Sensors** (detect shake movements with accelerometer sensor values);
8) **Microphone** (perform speech to text recognition for hands-free voice search);
9) **Databases** (use local/Hive and cloud/Firebase databases to save user/app data);
10) **BLOC** (manage states to access the search query value between widgets/pages and listen for changes in its value to trigger search mode on every tab).

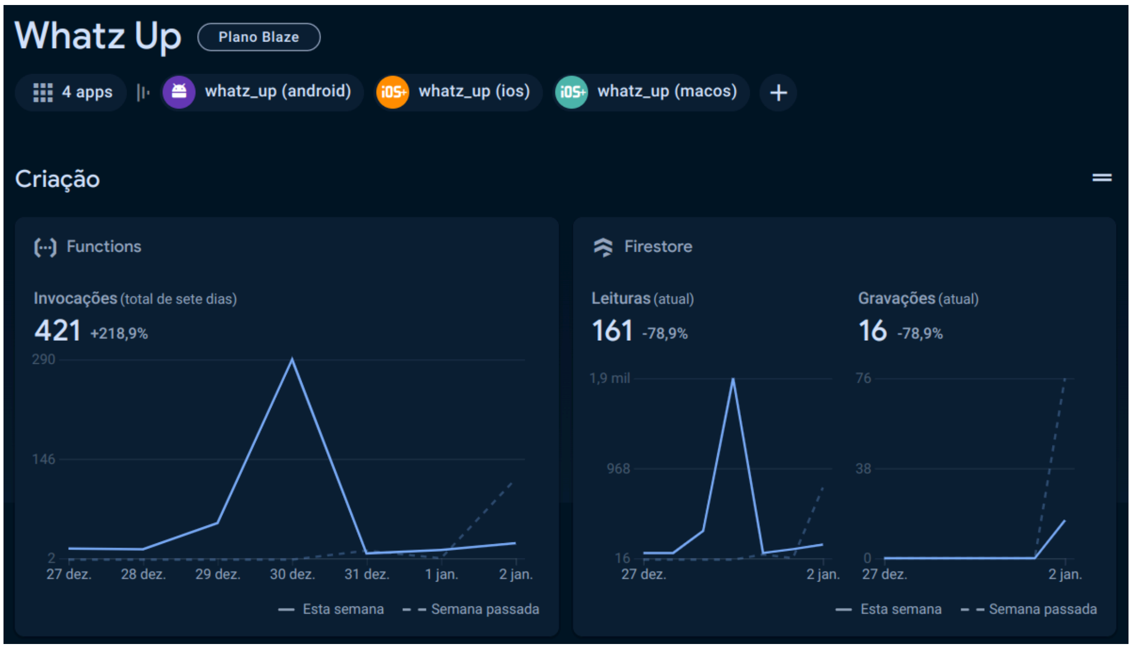
3 | Architecture design

The base Widget tree is composed of a Bloc provider for the text search filtering feature, a screen-transversing Scaffold comprising the bug reporting activated by the device shaking, and a top bar with search input and functionality icons.

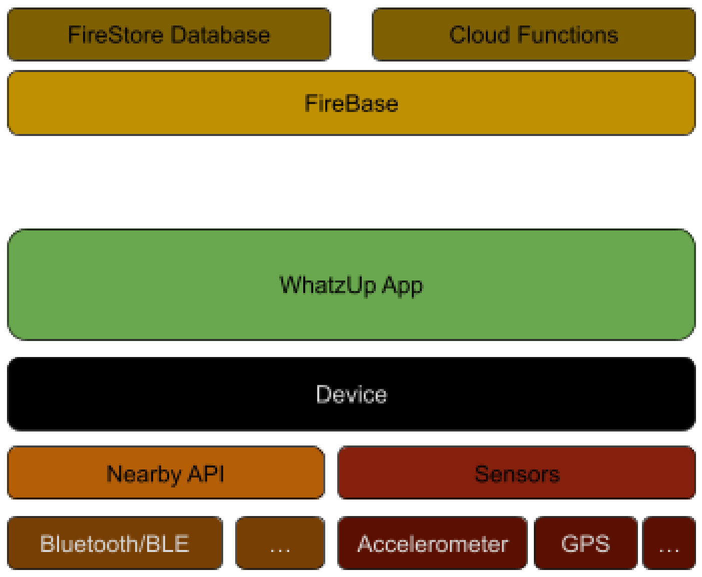


```
Widget Tree
[root]
  BlocProvider<SearchBloc>
    App
      LocaleBuilder
        ValueListenableBuilder<Box<bool>>
          MaterialApp
            HomePage
              DefaultTabController
                Scaffold
                  ShakeGesture
                  AppBarWithSearchSwitch
            PopupMenuItem<String>
            PopupMenuItem<String>
              Text: "Settings"
            PopupMenuItem<String>
              Text: "Logout"
```

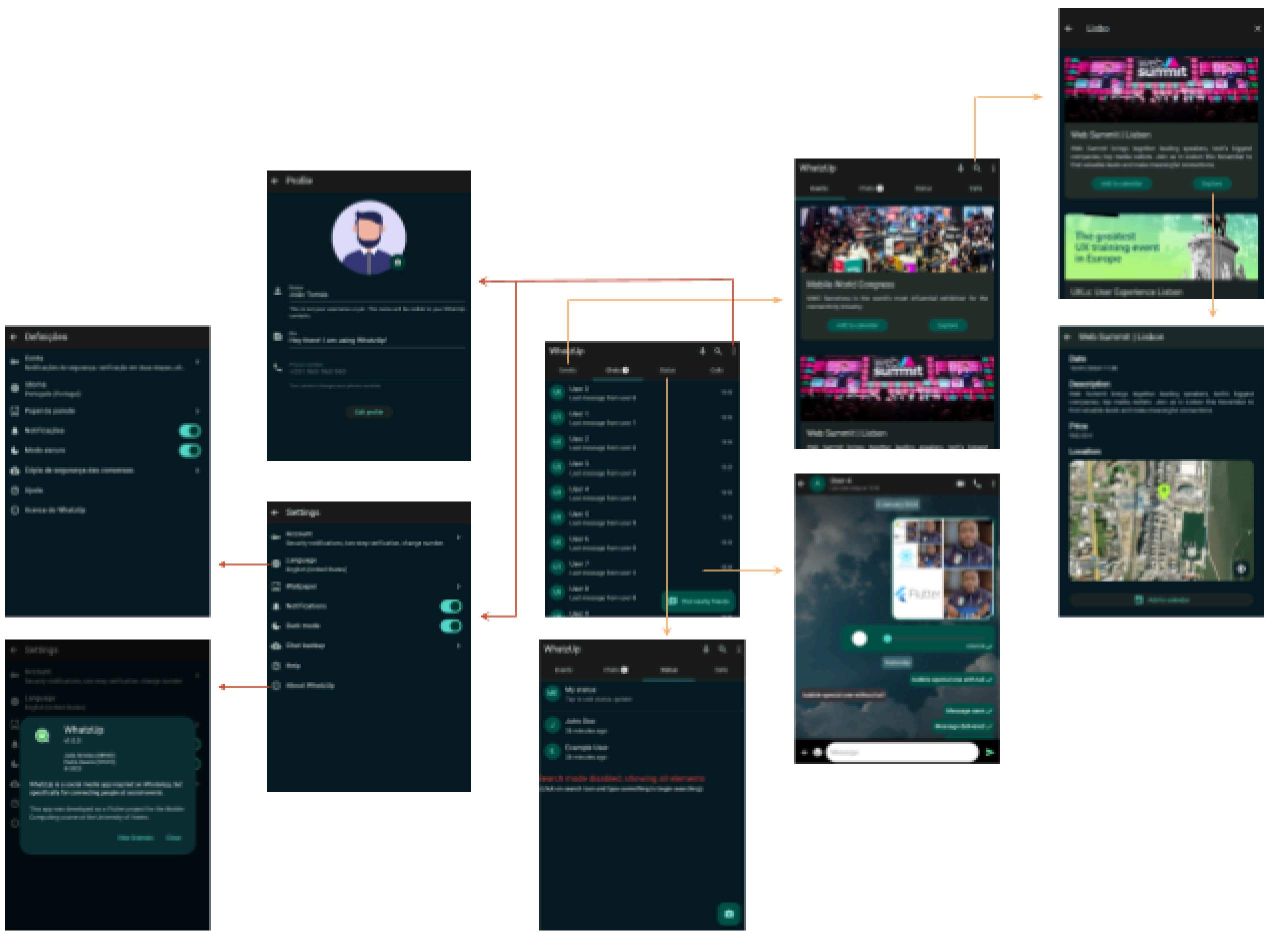
As shown, the app uses Bloc for state management and also the Hive database for storing data locally, offline. Firebase is used by providing a layer of cloud functions that exposes an API for the events, backed by its FireStore database.



In summary, the following diagram shows the most relevant architecture components around the application.



4 | Application interaction flow



The app starts at the home page which opens with the “Chats” tab by default (the focus of this app). In this screen, they can see details of each user and the last message on the chat, as well as the time that message was sent. Clicking on any chat will lead to the Chat page where the user can see all the messages exchanged in the past as well as sending a new one.

Through the TabBar, a user is able to navigate to the “Events” page, where they can find all the events registered in the platform, as well as filter them by name or description, through the same process as described before. Each event card has two buttons, allowing the user to add the event to the calendar or open the respective individual “Event” page. On that page, they can see all the details of the event - image, name, description, price, date, and location - as well as having, once more, a button to add the event to the calendar.

Also through the TabBar, the user can access a prototype for the “Status” page where the UI for seeing, creating, and sharing status updates in the form of image, video or text, similar to Instagram’s stories.

As already described in early sections, the TopBar has an icon with three horizontal dots where the user can find the options to edit his profile details and manage app settings and preferences.

5 | Conclusions

Given the state of implementation, we strongly believe that our main goal, giving the user the full experience from finding an interesting event to practical networking with other users, was a success. Through the app, we can, at the present time, browse different mock events, connect to different devices through Nearby APIs, and chat with the other attendees of the event.

The main challenge in this project was working with Nearby APIs. Despite the similar names, Android’s Nearby and Apple’s Nearby Interaction are completely distinct APIs and there is no transparent solution to connect users from different platforms. For this project, we decided to target only Android devices, as problems were being detected in relation to the iOS ecosystem, with regard to special permissions and some Android blockages.

Despite the adjustments we had to make along the way, we consider this experience very positive for our individual curriculums, giving us direct insights into app development in Flutter and, specifically, to the Nearby technology in Android, a technology that no member of the group had yet explored.

6 | Limitations and future work

First of all, there are some minor/extra features in this app that would need some extra work, such as real video and voice calls, real login methods, the connection of the app to the phone number, and implementing the functionality of some of the settings.

As mentioned above, one of the main limitations of this work was connecting users of different platforms on the app. As future work, we would most likely need to replace the Nearby API with a different technology, such as pure Bluetooth to keep the device-to-device communication or replace this idea altogether and require events to provide infrastructure, such as a simple router, even without connection to the internet, to allow communications to occur via LAN.

Also as future work, we think it would be interesting to allow users to buy tickets to events in-app, implementing payment services and ticket management. With this feature, we would allow our users to have all the event experience without ever leaving the app.

To further extend the interactivity inside the app, we could also create a separate app for the event holders to see who is there, see statistics of the event, and even allow direct communication between users and event holders.

7 | Author contributions

Taking into consideration the overall development of the project, the contribution of each team member was similar and therefore evenly distributed: João Simões (88930) did 50% of the work, and Pedro Duarte (97673) contributed also with 50%.

8 | Project resources

Resource:	**Available at:**
Code repository:	github.com/jtsimoes/CM/tree/main/Flutter/whatz_up
Ready-to-deploy APK:	github.com/jtsimoes/CM/releases/latest/download/whatz_up.apk
Minimum requirements:	Android 6.0 (Marshmallow) with 70 MB of free storage
Permissions required: (all optional but recommended)	Internet (WiFi or mobile data), Bluetooth, Location/GPS (fine and coarse), Calendar (read and write), External storage (read and write)

9 | Packages used

- add_2_calendar - adds events to user calendar default app;
- app_bar_with_search_switch - improves the native AppBar, turning it into a toggleable search field for events by typing or speaking;

- cached_network_image - keeps external images from the internet on cache instead of getting them from the Internet every single time;
- chat_bubbles - implements chat bubbles similar to the WhatsApp for nearby chats;
- cloud_functions - allows using Firebase Cloud Functions in Flutter;
- firebase_core - allows the use of Firebase Core API for fetching events;
- flutter_bloc - implements the BLoC design pattern, useful to manage app states;
- flutter_launcher_icons - simplifies the task of generating Flutter app’s launcher icon;
- flutter_local_notifications - displays local notifications on the user device for ongoing calls and new messages received;
- flutter_locales - manages the multiple locales/languages of the app;
- flutter_map - implements Leaflet maps in Flutter;
- flutter_map_location_marker - displays the current location of the user on a map;
- flutter_native_splash - adds nice splash screen while the app is loading;
- flutter_nearby_connections - supports peer-to-peer connectivity and discovers nearby devices for Android devices to chat between them;
- go_router - provides a convenient, url-based API for navigating between different screens and pass data through them;
- hive - serves as a lightweight and fast key-value database to save chats, profile details and settings preferences;
- hive_flutter - assists Hive’s main package with specific widgets to Flutter;
- http - allows HTTP requests to be made to obtain status updates from an external API;
- image_picker - makes possible to pick images from the image library and take new pictures with the camera to change avatar picture;
- latlong2 - performs latitude and longitude calculations (useful for event maps);
- shake - detects phone shakes with sensors to open report problem modal bottom sheet;
- story_view - displays stories just like WhatsApp and Instagram;
- url_launcher - launches `tel` (phone dialer) and `mailto` (email app) URL schemes;
- whatsapp_story_editor - adds story editing features similar to WhatsApp (texts, stickers, painting, crop, filters, etc.).

Note that, although these are all the dependencies imported into the `pubspec.yaml` file (direct dependencies), each of these packages can also have more sub-dependencies associated with it (transitive dependencies). For example, the “shake” package makes use of the “sensors_plus” package to obtain the accelerometer values and thus detect the shake movement.

10 | Reference materials

- Material Design 3 documentation - https://m3.material.io/develop/flutter
- Flutter API documentation - https://api.flutter.dev/
- Flutter packages - https://pub.dev/
- Flutter Gallery - https://gallery.flutter.dev/