Medify – A medication reminder App for Android

Ekrem Emre
HTWG Konstanz – University of Applied Sciences, Germany
Matriculation number: 302110
ek741emr@htwg-konstanz.de

***Abstract*—This paper discusses the design and development of an application for Android, to remind users to take their medicine in a periodic time of day. The source code can be found at [1].**

I. Introduction

A majority of people rely on daily intake of medication. [2] shows that in 2019 53% of the German adult population took at least one medication periodically. Slip ups while constantly taking medication might lead to health problems. To ensure the regular intake, reminders might assist the consumer. A possible tool is an application for smartphones, which notifies the user when time has come for applying the medicine. The Swiss study by [3] shows that while 8% of the asked population already uses such an app, 50% are open for such a tool of aid.

II. Goal of the project

This project aims to design and develop an application for the Android system. It should help the user to keep up with their regular medication intake. First a requirement analysis should provide a in depth view over the necessary components. The following step deals with the conceptualizing a first model of the solution, in order to map out the development. After unfolding design decisions, results are discussed. Finally the conclusion of the project is reviewed.

III. Requirement analysis

First step of designing the application is a requirement analysis. By drafting personas and creating storyboards the needed attributes of the application are worked out.

A. Personas

To simulate users for this application, two personas were created, which represents the user types that might use this application.

1) First persona:

Name Rosalyn Foster
Age 68
Marital status Married
Occupation Retiree
Computer skills poor

Fig. 1. Portrait of the persona Rosalyn Foster. Source taken from [4].

Key characteristics:

- Retiree, former tailor.
- Because of her advanced age, she got forgetful.
- Is regularly taking different medication, which she needs to be reminded of.
- Is concerned about her health, because she just became a grandparent and wants to spent as much time as possible with her grandchildren.

Goals:

- Be healthy.
- Remember taking her medication regularly.

User story: As an older person, Rosalyn wants to be reminded to take her medicine, because she forgets them and wants to get healthy.

2) Second persona:

Name Christian Metz
Age 32
Marital status Engaged
Occupation Physiotherapist
Computer skills strong

Fig. 2. Portrait of the persona Christian Metz. Source taken from [4].

Key characteristics:

- Working as a physiotherapist and doing a lot of sports, therefore is very healthy.
- He's a member in several sport clubs and a volunteer in the local animal shelter.
- His schedule is always pretty full and he's always on the go.
- He wants to increase his health by taking some vitamins, because he suspects he has some deficiencies.

Goals:

- Be reminded, to take his vitamins.

User story: As a healthy person, Christian wants to be reminded to take his vitamins, because he's busy all day and not thinking of it and wants to stay healthy.

3) Summary: These personas show a user group, which are in need to take medication. However they struggle to remind themselves to take them in their daily lives.

B. Storyboards

To illustrate the practical use of a medication reminder application a storyboard was designed. Fig. 3 shows an elderly person, which after some distractions in her daily routine forgets to take her medicine. Fig. 4 shows the same elderly person in a similar scenario. In order to manage her medication, she uses her smartphone to get reminded when to take her medicine.

C. Requirement specification

The requirement analysis shows, that the application must be capable of reminding the user when to take their medicine. The time of notification shall be set by the user. In order to accommodate users who need to take different medications at different points in time, the application shall be able to record multiple medications with multiple alerts.

In order to facilitate the usage of the application, it shall include a way to scan GTIN/EAN–labels and autonomously recognize the name of the scanned medication.

IV. Conceptual model of the solution

For the development of the application, first a conceptual model was designed. The data model was conceived using an entity relationship diagram. The control flow was resulted designing an activity diagram. In order to construct the graphical user interface mockups were sketched by hand.

A. Entity relationship diagram

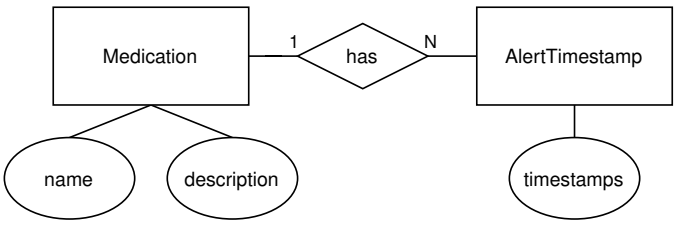

Fig. 5. Entity relationship diagram to depict the data model.

Fig. 5 shows a entity relationship diagram of the application. Two entities emerged when designing a first solution. `Medication` has the attributes `name`, `description` and a `1:n` relationship to `AlertTimestamp`, which has `timestamps`. By designing the data model this way, it is possible to have multiple `Medication`-entries associated with multiple different `AlertTimestamp`-entries.

B. Activity diagram

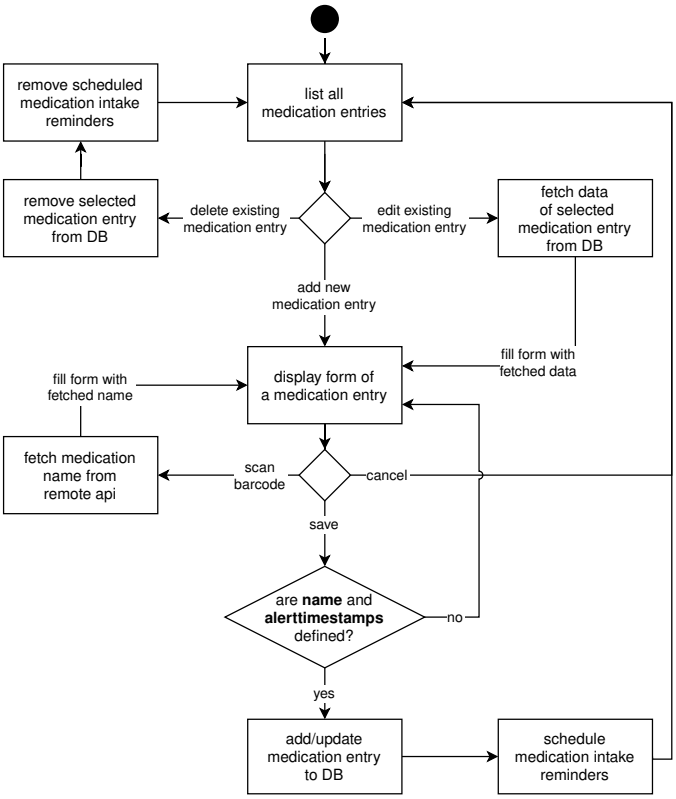

Fig. 6. Activity diagram showing the main workflow of the application.

Fig. 6 shows the workflow of the application. Beside the basic *CRUD* operations to the database, actions for scheduling reminders and fetching data from a remote *API* included. In order to keep this diagram plain, the scheduling and *API* actions are deliberately kept simple.

C. Mockups

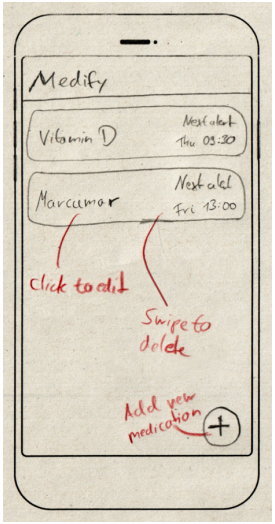

Fig. 7. Mockup of the main activity. Red annotations shall clarify the functionality of the elements.

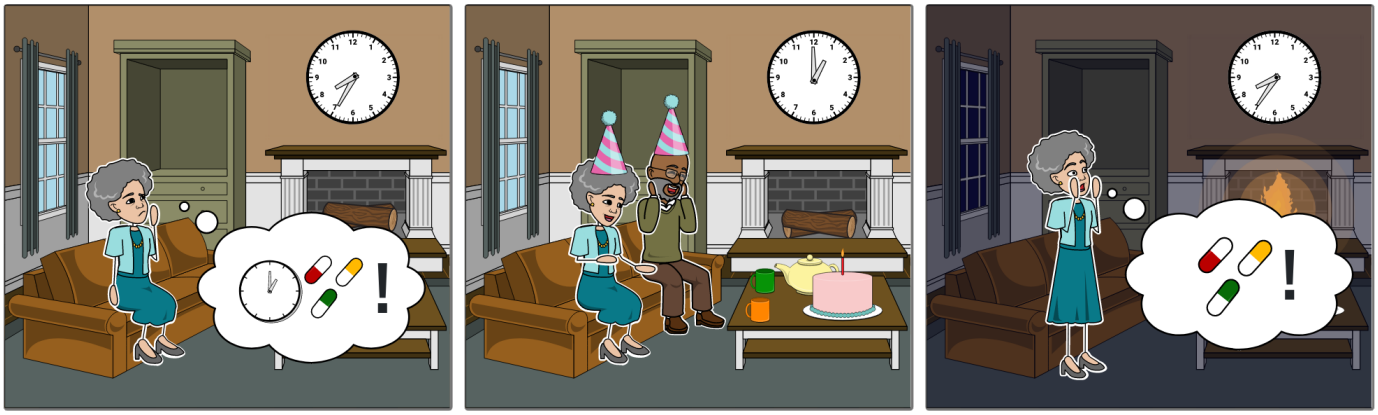

Fig. 3. Storyboard showing an elderly person on her daily activities. In the first panel she reminds herself to take her medication at one o'clock. The second panel shows her enjoying her day with a friend, however she forgets to take her medicine. The last panel shows her in the evening and she realizes her blunder. Graphic created using [5].

Fig. 4. Storyboard showing an elderly person on her daily activities while using a medication reminder application. In the first panel she sets an alert on her smartphone, using the medication reminder app, at one o'clock. The second panel shows her and a friend enjoying their day. The medication reminder app alerts her at the set time, to take her medication. In the last panel she picks up her phone and takes her medication, just as advised by the application. Graphic created using [5].

The mockup of the main activity is shown in Fig. 7. Every medication entry is displayed in a list. The list entries include the names of the medication and the next time an alert is triggered. The entries are clickable, in order to edit them. Swiping an entry shall remove it. A button at the bottom of the screen enables the user to add a new medication entry.

Fig. 8 displays the activity, when a new medication is to be added or an existing one was selected to be edited. Two text input fields enables the user to specify the name and a description of the medication. A list is used to show all the alerts of the associated medication. The entries include the time in a 24-hour notation and abbreviations of the weekdays. The selected weekdays shall be highlighted. Adding new alerts is handled by clicking a button above the list. Removing existing alerts is done by swiping the list entry. Editing existing alerts is not provided. At the lower right part of the screen a floating button enables the user to activate the GTIN/EAN–label scanner. A new activity shall start to handle the scanning of the label. In order to save the medication entry, a button is presented in the upper right top of the screen. To cancel the

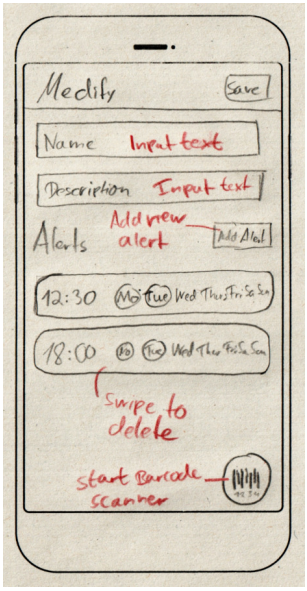

Fig. 8. Mockup of the form activity.

editing or adding operation, the system wide back-button in the navigation-buttons is to be used.

V. Design decisions

This section describes the design decision, which taken while developing the application.

A. Database

In order to implement the *CRUD* operations, the package *Room* by [6] was used. In an attempt to realize the data model in Fig. 5, the implemented model needed to be altered.

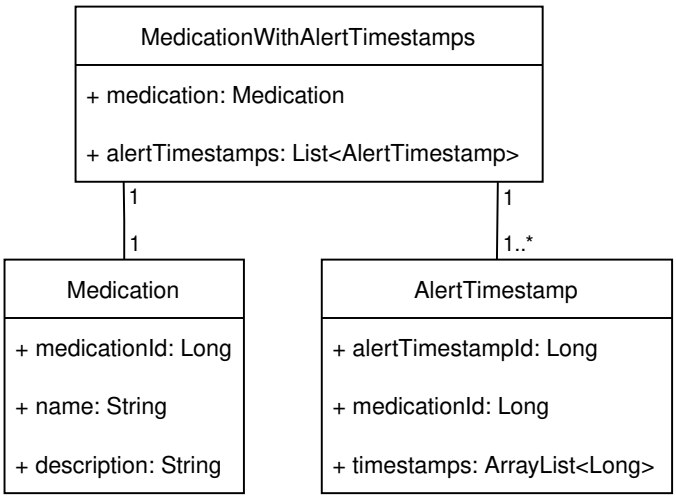

Fig. 9. UML diagram depicting the data model of the application.

Fig. 9 is the resulting UML diagram of the implemented data model. To achieve a `1:n` relationship between the entities `Medication` and `AlertTimestamp` a helper class `MedicationWithAlertTimestamps` was introduced. This way *Room* is able to fetch a medication entry and all its associated alerts. To abstract the data model structure from the controller and view, a `ViewModel` was implemented to manage the database operations, and to offer the view a `LiveData` representation of all medication entries.

An attentive reader might wonder, why the implementation contains a `1:n` relationship, when a single `AlertTimestamp` class has also a list of `timestamps`. One might come to the conclusion, that a `1:1` relationship be sufficient. Or even just the `Medication` class with an list attribute called `timestamps`.

The decision was taken to implement this model the way as in Fig. 9, because a single `AlertTimestamp` entry shall represent an alert at a specific time of the day, which can be repeated at every given weekday. This results the `timestamps` attribute list having up to seven entries and no more. In order to have multiple alerts on the same weekday, the `Medication` class can have multiple `AlertTimestamp` entries.

B. Managing alerts

To manage alerts the class *AlarmManager* [7] was leveraged. This class enables the application to run code at a specific time.

Every time a new medication entry is added, the *AlarmManager* shall schedule new alarms. When a medication is updated, the manager shall update the existing alarms, which are associated to the medication. Deleting a medication, shall cancel the associated alarms. In order to ensure consistency, these operations are included in the `ViewModel`. When the `ViewModel` is commissioned to alter the database, it also guarantees, that the scheduled alarms are consistent to the current database.

The specific time of the alert is provided as an Unix timestamp stored in the `AlertTimestamp` class. The scheduled code is a notification, which provides the user the name and the medication description. Every alert which is added to the *AlarmManager*, gets repeated in a weekly cycle.

C. Scanning GTIN/EAN–label

Scanning the labels on products requires the application to obtain permissions to the camera. By using the package *code-scanner* [8] the scanning is achieved. The scanning-process is started, by starting an activity containing an intent with a result-request. The started activity uses the package *code-scanner*. When a label is successfully registered, the activity sets the recognized GTIN/EAN–number as the result and finishes. The parent activity receives this number and can continue processing it.

D. API calls

Having a GTIN/EAN–number, the application can try to fetch information about the product from an online database. An open database is provided by [9]. This host has a limited number of daily API calls and has a narrow spectrum of products registered. However it is sufficient to showcase the functionality. To manage the API calls, the package *Retrofit* [10] is used. Unfortunately the responses from the database are not in any common form, like *JSON*. Therefore a custom parser was implemented to interpret the server response.

E. User interface

The implemented user interface is similar to the mockup shown in section IV-C. Some feedback for the user is included, like displaying a *Toast* when the user tries to save a medication with no name. The selection of a time of day is done by using a *TimePicker* and multiple *Checkbox* elements for the weekdays. Swiping an `AlertTimestamp` displays a *Snackbar* with a option to undo the deletion. Swiping a `Medication` prompts an *AlertDialog* with the option to confirm or cancel the deletion.

VI. Results

This chapter shows the results of the implemented application, elaborates on the testing procedure and discusses the completion of the requirements.

A. Showcase

To showcase the application, some distinct screenshot were taken described here.

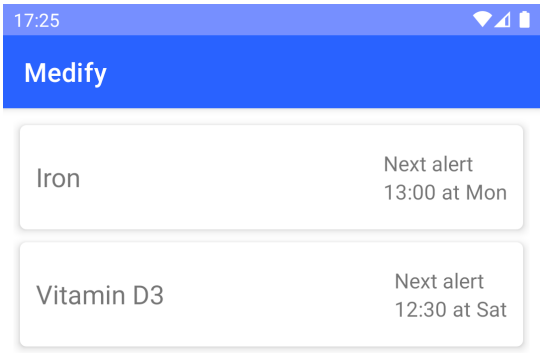

Fig. 10. Clipped screenshot of the main activity.

Fig. 10 shows a clipped screenshot of the main activity. Two medications are shown with their corresponding alerts. This list is scrollable and gets updated every time, when changes occur to the medication.

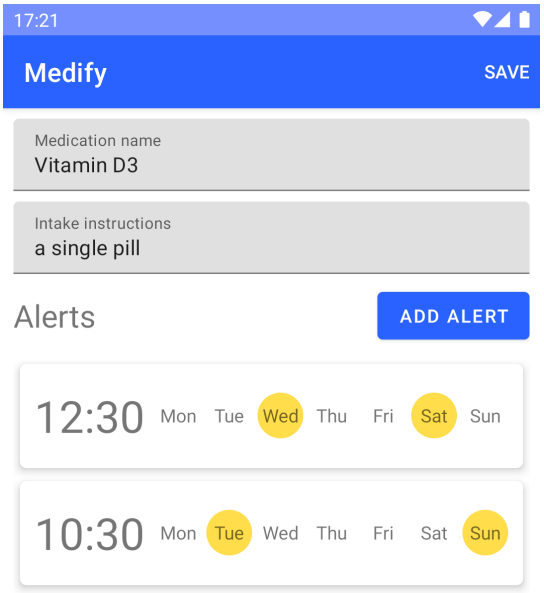

Fig. 11. Clipped screenshot displaying the activity for editing a medication.

Fig. 11 is the result, when the list entry "Vitamin D3" was clicked. The label *Description* was replaced by *Intake instructions* to be more clearer of its usage. As intended the selected weekdays are highlighted and the day of time is displayed.

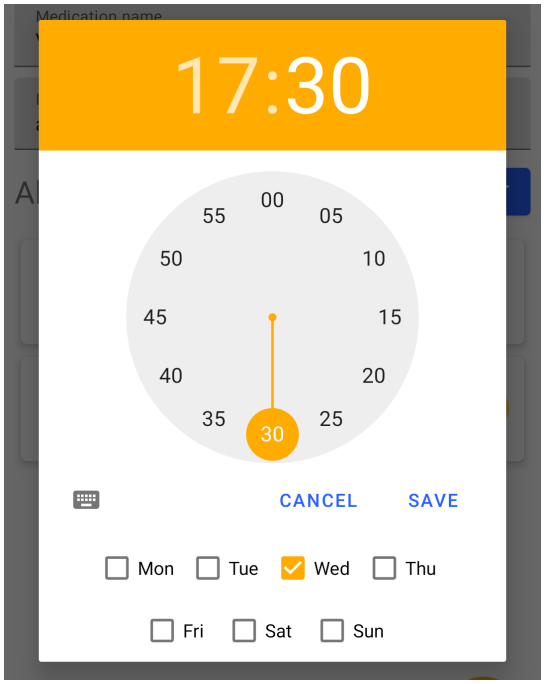

Fig. 12. Clipped screenshot picturing the *AlertDialog* for selecting the alert time.

When pressing the *Add Alert* button in Fig. 11 the *AlertDialog* from Fig. 12 is presented. Here the user is able to select a day of time and which weekday shall include the alert.

If the user presses the lower right button in the activity from Fig. 11, which is not pictured in the screenshot, the GTIN/EAN–label scanner is started.

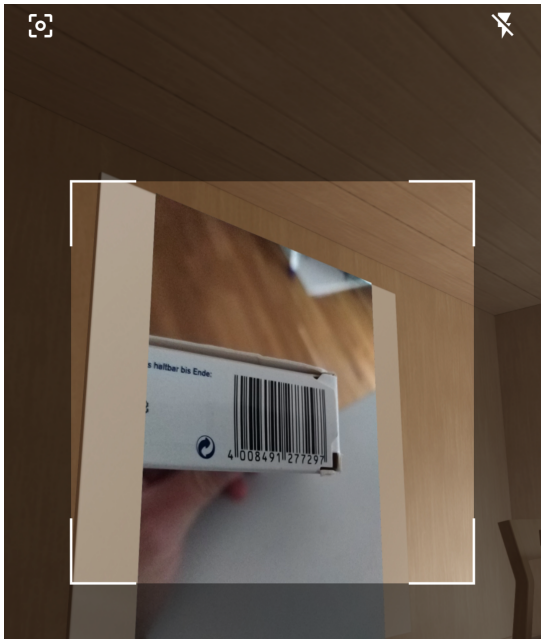

Fig. 13. Clipped screenshot showing the implemented GTIN/EAN–label scanner.

Fig. 13 shows the scanner activity. The button at the upper

right toggles the autofocus of the camera. The button on the other side toggles the integrated flashlight. As soon as a label is successfully scanned the activity finishes and returns to the activity depicted in Fig. 12 and tries to fill out the medication name.

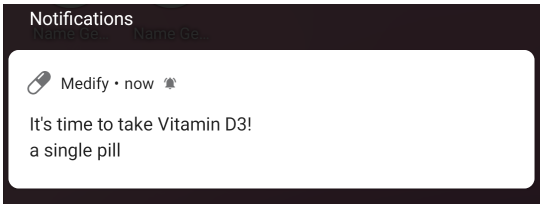

Fig. 14. Clipped screenshot showing the alert of the medication.

When a alert takes place, the application emits a notification, which is shown in Fig. 14. The medication name is shown in the title of the notification, while the intake instructions are shown in the description.

B. Testing

Testing was done manually. The consistency of the database was tested, by applying all the *CRUD* operations and verifying the results. The same was done with the scheduled alarms of the *AlarmManager*. The reliability of the *AlarmManager* was tested by scheduling alerts at 12 o'clock on all weekdays on a physical device.

The handling of the required camera permission was also tested, if the application acts in a predictable way, when the permission is not granted.

C. Degree of completion

The requirements set in section III-C were fulfilled completely. The developed application is able to record multiple medications with alerts, which points in time are chosen freely by the user. The user gets alerted, whenever a scheduled alert is triggered, by a notification. Furthermore a GTIN/EAN–label scanner and an autonomous product name lookup is implemented.

VII. Conclusion

This section concludes the documentation and discusses possible feature ideas of the application.

As explained, all requirements were fulfilled completely in course of the development of Medify. However, there are still some features users would benefit from. Currently if an alert for a medication goes off, only a notification gets dispatched. Future revisions could include a more attentive seeking alarm, like a ringtone which needs to be canceled by the user. Another functionality might include, that the user is able to optionally add a self shot image of the drug to the medication record. This way if an alarm gets triggered, the image will be displayed on the screen, for the user to identify it faster and they don't have to struggle, if they depend on multiple medications. Adding alerts could be more fine-grained. Options like "Remind every n weeks." or even disabling currently unused medications could be useful. Backups and restores of the database when users change devices, would increase portability.

In summary this project has been successfully implemented, using the principles of requirement engineering and building a concept model of the solution, before implementing it.

References

[1] E. Emre, "eco3/Medify: A medication reminder App for Android," *github.com/eco3/Medify/*, [Online]. Available: https://github.com/eco3/Medify/. [Accessed Dec. 23, 2021].

[2] ABDA - Bundesvereinigung Deutscher Apothekerverbände, e.V., "Polymedikation - Bevölkerungsanteil in Deutschland nach Anzahl eingenommener Medikamente 2019 - Statista," *statista.com*, June 2020. [Online]. Available: https://statista.com/statistik/daten/studie/561628/umfrage/bevoelkerungsanteil-in-deutschland-nach-anzahl-eingenommener-medikamente/. [Accessed Dec. 23, 2021].

[3] gfs.bern, AG, "Schweiz - Nutzung Apps Erinnerung Medikamenteneinnahme 2021 - Statista," *statista.com*, March 2021. [Online]. Available: https://statista.com/statistik/daten/studie/527313/umfrage/umfrage-in-der-schweiz-zur-nutzung-von-erinnerungs-apps-fuer-medikamente/. [Accessed Dec. 23, 2021].

[4] P. Wang, "This Person Does Not Exist," *thispersondoesnotexist.com*, [Online]. Available: https://thispersondoesnotexist.com/ [Accessed Dec. 17, 2021].

[5] Clever Prototypes, LLC, "Storyboard That," *storyboardthat.com*, [Online]. Available: https://www.storyboardthat.com/. [Accessed Dec. 17, 2021].

[6] Google, LLC, "Room - Android Developers," *android.com*, [Online] Available: https://developer.android.com/jetpack/androidx/releases/room/. [Accessed Dec. 22, 2021].

[7] Google, LLC, "AlarmManager - Android Developers," *android.com*, [Online]. Available: https://developer.android.com/reference/android/app/AlarmManager. [Accessed Dec. 22, 2021].

[8] Y. Budiyev, "yuriy-budiyev/code-scanner: Code scanner library for Android, based on ZXing," *github.com/yuriy-budiyev/code-scanner/*, [Online]. Available: https://github.com/yuriy-budiyev/code-scanner/. [Accessed Dec. 22, 2021].

[9] "Open EAN Database - Datenbank und Produktbewertung," *opengtindb.org*, [Online]. Available: https://opengtindb.org/. [Accessed Dec. 22, 2021].

[10] Square, Inc., "Retrofit," *square.github.io/retrofit/*, [Online]. Available: https://square.github.io/retrofit/. [Accessed Dec. 22, 2021].