Mobile and Ubiquitous Computing: PharmacIST Group - 10

Francisco Sousa
Instituto Superior Técnico

Miguel Porfírio
Instituto Superior Técnico

Sara Aguincha
Instituto Superior Técnico

1 Architecture and Technology

In this section, we address the choices made for the back-end and database components of our application. To ensure our server operates with a static IP address, we used Linode's cloud services. We provisioned a small virtual machine (VM) on Linode where both our back-end and database are hosted.

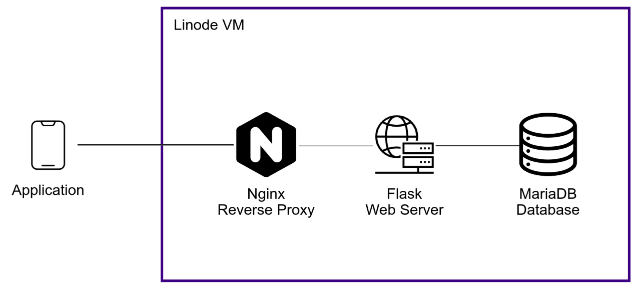


Figure 1: Schema of the system

1.1 Reverse Proxy

To ensure that the Virtual Machine was accessible outside of its network, a reverse proxy was used. This ensures anyone that accesses the public IP of the VM on port 443 (HTTPS) accesses the flask web-server. To configure this reverse proxy, we used Nginx, all of the configuration files and scripts that were used are given in the project's README.

1.2 Back-end

For our back-end, we chose the Flask framework, a lightweight Python web framework. Flask's simplicity and extensibility made it an ideal choice for developing our RESTful API.

The back-end is responsible for handling HTTP requests from the Android app, processing these requests, interacting with the database to perform necessary queries, and returning the appropriate responses. Flask's modular design allowed us to easily integrate various extensions to enhance functionality, such as security, request validation, and authentication mechanisms.

1.3 Database

We chose MariaDB for our database management system due to its robustness, performance, and open-source nature. Our database schema is designed to ensure data persistence and integrity, comprising several tables tailored to our application's requirements.

- **Users:** Stores user information such as user ID, username and password (hashed).
- **Pharmacies:** Contains information about pharmacies, including pharmacy ID, name, address, latitude and longitude.
- **Medicines:** Holds data about medicines, including medicine ID, name and description/purpose.

In addition to these primary tables, we have several relational tables that help manage the interactions between the primary entities.

2 Mandatory Features

This section focuses on how we implemented all of the mandatory features in our project.

2.1 Users

Users in PharmacIST are designed to offer a seamless and **persistent** experience across **all devices.** By creating a **username and password,** users can log into their account on multiple devices, ensuring that their data and preferences are **synchronized everywhere** they use the app. This includes favorite pharmacies, medicine information, and notification settings, providing a consistent and personalized experience.

For those who prefer not to create an account, there is also a guest login option. Even as a guest, users can still benefit from data persistence within the app. This means that if a guest user closes the app, their session data remains intact, allowing them to continue from where they left off when they reopen the app.

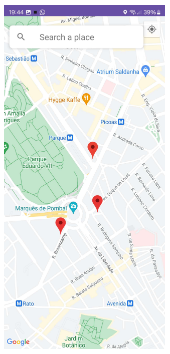
This flexibility ensures that the application is **accessible to all users**, regardless of their preference for account creation. The implementation details of user **authentication/login** and **guest access** will be discussed further in Section 3.2.

2.2 Maps

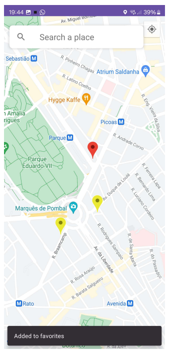
In the app, we integrated a map feature that displays pharmacy locations, providing a user-friendly way for users to find and interact with pharmacies. Key functionalities of the map include:

- **Interactive Map:** Users can drag the map, search for addresses, or center it on their current location. This allows for easy navigation and exploration of pharmacy locations.
- **Adding Pharmacies:** When adding a new pharmacy to the system, users can pick a location directly on the map, ensuring accurate geolocation data for each pharmacy.

To implement these features, we utilized the **Google Maps API**, which offers robust and reliable mapping capabilities. For address searching, we integrated the **Google Places API**, allowing users to enter **addresses or place names** and receive precise location results.



(a) Pharmacies



Added to favorites

(b) Favorite Pharmacies

Figure 2: Favorites are displayed with a yellow marker

2.3 Pharmacies

While still within the map feature, pharmacies have a **marker** that the users can tap to open a **pharmacy information panel**. This information panel allows users to see the pharmacy's **name, location and picture**, with a button to **navigate** to the pharmacy, which opens **Google Maps** and sets the pharmacy location as the destination. Users can add/remove the pharmacy from their **favorites** with the click of a button, which will reflect on the **color** of the marker shown on the map. Also, they can **share** this to other social apps, which we will talk about more in section 3.3.

The pharmacy information panel also includes a **search feature for medicines**, listing available ones and allowing users to tap searched medicines in order to open a **medicine information panel**. Alongside that, it features a way to **add medicine stock** by scanning a barcode, or create a medicine, by providing a name, box photo taken from camera or file, quantity and purpose. Additionally, it is also possible to **purchase medicine stock** by scanning a barcode.

And lastly, in the home screen, users can access a tab in order to add a **new pharmacy** to the system. This can be done by providing a name, taking a picture, and either picking the location on the map, searching an address using the **Places API** or using the current location of the user.

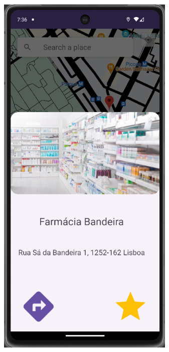
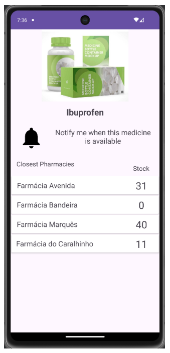


Figure 3: Pharmacy Drawer

2.4 Medicine

Regarding the **medicine information panel** mentioned previously, within it, users can view the **name and picture** of the medicine, and also a **list of pharmacies** that have it, sorted by distance. They can also tap on a button to get a **notification** of when that medicine is available in one of their favorite pharmacies.

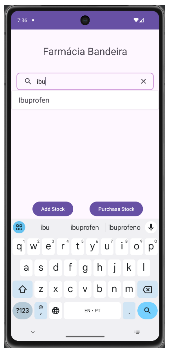
The app also includes a feature where users can **find medicines** by performing a sub-string search and tapping the medicine to take them to a panel showing the medicine information, the **closest pharmacy** that has it, a button to **navigate** to said pharmacy, a button to show more information on the medicine, taking them to the **medicine information panel**, and an option to **share** this information to other apps.


Ibuprofen

Notify me when this medicine is available

Closest Pharmacies	Stock
Farmácia Avenida	31
Farmácia Bandeira	0
Farmácia Marquês	40
Farmácia do Caralhinho	11

(a) Medicine Panel



(b) Medicine Search

Figure 4: Medicine Panel and Search

3 Additional Features

In addition to the mandatory requirements, we chose and added the additional following features to PharmacIST.

3.1 Securing Communication

PharmacIST provides secure communication by the use of **SSL certificates** and **TLS communication**. The web server is hosted here. By using a web domain, PharmacIST was able to receive an SSL certificate from the certification authority **Let's Encrypt**. This way, there is no need for a user to install any third party certificates as this certification authority is accepted in almost all devices.

The SSL certificate enables the use of TLS, which ensures **confidentiality** between the user and the web server, making this application robust against any snooping attacks and man in the middle attacks. This confidentiality aspect is essential for an application of this nature, as an attacker being able to know what medication a certain user is interested in, might expose some medical condition that the user had.

3.2 User Accounts

As previously mentioned in **section 2.1,** PharmacIST's user accounts synchronize data and preferences across devices, ensuring a consistent experience. Creating an account allows access to favorite pharmacies, medicine info, and notifications on any device. The guest login option also maintains session data, letting users continue from where they left off even after closing the app.

The app utilizes **token-based authorization** to streamline the login process. Upon logging in, an authorization token is **generated** in the back-end and securely **stored** on the device. This token is used for **automatic authentication** every time the app is opened, eliminating the need for manual login and enhancing convenience.

Additionally, for notifications, PharmacIST uses **Firebase Cloud Messaging (FCM).** Each device receives a **unique FCM** token upon first login. This ensures that notifications about favorite pharmacies and medicine availability are sent to **all devices** linked to the **same account,** keeping users informed and connected.

When a user logs out, guest users **lose all** their session data, including favorite pharmacies and notifications, as shared preferences such as (username and tokens) are **deleted.** Registered users only need to log in again to **restore access** to their synchronized data and preferences.

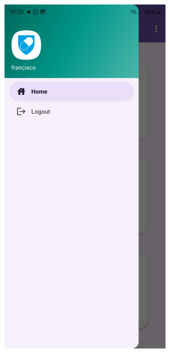


Figure 5: User Drawer

3.3 Social Sharing To Other Apps

Social sharing is available on PharmacIST in **two different locations**. When users are in a pharmacy information panel and want to share the pharmacy they just found, and when they are searching for the closest pharmacy that has a specific medicine and want to share their discovery. This is done by simply **clicking a button**, which brings up a **panel**, allowing the users to **choose which app** they want to share the information to. Upon clicking the app of their choice, this will immediately open that app and and paste the information to the corresponding field, ready to be **posted**.

As an example, if the user chooses to share it via e-mail, this will open the user's email application on the "New Message" feature, with the information already pasted in the body of the email, ready to be sent to whoever they want.

This is done through the use of **intents** and Android's own sharing panels.

3.4 Localization

In order for PharmacIST to reach the biggest audience possible, ensuring **localization** is essential. We chose to translate the app to **Portuguese**, as it's the language that we are most comfortable with, but further translations can be easily added and were taken into consideration when developing PharmacIST.

Every time there is any type of text displayed on the application, these strings are fetched from the *strings.xml* file, which ensures ease of manipulation for different languages, by the use of methods offered by Kotlin like *getString*.

3.5 UI Adaptability: Rotation

UI adaptability is key to ensure that PharmacIST works in a wide variety of devices. For example, most people use their tables in **landscape** mode, while phone are mostly used in **portrait** mode. To ensure a seamless transition between these devices, PharmacIST supports both portrait and landscape views.

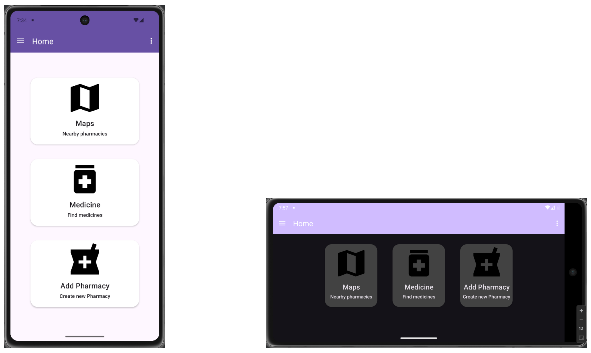
Every layout has both a portrait and landscape resource file with the layout carefully put together to accommodate both view, providing a more satisfying user experience.

3.6 UI Adaptability: Light/Dark Theme

Users may want to switch between **light and dark mode** when using their applications, so we felt this was an important feature to add to ours.

To do so, we took advantage of Android's **DataStore** technology, which is a **persistent** data storage solution that allows to store **key-value pairs** or typed objects with protocol buffers **asynchronously** and **consistently**. By storing the value of a "Dark Mode" key, we can persist its **state** through every application usage.

Upon application start, the value stored in the **DataStore** is checked to determine which theme the application should be using, both provided by Android and chosen by the user. Once inside the application, the user can click the settings icon and there, they will find a **switch** to **toggle between light and dark mode**, according to their preferences, which will change the value of the key stored in the **DataStore**. The app will then **maintain this state**, even through subsequent uses.



(a) Portrait (b) Landscape

Figure 6: Portrait and Landscape in both theme modes

4 Resource Frugality

In PharmacIST, pharmacy and medicine information is **synced** across all devices while trying to use the network **efficiently**. Information that has to be always **up-to-date** and synced, such as favorite pharmacies, is always fetched from the server, whereas information like pharmacies, medicines and their images are only fetched **from time to time** through the use of a **local cache** on the phone. When the user is actively viewing something, that content is assured to show up quickly.

Plus, when the user is searching for something, results only show after **three characters** are typed to **reduce network and server stress**, while also only applying filters on the server side to reduce data transmission.

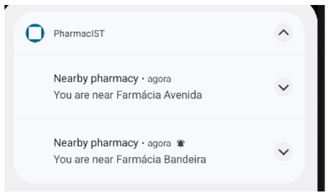
Pharmacy and medicine photos also have a **placeholder** for when the user is not able to download that information from the server in a timely manner.

The app allows users to get in **without** an internet connection and use some features such as viewing pharmacies on the map and their information, but a lot of features will be unavailable until they restore either a WiFi or mobile data connection. However, if the app is under maintenance (the server is down), users will not be allowed to enter the app during this period.

5 Context Awareness and Privacy

The PharmacIST application is **aware of its location** and as such, it includes a feature that allows the user to receive **notifications** on their phone when within range (200m) of a pharmacy. By clicking this notification, the user will be directed to a screen showing the map with the **marked pharmacy** that they selected. They can then click on the marker to open the **pharmacy information panel**.

This is done by having a **TimerTask** run in the background every few seconds, checking if the user location has changed **significantly**. If it has, a call is made to the server to fetch the pharmacies that are within range (200m) of the user. It then creates a **notification** for each of the pharmacies fetched.


PharmacIST

Nearby pharmacy · agora
You are near Farmácia Avenida

Nearby pharmacy · agora
You are near Farmácia Bandeira

Figure 7: Context Awareness notification

6 Caching

The application **avoids** downloading the same content **multiple times** in cases where that content would likely be repeated, such as downloading pharmacies multiple times every few seconds. As such, the PharmacIST app employs a **local cache** to store this information until a new fetch is necessary. This is particularly useful when dealing with larger downloads such as images and many pharmacies.

There is also a **cache pre-loading** feature, that by storing the nearest pharmacies images when a Wi-Fi connection is available, as Wi-Fi is usually not a metered network and downloading heavier files is not a problem. Having this feature, some of the most likely images that the user would fetch are already downloaded. This is important, as later on, if the user is using mobile data or doesn't have a internet connection at all they can still view some images.

6.1 Data saving mode

PharmacIST has an optional setting that activates a data saving mode. When this setting is activated, no images are fetched outside of a Wi-Fi connection, instead, if an image for a certain pharmacy or medicine is not present in cache, the application loads a default image that is stored locally. This way, the user never uses a considerable amount of mobile data, as most of the network usage in this application is used for fetching images.

7 Conclusion

Overall, we are very satisfied with our PharmacIST application as we felt we did everything we wanted by implementing every feature we set out to implement. We have developed every mandatory feature and the extras we found the most interesting and pertinent. The app displays all data to the user **quickly** and does so **without compromising the workload** and **networking** behind data downloading. We strived to find a good balance and we believe we have achieved it.

Performance wise we are also satisfied with the results. This being a mobile application, network usage matters and we feel like with the use of **cache** and **pre-fecthing** when in Wi-Fi we achieved our intended goals.