Mobile Developer at Empatica

Documentation

Gabriele Bressan

THECALENDAR

Empatica

April 2019

Contents

Contents **1**

1 Setup instructions **2**
1.1 GitHub Clone . . . 2
1.2 Android Studio . . . 2
1.3 Installation requirements . . . 3

2 TheCaregiver Overview **3**
2.1 User Experience . . . 4
2.1.1 TheCalendar Home Page . . . 4
2.1.2 TheCalendar Reservation Page . . . 4
2.1.3 TheCalendar Home Page with Placeholder . . . 5
2.1.4 TheCalendar Reservation Modification Page . . . 8
2.1.5 TheCalendar auto fill . . . 8
2.2 Implementation choices . . . 8
2.2.1 Auto Fill feature implementation . . . 9

3 Technologies used **10**

4 System architecture **12**

5 Design Pattern **13**
5.1 Adapter . . . 13
5.2 Observer pattern . . . 13
5.3 Singleton . . . 14
5.4 Model-View-Controller . . . 14
5.5 Client-Server . . . 15

6 Testing **15**
6.1 Logical components . . . 15
6.2 Graphical components . . . 15

7 Future improvements **16**
7.1 Improvements on current version . . . 16
7.2 Improvements on new version . . . 16

8 Used tools **16**

1 Setup instructions

1.1 GitHub Clone

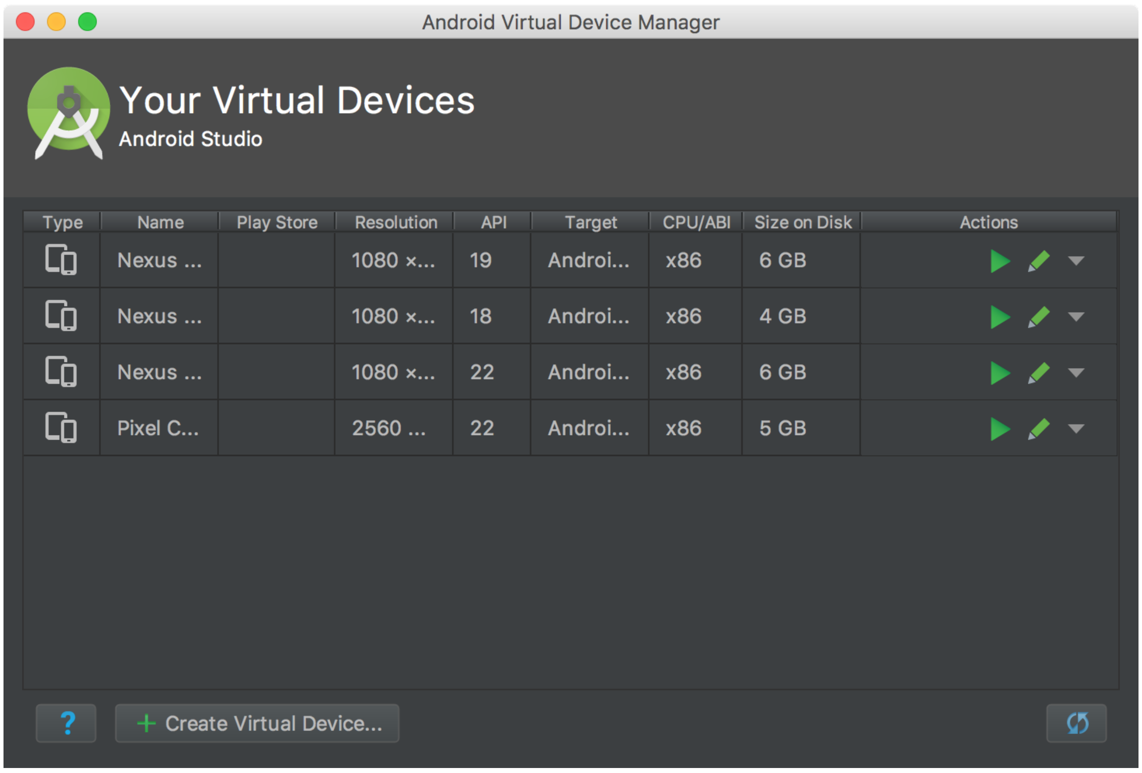
The TheCalendar project is available on the following GitHub Repo. First of all you have to clone the TheCalendar project on your PC so open your terminal, go to the folder you want and clone the repo by using the following command:

```
git clone https://github.com/Gabriele1606/TheCalendar
```

1.2 Android Studio

Once cloned the GitHub repo, open Android Studio and import the TheCalendar project by clicking on **File >Open...**, go into the directory of the project and select **Open**.

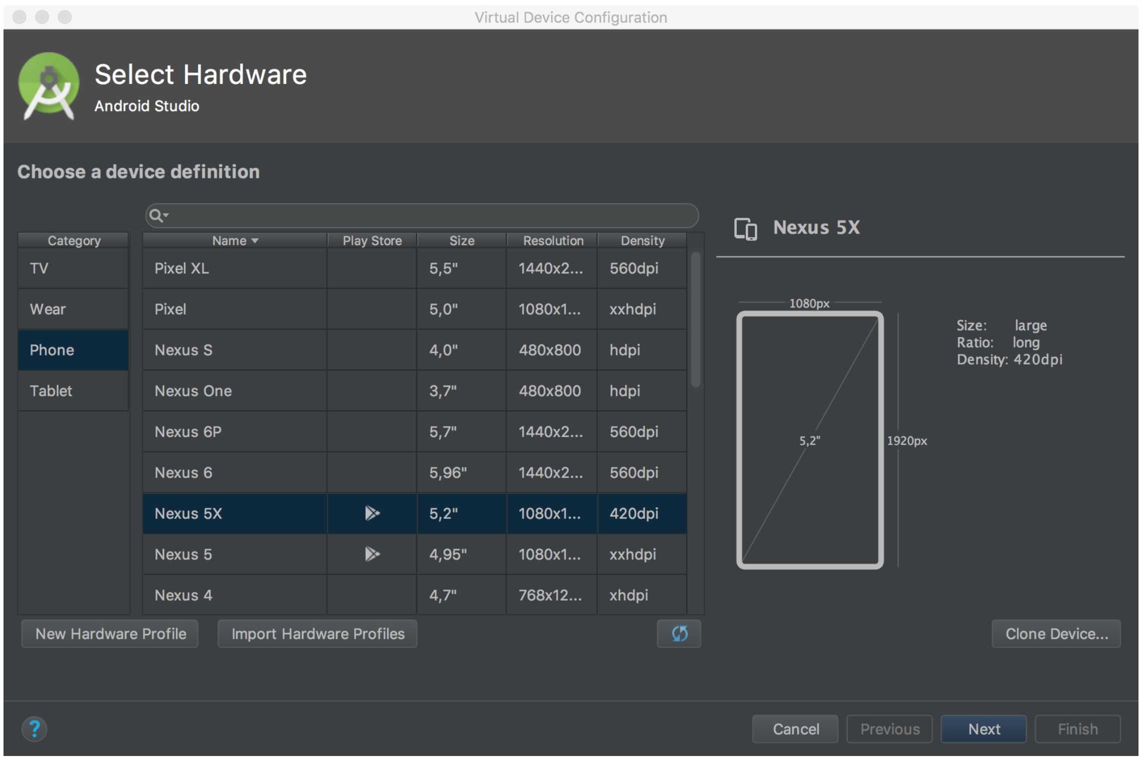
Before starting the application, you have to set the Android Virtual Device (AVD) correctly. To do this, you have to select **Tools >Android >AVD Manager** from Android Studio Toolbar, and a window like this will appears (Figure 1).



Type	Name	Play Store	Resolution	API	Target	CPU/ABI	Size on Disk	Actions
	Nexus ...		1080 ×...	19	Androi...	x86	6 GB	
	Nexus ...		1080 ×...	18	Androi...	x86	4 GB	
	Nexus ...		1080 ×...	22	Androi...	x86	6 GB	
	Pixel C...		2560 ...	22	Androi...	x86	5 GB	

Figure 1: AVD Manager

Now click on **Create Virtual Device...**, select **Nexus 5X** as virtual device Phone (Figure 2) and click **Next**.


Virtual Device Configuration

Select Hardware
Android Studio

Choose a device definition

Category
TV
Wear
Phone
Tablet

Name ▾	Play Store	Size	Resolution	Density
Pixel XL		5,5"	1440x2...	560dpi
Pixel		5,0"	1080x1...	xxhdpi
Nexus S		4,0"	480x800	hdpi
Nexus One		3,7"	480x800	hdpi
Nexus 6P		5,7"	1440x2...	560dpi
Nexus 6		5,96"	1440x2...	560dpi
Nexus 5X	▶	5,2"	1080x1...	420dpi
Nexus 5	▶	4,95"	1080x1...	xxhdpi
Nexus 4		4,7"	768x12...	xhdpi

Nexus 5X

Size: large
Ratio: long
Density: 420dpi

Figure 2: Hardware Selection

Now it is the time to select the System Image, so select the **x86 Images** Tab and in particular, select **Lollipop API 22 (Google APIs)** as in Figure 3, then **Next** and **Finish**. The AVD now is correctly set.

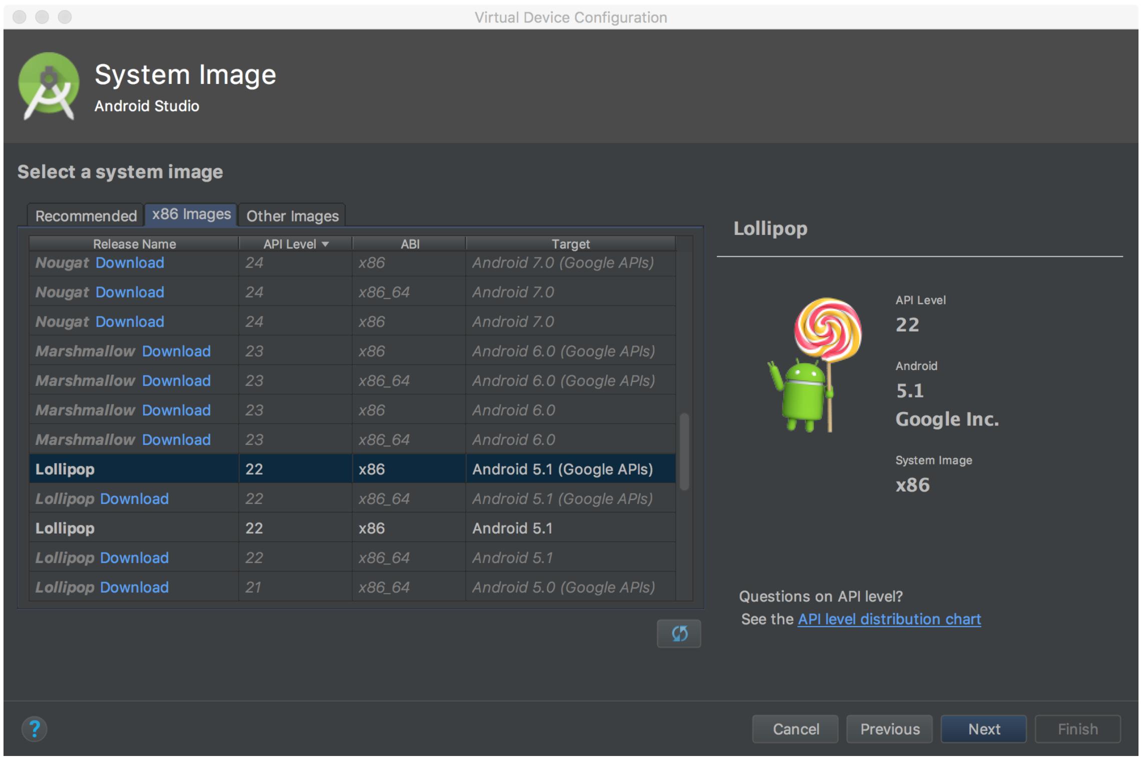
To execute the TheCalendar application, you have to click on **Run** symbol and select the AVD just created (Nexus 5X API 22).

1.3 Installation requirements

The minimum SDK required for the installation is **API 21**, with at least 4MB of memory size available on your device. For time reasons the application GUI was developed **only for** smartphones (portrait and landscape).

2 TheCaregiver Overview

This section is dedicate to explain to you all the application graphical components and how the functionalities required were implemented.



Release Name	API Level	ABI	Target
Nougat Download	24	x86	Android 7.0 (Google APIs)
Nougat Download	24	x86_64	Android 7.0
Nougat Download	24	x86	Android 7.0
Marshmallow Download	23	x86	Android 6.0 (Google APIs)
Marshmallow Download	23	x86_64	Android 6.0 (Google APIs)
Marshmallow Download	23	x86	Android 6.0
Marshmallow Download	23	x86_64	Android 6.0
Lollipop	22	x86	Android 5.1 (Google APIs)
Lollipop Download	22	x86_64	Android 5.1 (Google APIs)
Lollipop	22	x86	Android 5.1
Lollipop Download	22	x86_64	Android 5.1
Lollipop Download	21	x86_64	Android 5.0 (Google APIs)

Lollipop

API Level
22

Android
5.1
Google Inc.

System Image
x86

Questions on API level?
See the API level distribution chart

Figure 3: System Image Selection

2.1 User Experience

2.1.1 TheCalendar Home Page

When you start the application, the first activity will appears and basically it contains an horizontal calendar on top, 23 time slots that can be filled with caregivers reservation and a floating button used by the user to start the auto fill function . As shown in Figure 4 the time slots can have two different types of graphical design. The time slot is gray without the red cross for the following reasons:

- the time slot belong to a past day;
- the hour of the time slot has expired;
- the time slot cannot be reserved because the working time is 9AM- 5PM;
- the time slot is full (all the rooms have been reserved).

Otherwise the time slot has a white background color with a red cross under the hour slot indicator.

2.1.2 TheCalendar Reservation Page

If the user wants to reserve a caregiver for a specific day and hour, he has to tap on associated slot (if available) and a screen like Figure 5 will appears. In this view

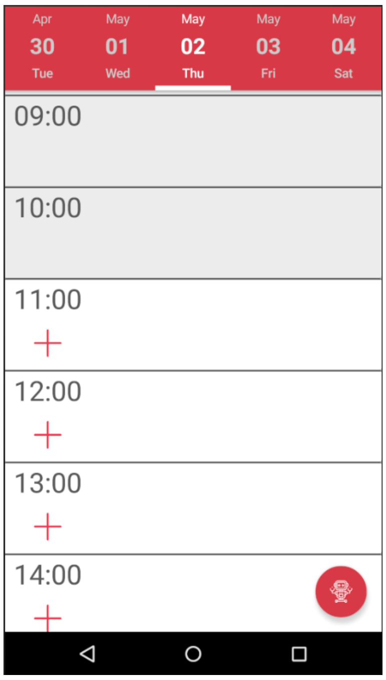


Figure 4: TheCalendar Home Page

the user can select a preferred caregiver among those available in terms of hours per week and overtime (as shown in Figure 6), insert a patient's name and select a room number among those available. The list of available caregivers is also available in offline condition (except for the first time) because the results coming from the GET API call are stored in an internal cache. The list of available caregivers can be also refreshed by swapping down the list.

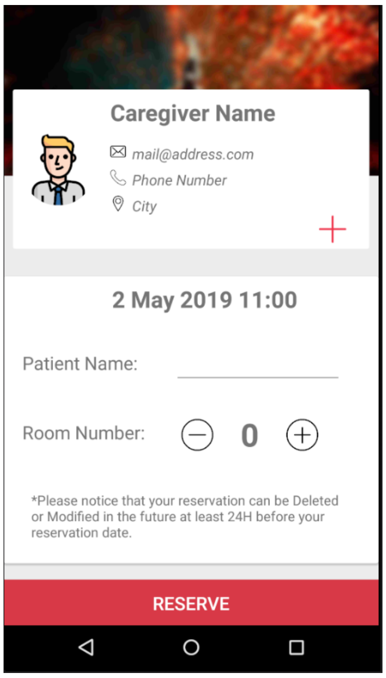
Once the caregiver has been selected, the user can check all the recap information before to confirm the reservation (Figure 7) and then confirm it. The reservation is stored in an internal SQL database and so it is accessible each time you run the application.

2.1.3 TheCalendar Home Page with Placeholder

When the reservation is confirmed, a placeholder will appears on the Home Page and it contains all the information required (Figure 8):

- the caregiver name;
- the first letter of the surname;
- a small rounded caregiver photo;
- room number.

If in the same calendar cell there are more than one placeholder, they will be automatically marked with a color selected by the system.



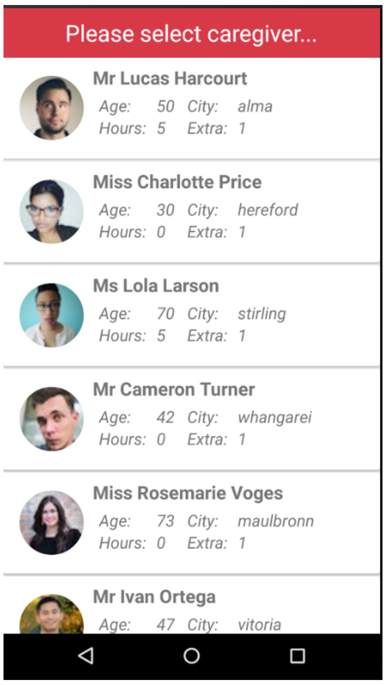
2 May 2019 11:00

Patient Name:

Room Number: 0

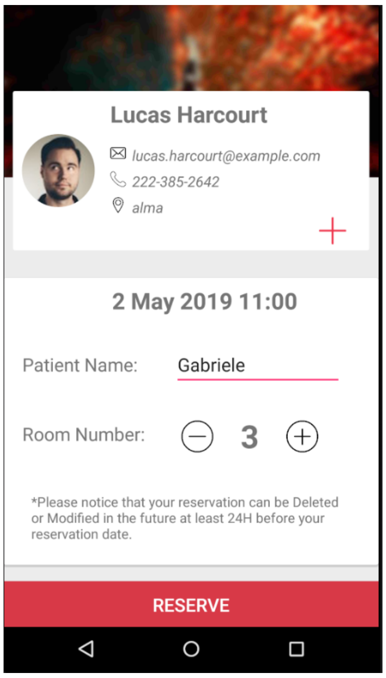
*Please notice that your reservation can be Deleted or Modified in the future at least 24H before your reservation date.

Figure 5: TheCalendar Reservation Page



Mr Lucas Harcourt	Age:	50	City:	alma
	Hours:	5	Extra:	1
Miss Charlotte Price	Age:	30	City:	hereford
	Hours:	0	Extra:	1
Ms Lola Larson	Age:	70	City:	stirling
	Hours:	5	Extra:	1
Mr Cameron Turner	Age:	42	City:	whangarei
	Hours:	0	Extra:	1
Miss Rosemarie Voges	Age:	73	City:	maulbronn
	Hours:	0	Extra:	1
Mr Ivan Ortega	Age:	47	City:	vitoria

Figure 6: TheCalendar Caregiver selection Page



Lucas Harcourt

lucas.harcourt@example.com

222-385-2642

alma

2 May 2019 11:00

Patient Name: Gabriele

Room Number: 3

*Please notice that your reservation can be Deleted or Modified in the future at least 24H before your reservation date.

Figure 7: TheCalendar Recap Information

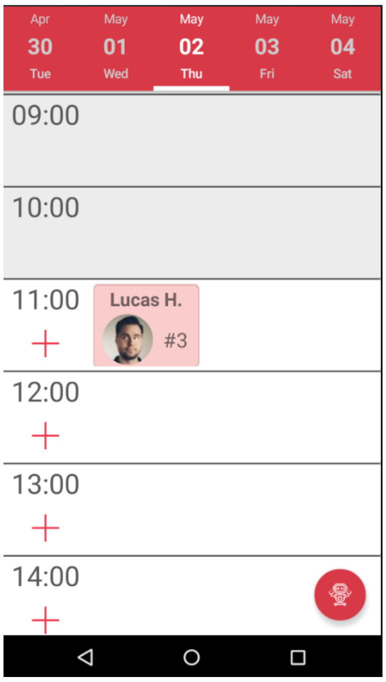


Figure 8: TheCalendar Recap Information

2.1.4 TheCalendar Reservation Modification Page

The reservation can also be deleted or modified before it expires, by tapping on a specific reservation placeholder and selecting the button "Delete" or "Modify" (Figure 9).

When the user tap on "Delete", the reservation is removed from the database. On the other hand, if the user click on "Modify" he can totally modify the reservation details exactly as Figure 5.

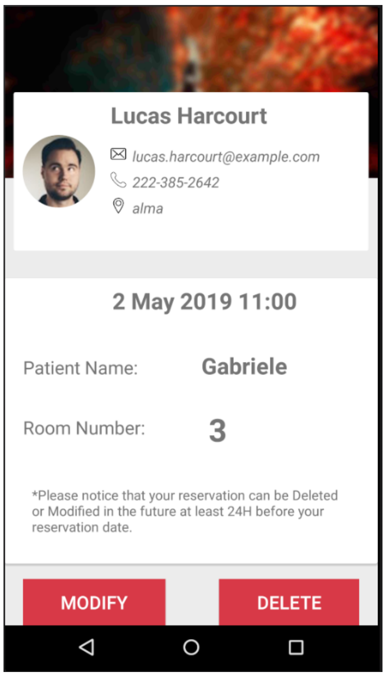

Lucas Harcourt
lucas.harcourt@example.com
222-385-2642
alma
2 May 2019 11:00
Patient Name: Gabriele
Room Number: 3
*Please notice that your reservation can be Deleted or Modified in the future at least 24H before your reservation date.

Figure 9: TheCalendar Delete/Modify Page

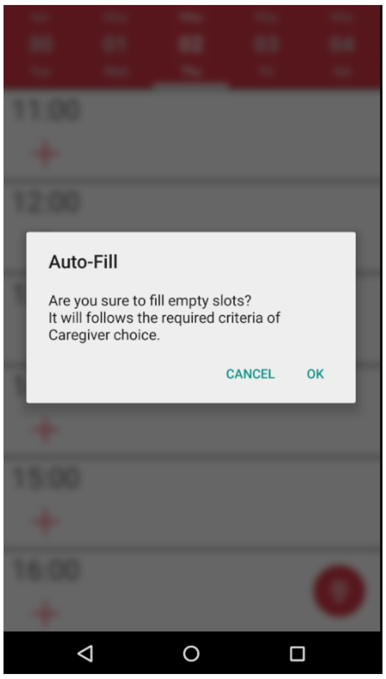
2.1.5 TheCalendar auto fill

As required by the assignment, the auto fill feature was developed and it automatically fill the vacant time slots of a specific day, given a set of constraints and caregiver selection priorities. To use this feature, the user has to tap on the red floating button contained in the application home page (presented in Figure 4) and confirm his choice (Figure 10)

2.2 Implementation choices

First of all in the project, there is a class name **AppParameter** which contains all the assignment constraints that can change during application lifetime. For this reason it was decided to not hardcode them in the project but simply modify them in a specific file. The parameters are the following:

- roomNumber: 10 by assignment rule;



Auto-Fill

Are you sure to fill empty slots?
It will follows the required criteria of
Caregiver choice.

Figure 10: TheCalendar Autofill Page

- startHour: at 9:00 by assignment rule;
- stopHour: at 16:00 by assignment rule (the last hour slot reservable of the day 16:00 -17:00);
- cacheSize: set to 10 MB, used to cache the result coming from GET caregiver request;
- hourPerWeek= 5 hours by assignment rule;
- hourPerWeek= 1 hour by assignment rule;

2.2.1 Auto Fill feature implementation

One of the challenging feature to develop was the auto fill feature. This functionality fill the vacant time slots of a day given the following constraints:

- a caregiver can work up to 5 hours per week;
- a caregiver can have a maximum of 1overtime hour per week;
- working day is comprised of 8 one hour slots;
- a time slot is vacant if no caregiver is assigned to it;
- there are 10 room on the same floor, numbered from 0 to 9;

- consecutive rooms have a distance of 1.The distance is measured as abs(room_number_A-room_number_B).

The auto-fill feature should follow the criteria below:

- if multiple caregivers are eligible, follow the priority scale below:
  - a caregiver with no overtime;
  - a caregiver already working on the same day, in the nearest room;
  - a caregiver that has worked less hours in the past 4 weeks.
- if no caregiver are available, the slot will stay vacant:

One of the "open points" of the assignment was about the auto fill function. In particular it is not explained if the auto fill feature has to fill all the available rooms of all empty time slots of a day or at least one available room of all empty time slot of a day. To make the challenge more interesting I decided to solve it by filling all available rooms of all empty time slots of a day.

The high level algorithm used to reduce the complexity is represented in Figure 11. The idea is to create three different sets: the first contains all the caregiver that still have hours of work available (overtime not considered) in the week considered. The second set contains the caregiver that already working on the same day, in the the nearest room. My solution to the "nearest room" open point was to take the caregiver that already working on the same day in the room closest to the one to be assigned. For example, suppose you have two caregivers: Lucas working in the room number 2, Ivan working in the room number 9 and the room number 0 has to be assigned to one of the two. The room number 0 will be assigned to Lucas because he is nearest (2-0=2 smaller than 9-0=9). As you can see in Figure 12, for each time slot the first room to be assigned is the room number 0, and the nearest between Lucas and Ivan is Lucas (he has the priority). When for each slot the room number 1 has to be assigned, the priority is for Ivan because Lucas is no more available.

The time slots 11:00 and 13:00 already contain at least 1 caregiver, so they are not fully filled.

The last set contains caregivers that have worked less hours in the past 4 weeks. Now, if multiple caregivers are eligible, it is possible to define the priority scale by making the intersection of the three sets in the following order: the first set with the second and if more than one caregiver are available, the result is intersected with the third. Could happen that in the final set result the solution is not unique, in that case the first caregiver in set is taken.

3 Technologies used

In order to provide the best experience and performance to the final user, the application make use of different technologies:

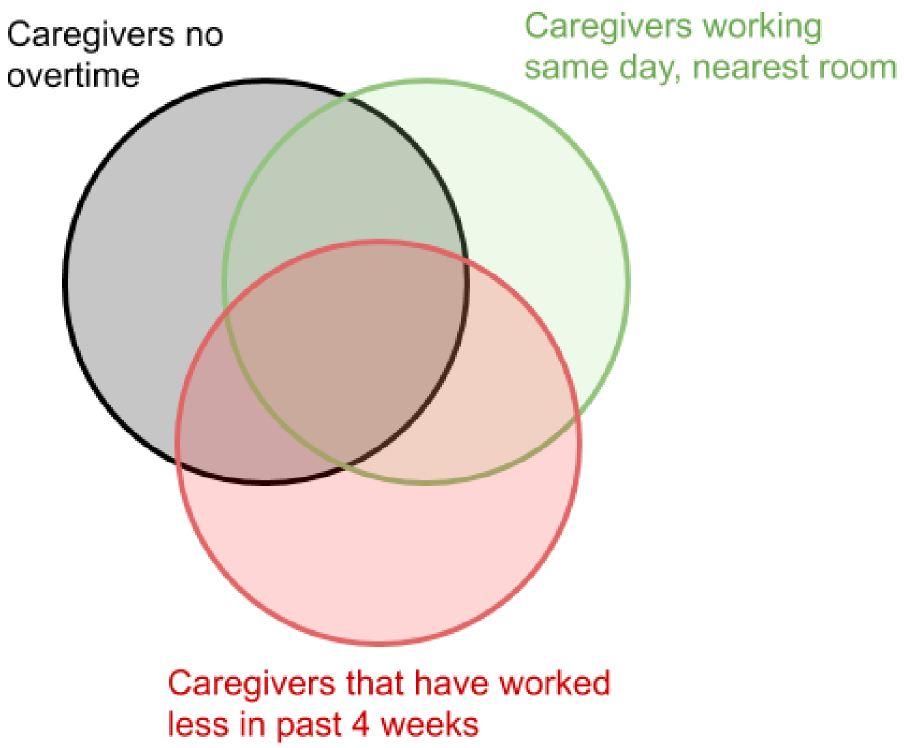
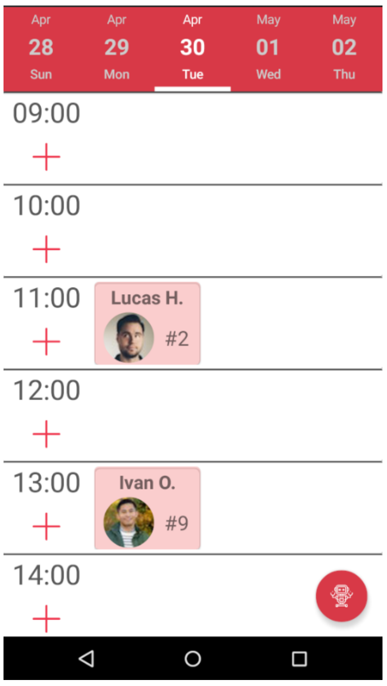
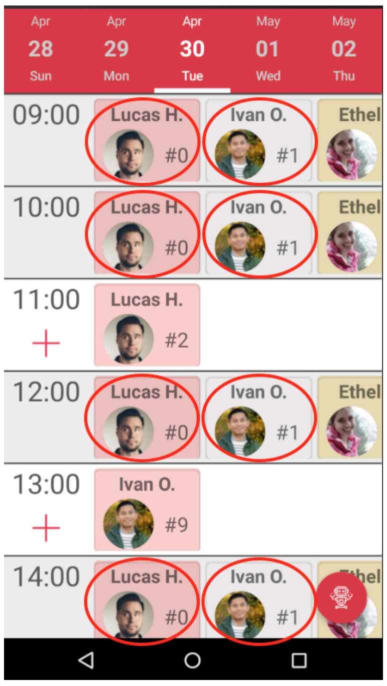

Caregivers no overtime
Caregivers working same day, nearest room
Caregivers that have worked less in past 4 weeks

Figure 11: Algorithm representation



(a) Before auto fill



(b) After auto fill

Figure 12: Caregiver working same day nearest room

- RecyclerView widget is a more advanced and flexible version of ListView. Used the the application needs to display a scrolling list of elements based on large data sets, or data that frequently changes. In the TheCalendar application the RecyclerView was used to contains the hour slots and also to scroll horizontally the placeholder of a specify hour slot (developed a nested RecyclerView).
- Glide for image contents, crop transformations, and caching to achieve best goals in terms of performance and lightweight.
- Blurry, a library that let the possibility to add Blur effect to a background images.
- Retrofit used to send GET API Request and collect and handle the response body. In TheCalendar application Retrofit was used to retrieve caregivers from a GET Request, but also for caching the results. In this way it is possible to use the application also in offline condition, and optimize the internet connection by limiting the number of GET Requests if they are just been send.
- DBFlow, a blazing fast, powerful, and very simple ORM android database library to store reservations and retrieve them as fast as possible. DBFlow provides a powerful SQLite query language. The Database schema is reported in Figure 13.
- HorizontalCalendar which is an external library, used to create the calendar visible in the Home Page of the application.
- CardView a graphic library used to create the placeholders and the caregiver information section in the reservation page. A CardView is basically a FrameLayout with rounded corner and shadow.
- CircleImageView a graphic library used to create the rounded images of caregivers.
- Robolectric which is a framework used for testing. In particular it was preferred over JUnit because allows to test class/methods that use a database in a simple way.

4 System architecture

The high level system architecture of the TheCalendar application is reported in Figure 14. Basically it is composed by different components:

- Reservation Manager is the core of the application, in fact it provide the reservation stored into the database to the View and it creates reservations by asking for caregivers to the API Manager.

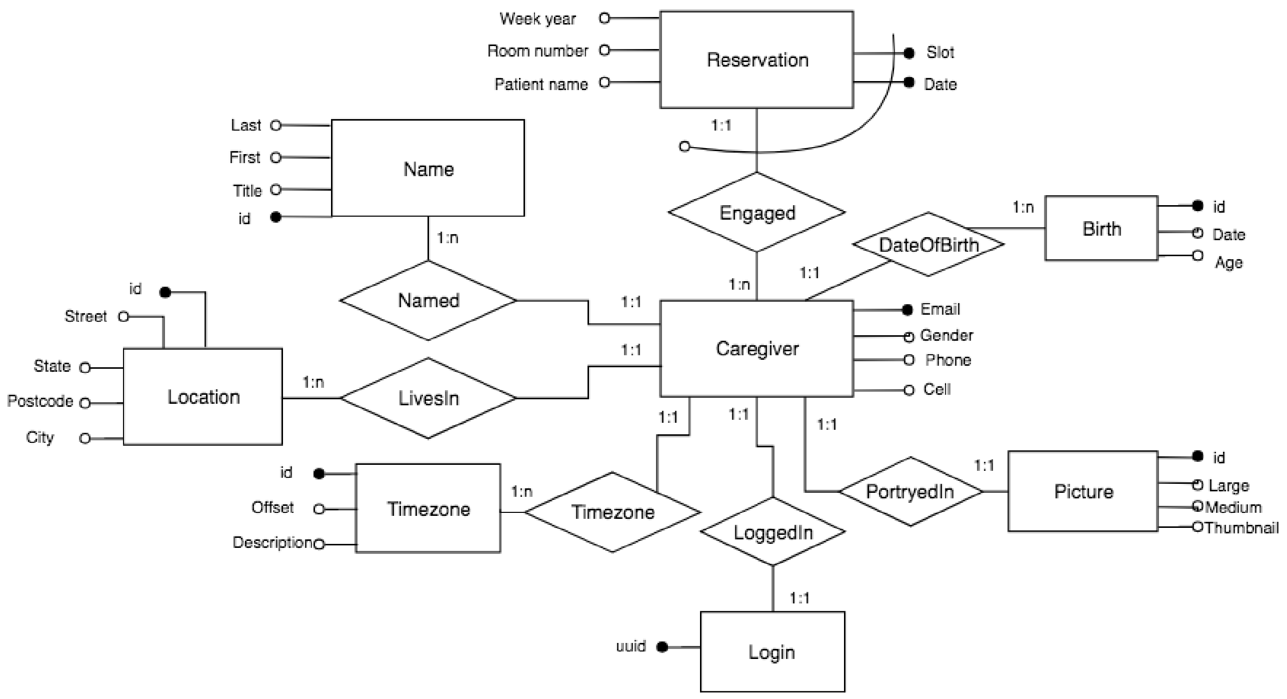


Figure 13: Entity-Relation database schema

- API Manager is the part of the application dedicated to communicate with an external server to send GET request, retrieve response body and cache it. API Manager provides the information stored in cache to the Reservation Manager when required.
- Database Manager is the part of the application dedicated to store and read reservation info into an SQL database. It belongs to the Model part of the Model-View-Controller pattern.
- The View component refers to all the files needed to visualize the information to the user.

5 Design Pattern

5.1 Adapter

The Adapter design pattern is use mainly to mange the RecyclerView that shows time slots and placeholders

5.2 Observer pattern

Observer deign pattern allows the application to define some listeners that in many cases requires to be notified about changes such as asynchronous activities. Observer

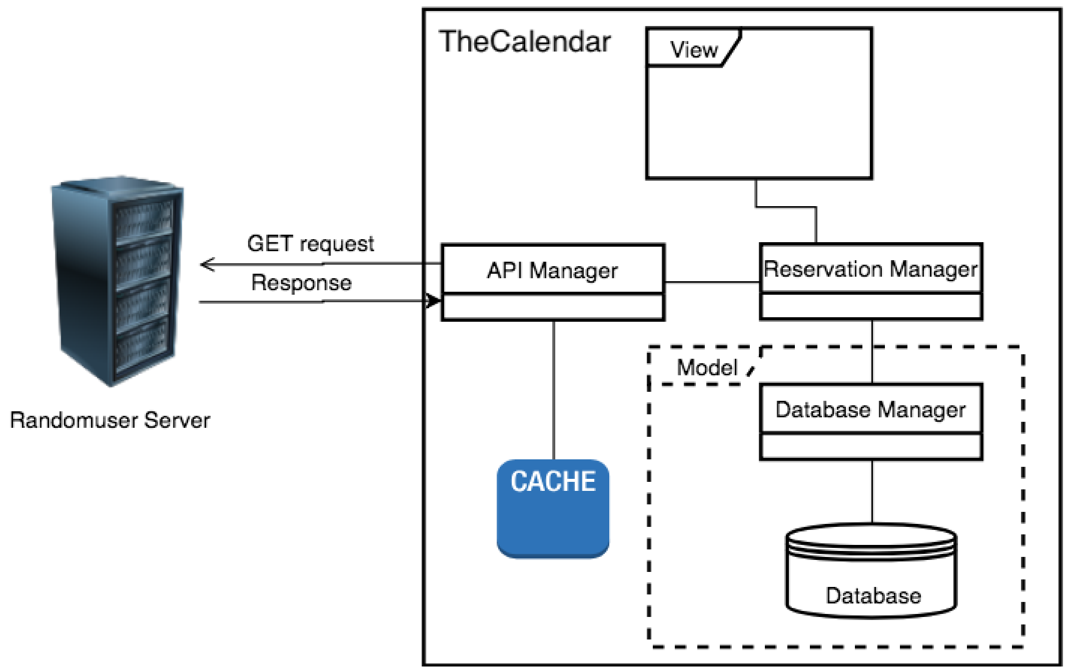


Figure 14: System architecture overview

was used to notify the RecyclerView when a new reservation is added to database by using onDataSetChanged() method, to notify the main activity when the GET API response body is available and so visualize the results and notify to the main activity and also when the auto fill thread completes its execution.

5.3 Singleton

Singleton design pattern is useful to initialize an object only once, and it provides an useful global access to that instance. In the TheCalendar application the singleton pattern was used to initialize a single object of Data class, in fact the Data object contains a list of all active fragments to better manage them.

5.4 Model-View-Controller

This separation principle has been widely applied in TheCalendar application, because it is the most common and the most convenient pattern when it comes to deal with complex application, and allows to design software with the concept of separation in mind. In fact in the Android project you can find the **Model** package which contains all the classes and database used for feed business logic, the View part is represented by all XML files contained into **Resources** package and the Controller part is represented by all Activities/Fragment for business logic.

5.5 Client-Server

The Client-Server communication model is the most suitable for TheCalendar application, in fact it allows to send a GET caregivers request to the server and elaborate the response body in to the client. Inside the Android project was created a specific package named **API** which contains an interface created to manage the request/response to the server/client.

6 Testing

6.1 Logical components

The testing phase was computed by using Robolectric. Robolectric is a framework that allow you tu write unit tests an run them on a desktop JVM while still using Android API. Robolectric provides a JVM compliant version of the android.jar file and it handles inflation of views, resource loading, creation of a database instance, and lots of other stuff.

To better manage the trade-off between time available and testing coverage, it is was decided to cover the most complex part of the project which is the auto fill feature. In particular the following AutoFill class methods were covered:

- getAvailableCaregiversInWeek()
- getCaregiversWorkInDayNearestRoom()
- getCaregiversWorkLessPreviousWeeks()
- getEmptySlots()
- getAvailableRooms()
- addReservationToDB()

Robolectric was preferred over JUnit because since my intention was to test the auto fill feature and its related methods that read/write on database (DBFlow), the simplest way to create a database instance during the test is by using Robolectric framework. You can find the tests inside the **Test >Model** package.

6.2 Graphical components

For testing the graphical component were used: the AVD emulator and a physical device. In particular the AVD device emulator chosen was the **Nexus 5X** with Android API 22 and the physical was an **Honor 5C** with Android API 23.

7 Future improvements

7.1 Improvements on current version

From **algorithms** point of view the improvements could be:

- reduce the number of database accesses which slow down the system;
- reduce the execution time of the auto fill feature;
- creation of a new screen in which it is possible to set the parameters/constraints contained in the AppParameter java class;
- creation of a web service to synchronize the reservations on all devices;
- increase coverage test cases.

From **GUI** point of view the improvements could be:

- creation of a first time app execution tutorial;
- quality improvement of screen style (text style, objects disposition and animations).

7.2 Improvements on new version

In the new versione of the TheCalendar app, could be funny add some extra features like:

- the possibility tho share the reservation with other users;
- show only caregivers that live near the user (exploiting Google Maps API);
- automatically recognize who is the preferred caregiver of a user.

8 Used tools

The tools used to create the TheCalendar project are:

- Android Studio for coding
- GitHub for project version control
- Flaticon web page for free icons
- Photoshop to create TheCalendar logo
- LaTeX editor to write the documentation.
- Draw.io (Google extension) to generate schemas.