FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

# ASSIGNMENT OF BACHELOR'S THESIS

| | |
|---|---|
| **Title:** | Slevomat Partner - Android application |
| **Student:** | Tadeáš Valenta |
| **Supervisor:** | Ing. Tomáš Krabač |
| **Study Programme:** | Informatics |
| **Study Branch:** | Web and Software Engineering |
| **Department:** | Department of Software Engineering |
| **Validity:** | Until the end of winter semester 2020/21 |

## Instructions

Analyse the existing version of Slevomat Partner application for Android and design and implement a new version. Slevomat Partner app is for business partners of Slevomat.cz Ltd, and its primary purpose is voucher validation and reservation management. The backend API design and implementation are not part of this thesis.

Main goals of the thesis are:
1. Analysis of the existing features, their redesign and reimplementation according to new standards
2. Analysis, design and implementation of an extended set of functionalities
 a. Support for multiple partner accounts on one device
 b. Dashboard screen with data about partners deals
 c. Access to discussions and customer reviews, and the ability to react to them
3. Prepare project for multi-language support
4. Create UI layouts based on graphical designs from the project leader
5. Connect the app to existing Slevomat API
6. Create a user guide for the application

## References

Will be provided by the supervisor.

Ing. Michal Valenta, Ph.D.
Head of Department

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
Dean

Prague March 28, 2019

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Bachelor's thesis

# Slevomat Partner – Android application

***Tadeáš Valenta***

Department of Software Engineering
Supervisor: Ing. Tomáš Krabač

January 7, 2020

# Acknowledgements

I want to thank my supervisor Ing. Tomáš Krabač for guidance, my colleagues for advice and my family and friends for their support throughout my whole studies.

# Declaration

## Citation of this thesis

Valenta, Tadeáš. *Slevomat Partner – Android application.* Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2020.

# Abstrakt

Cílem bakalářské práce je analýza, návrh a vývoj nové verze aplikace Slevomat Partner pro operační systém Android. Funkcionality aplikace jsou: uplatňování voucherů, správa rezervací a další funkce pro péči o zákazníky. Práce také obsahuje analýzu současných funkcí i nových požadavků na aplikaci od společnosti Slevomat, návrh wireframů včetně jejich validace a implementaci aplikace.

**Klíčová slova** Android, mobilní aplikace, Kotlin, Gradle, QR kód, správa rezervací, Slevomat

# Abstract

The goal of this bachelor thesis is to analyse, design and develop a new version of application Slevomat Partner for Android OS. The application includes: voucher validation, reservation management and other customer relations features. The thesis includes analysis of both the current functionalities and new requirements from the Slevomat company, wireframe design and its validation and implementation of the application.

**Keywords** Android, mobile application, Kotlin, Gradle, QR code, reservation management, Slevomat

# Contents

# List of Figures

# List of Listings

# Introduction

Smartphones are still a young segment of personal electronics, but they experienced breakneck growth making the worldwide penetration of smartphones around 40 % as of 2019 [1]. Smartphones capability also increased throughout the last decade, and their robust computational power, camera, location services, touch screens and other features make them ready for wide spectrum of tasks.

Slevomat is originally a Czech company, and it is now part of a British company – Secret Escapes Limited. Slevomat is branding itself as "inspirational portal". It offers customers a wide selection of deals ranging from travel packages and leisure activities to goods. For its business partners, it offers access to over 1.3 millions of registered users. Slevomat Partner is an Android application for business partners of the Slevomat company. It helps them serve customers coming with Slevomat vouchers.

The thesis is split into 6 chapters. In the first chapter of the thesis, you can find a specification of the project goals and analysis of its requirements.

The second chapter consists of an analysis of the current state of mobile development, how applications for mobile phones can be developed and what applications are in the market of voucher scanning and reservation management including the current version of Slevomat Partner.

Android operating system and how are native applications created for it is described in the third chapter.

The fourth chapter deals with the analysis and design of the new application. It includes a full wireframe design of the application and its validation.

The fifth chapter talks about the implementation of the application, what tools were used, what libraries were used and how specific parts of the application were implemented.

Testing of the application can be found in the sixth chapter. It contains description of test types and design and implementation of the tests.

The conclusion contains evaluation of thesis' goals and the future of the development of the application.

Chapter 1

# Project specification and goals

## 1.1 Project specification

Slevomat Partner application is a business application of Slevomat company. It is in a functional state but lacking behind the market; therefore, I was allocated to a project of creating a new application afresh. Below are specified functional and non-functional requirements for the project.

## 1.2 Functional requirements for the new application

1. Voucher scanning and validation
2. View reservations
3. Create and edit reservations
4. Change account without logging out
5. View dashboard with information about deals
6. View deal discussion and reply to questions
7. View deal rating and reply to ratings

## 1.3 Non-functional requirements

1. Operable on Android OS 5.0 and newer
2. Application is available in Czech
3. Multi-language setup of the project

# Chapter 2

# State of the art

## 2.1 Mobile development

A mobile application is a piece of software developed primarily for mobile devices like smartphones or tablets. They are limited by lower performance and shorter battery life of the device compared to a desktop computer. Therefore, they need to use their resources reasonably, and they should not block the user from using the phone even when doing heavy computations. If the application prevents the user from using the phone for a time longer than the system allows, it usually kills the application.

Every OS has a different mechanism of asynchronous programming to avoid this result. It is mostly used for networking, notifications, alarms, loading data from the database and other time-consuming or repeating tasks. A modern mobile operating system takes over these tasks from the applications and tries to manage them with as little battery consumption as possible [2].

### 2.1.1 History of mobile development

Mobile application development is an ever-changing domain because of the abrupt hardware and software advancement connected with IT field expansion. In the 1990s, the first mobile OSs were released. In the 2000s, Symbian led the market but a decline started, and in 2010, the market share leader position was overtaken by Android. Symbian's decline continued, and in 2011, Apple's iOS gained second place, and today, Symbian's share of the market is almost non-existent. Currently, Android leads the market with a 73 %, followed by iOS with 25 % [3].

Mobile applications started as small units of software to solve a minimal set of problems – a small game, a web browser, or a music player. Therefore, the user could have chosen their own set of functionalities for their device without cluttering it with unwanted functionalities usually coming with larger desktop applications.

Mobile applications have different standards of data persistence and usage – they make use of touch screen, gyroscopes, light sensors and other features that differentiate mobile devices from the desktop versions. With faster hardware and a larger screen with better touch recognition came more complex applications that can boldly compete with their desktop counterparts.

The development of mobile applications has many variations now – ranging from native development for a specific platform to progressive web applications [4].

### 2.1.2 Process of mobile application development

The process of mobile application development, in general, copies the process of general software development. The core difference is in the UX/UI design which needs to be thought-through even more than the desktop one as the screen size is smaller and due to the fact that the device can lose internet access at any moment and should be able to still work without it. The developer, therefore, needs to take care of caching data from the server, updating it in the background, and other similar operations.

## 2.2 Native vs Hybrid vs Progressive Web Application development

Before the application development starts, the first question is what technology to use. The three main types are native, hybrid, and Progressive Web Application (PWA). Native and hybrid applications can be found in the app stores next to each other, and for the user, they can be indistinguishable.

### 2.2.1 Native development

Native development means developing and optimising an app for each platform using its native SDK (such as Android SDK for Android, iOS SDK for iOS). The advantage of this approach is having the newest features of the SDK available when they are released, and the performance of native apps is overall better than with hybrid ones. The cons of this type of development are that you need to develop and test the application for each platform separately; therefore, the development costs more.

### 2.2.2 Hybrid development

Hybrid development is a combination of web development and native development. One codebase can serve multiple platforms thanks to the framework, which runs over the platform OS. The framework works as a middle layer that translates operating system API calls to platform-specific ones.

A disadvantage of this approach is that the framework SDK is updated after the native SDK is released, which means that new features cannot be added to the application before the framework developers implement them into the framework and that the middle layer may slow down the application, which leads to a less smooth experience.

For example – React Native, one of the most popular mobile hybrid frameworks, uses so-called Bridge layer that connects the JavaScript part of the application with the native part to get functionalities like geolocation, motion sensor data and above all it uses native components of the operating system [5].

Examples of compiled frameworks without middle layers include Flutter, and Xamarin when using Xamarin.Forms (UI building framework) [6] and AOT (Ahead Of Time) compilation [7]. These two frameworks have the ability to imitate native components and create the user interface inside the framework itself. That makes them comparable in terms of speed with native applications when used carefully [8] [9].

### 2.2.3 Progressive Web Application development

Progressive Web Applications or PWAs were first named in 2015 by Alex Russell from Google. Google also defined 10 key concepts of PWAs which differentiate them from typical web applications – the main difference for the user is based on three concepts: Internet-free, Installable, and App-like.

These three concepts combined mean that user can use the PWA as an installable application from the store without the need to install it on their phone. This behaviour is made possible through Service Workers technology. Service Worker runs in the background and tries to service requests from the cache loaded when the PWA was loaded. In combination with background preloading, the website should load almost instantly and be able to send push notifications, work offline, and always stay up-to-date [10].

### 2.2.4 Conclusion

Although the hybrid development is growing, the native application still has an edge in performance, support from the community and the availability of libraries. Concerning the fact that the task is to develop the application solely for the Android OS and that my knowledge of JavaScript is minimal, which would slow down the development, the native development was chosen.

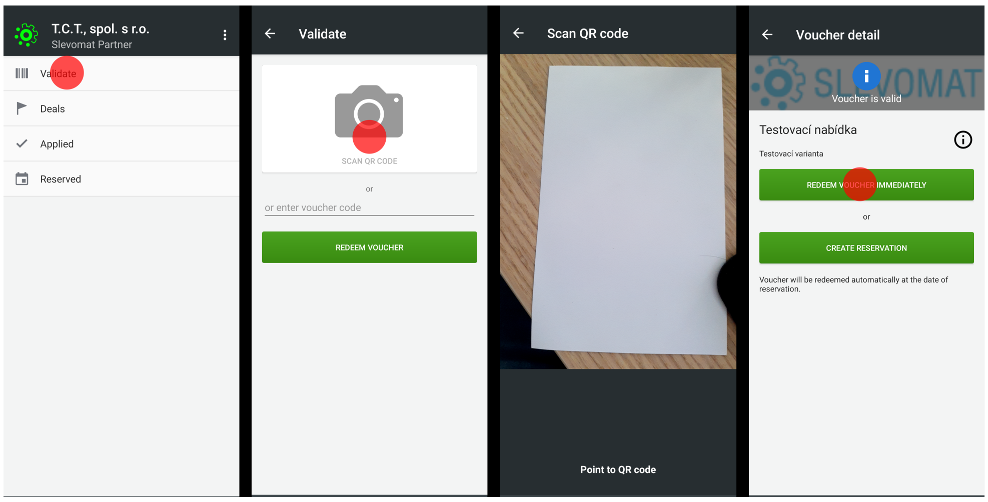
## 2.3 Slevomat's existing solution analysis

Slevomat already has the Slevomat Partner application. It has limited functionalities and falls short in terms of design and ease of use. The current application's development started in 2015 and was finished in 2017. The development was put on hold until now.

### User interface

The application has outdated controls with too many steps needed to solve simple use cases. All use cases need to start in the main menu screen, and from there the user can continue to the voucher validation, making reservations or information about his deals.

An example can be seen in the Figure 2.1. For the voucher validation use case, which is 90 % of the application usage, the user needs 3 taps and is led through 4 different screens.



Voucher will be redeemed automatically at the date of reservation.

Figure 2.1: Voucher validation with the current version of Slevomat Partner

### Technology

The current version is written in Java. One of the significant problems with the older version was the QR code scanner library which had been discontinued and needed to be replaced. Although the replacement works, it has also been discontinued. Another major pitfall of this version is the inability to use multiple partner accounts on one device without logging out of the account and then logging in again.

## 2.4 Other existing solutions

The problem of applications made for partners of other services is that the applications are mostly publicly available to download, but their functionalities are hidden behind a login screen. That makes them hard to analyse without a formal contract with the company. Information about the applications was obtained from public screenshots, public reviews and interviews with users of the applications.

### 2.4.1 GoOut Scanner

GoOut Scanner is an application created by GoOut ticket selling company. Its primary purpose is to scan and validate event tickets as fast as possible. The tickets can be validated via QR code or text code. An advantage of the application is the possibility to search for a client by name. It has no reservation management or other advanced functionalities [11].

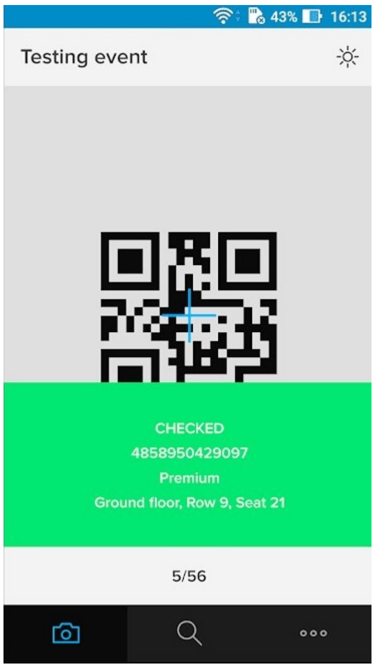


Figure 2.2: GoOut's scanner after ticket validation

### 2.4.2 Restu RIS

Restu is a company with a business centred around restaurant reservations. The RIS application helps restaurants organise their reservations and shows them reviews they have received from guest coming from Restu. It also shows some statistics about the restaurant and its customers (number of reservations, guests, average review). The application is no longer developed and its last update was released in November 2018 [12].

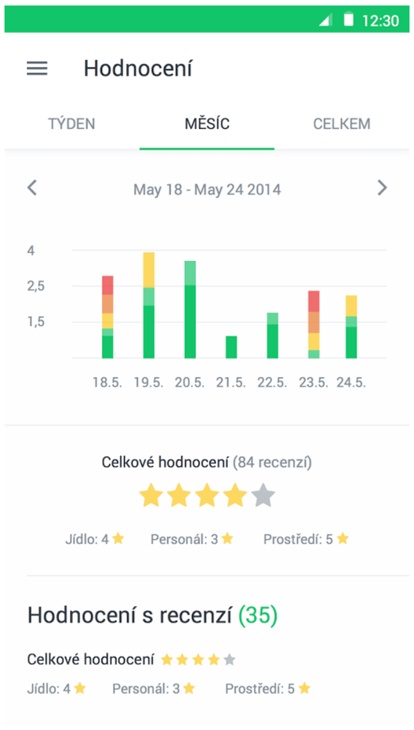
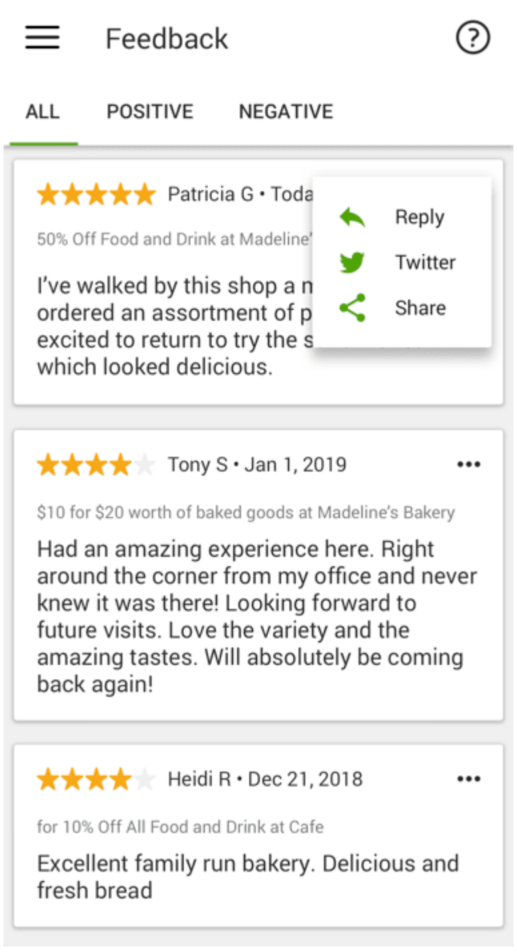


Figure 2.3: RIS's rating screen

### 2.4.3 Groupon Merchants

International company Groupon has its own Android application for its partners. It can scan and validate vouchers. The user can respond to reviews received and can manage the campaigns of the merchant. It has a modern and clean look in accordance with Material guidelines [13].



Patricia G • Toda

50% Off Food and Drink at Madeline'

I've walked by this shop a n
ordered an assortment of p
excited to return to try the s
which looked delicious.

Tony S • Jan 1, 2019

$10 for $20 worth of baked goods at Madeline's Bakery

Had an amazing experience here. Right around the corner from my office and never knew it was there! Looking forward to future visits. Love the variety and the amazing tastes. Will absolutely be coming back again!

Heidi R • Dec 21, 2018

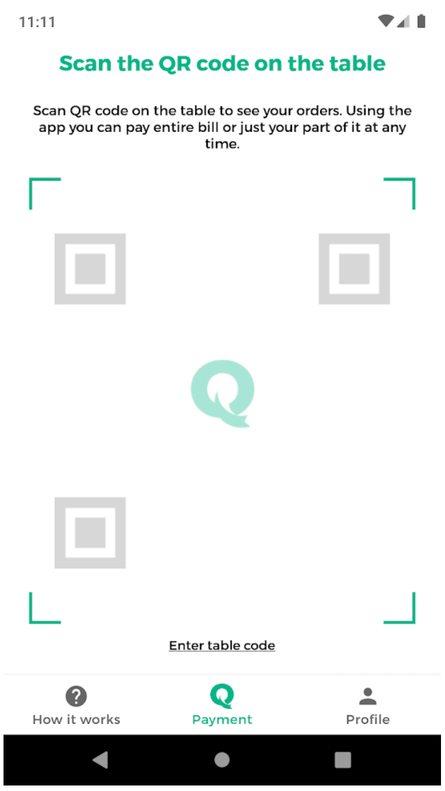
for 10% Off All Food and Drink at Cafe

Excellent family run bakery. Delicious and fresh bread

Figure 2.4: Groupon's feedback screen

### 2.4.4 Qerko

Qerko is an application from a company *It is paid!*. It offers restaurants payment service via QR codes on the tables. It is the only application listed focused on an end-user. The user scans the table QR code with the application, selects what food or drinks he/she want to pay and pays online with his debit/credit card. The user needs to understand the service quickly; therefore, the application has a very straightforward and simple UX [14].


11:11

Scan the QR code on the table

Scan QR code on the table to see your orders. Using the app you can pay entire bill or just your part of it at any time.

Figure 2.5: Qerko's table scanning

Chapter 3

# Android

This chapter introduces the Android operating system. It aims is to explain to the reader the basic inner-workings of Android OS.

Android is a Linux-based operating system initially created by Android Inc., which Google bought in 2005 and the first beta version was released in 2007. It is widely used around the world thanks to its open-source nature. Android is not made solely for mobile phones but also for tablets, TVs, car entertainment systems and can be used on almost any other device that can run its virtual machine. Even though Android applications can be written in Java, they are using Dalvik Virtual Machine or Android Runtime (depending on the version of Android OS) instead of Java Virtual Machine.

Google implemented their version of Java API for its virtual machines and that lead to a legal case between Oracle and Google [15] [16]. Android Open Source Project is the core element of the Android OS that has all elementary features of mobile operating system without Google Mobile Services included. Many device manufacturers use the AOSP project as the core for their own OS [17] [18].

## 3.1 Basic elements

This section describes the most widely used elements of native Android to introduce most of the elements used in the application design.

**Activity** It is the main user-interactive component of the Android SDK. All of the views (basic building blocks for Android user interface) and fragments exist inside an activity, and their visibility depends on the visibility of the activity they are attached to. Most of the applications have the Main Activity (with the same name), which is the first screen the user can interact with although it is not mandatory.

An important concept connected with activities is the Back Stack. The Back Stack holds activities that the user navigated through and allows the user to return to the previous screen quickly. Every visited activity is pushed to the Back Stack when a new activity replaces it. When the user taps the back button, the activity is popped out of the stack, and the replaced activity is destroyed [19].

**Service** It performs long-running operations. It does not provide any user interface. Services can be started by other components of the application and can run even if the user switches to another application. Services are mostly used for network connections, media playback, I/O operations, and interacting with Content Provider.

Services can run in the background, the foreground, and can be bound to an application. Bound services run only as long as another application is bound to it. Multiple components can be bound to a service at the same time, but when all of the components unbind, the service gets destroyed. Foreground services need to show a non-dismissible notification while they are performing an operation noticeable by the user, for example, while playing audio.

Background services, on the other hand, are used for operations unnoticeable by the user (network and I/O operations). With Android 8 (API level 26) the system imposes tighter restrictions on running background services and Scheduled Jobs were introduced. Scheduled Jobs serve as a replacement for Services. System's managers control when Scheduled Jobs are given the resources to run [20] [21].

**Broadcast Receiver** It makes use of a publisher-subscriber pattern for keeping all applications informed about the state of the operating system and for the applications to be able to send broadcasts to the system. Applications can subscribe to receive information about internet availability, incoming phone calls, battery level changes and other events. For more information, refer to [22] [23].

A crucial broadcast that applications can receive is the Intent. Intent's main use is in the launching of new activities. Explicit intents are used for starting another activity inside the same application. Android SDK also defines implicit intents. These are used for calling for an action without requiring a specific application. For example, when the application wants to start a phone call, the application sends out `ACTION_DIAL` intent, and the user can choose any application from the ones subscribed as receivers of the `ACTION_DIAL` intent [24].

**Content Provider** It is a component that manages data sets. Structured data can be shared via the Content provider with different activities throughout the application and even with other applications. The system holds Content Providers for the user's calendar and their contact list [25].

**Fragment** It represents a modular section of a user interface. Multiple fragments can sit in one Activity, or one can fill it whole. The fragment must always be hosted inside Activity and cannot stand on its own. Fragments are commonly used to combine several screens from a phone-centered interface in a tablet-sized device [26].

## 3.2 UI elements

Information on visual and structural elements are in this section. UI elements serve as the cornerstone of a mobile application that the user interacts with.

These elements can be sorted into two groups: Views, also called widgets, and View Groups, also called layouts. Views are visible to the user and he/she interacts with them. View Groups work as containers, they are invisible to the user but define a structure for the visible Views and other ViewGroups [27].

**View** The View class represents a fundamental building block for user interface; it represents a rectangular area of the screen and is responsible for drawing and event handling. Every other view and view group class inherits from it. Developer can create custom views inherited from this class [28].

**TextView** It is the most common visualisation of text on the screen. The user cannot directly edit it, TextView is editable only by the developer in code [29].

**EditText** It looks similar to TextView but can be directly edited by the user [30].

**Button** It is a child View that performs an action when the user taps or clicks it [31].

**ImageView** It handles displaying images. It can display bitmap or vector images, scale them and do some types of transformation of the image [32].

**ViewGroup** It is an abstract class for views that can contain other views (children). Layouts are inherited from it [33].

**FrameLayout** The most simple layout is the FrameLayout. It is mostly used for holding only one child view as it stacks its child views on top of each other [34].

**LinearLayout** It arranges its children into one column or row. It is straightforward to use, and, along with nesting, the developer can create almost any layout with this element. A drawback of this solution is that multiple nesting slows down the graphic interface [35].

**ConstraintLayout** It arranges its children using relative positioning. Every child view needs to have attributes defining their position in relation to its parent or other view in the layout. It has different types of attributes available – position, margins, bias, visibility, dimension and others. It has advantage of minimising the nesting of layouts [36].

**RecyclerView** The RecyclerView component is an advanced list view that uses view holders to keep items of the same type in the list. It is used when the count of the items is unknown, or all of the items cannot fit to the screen of the phone. It reuses the holders and fills them with data from the items. Advantage of this approach is lower CPU usage because RecyclerView recycles the holders and does not destroy them, it loads new items into the recycled views every time it is needed [37].

## 3.3 Building an Android application

Building is a process of compiling source code and other dependencies and combining the compiled files, resources and keystore (cryptographic container) into an executable application. During this process, obfuscation and minifying also take place. With advanced build settings, the developer can create two and more different applications with a shared codebase but different so-called "flavour" [38].

### Gradle

Gradle is an official build tool for Android. It is included with the Android Studio and works with it out of the box. Gradle has benefits over other the build tool Maven with its shorter and more clear syntax and faster recurring building thanks to the better caching.

### Gradle DSL and Gradle Kotlin DSL

From the start, Gradle used Gradle domain specific language (DSL) based on Groovy (a dynamically-typed language) as the programming language for Gradle files. A drawback of the language is deficient support of syntax check in Android Studio.

In 2016, Gradle Kotlin DSL was released [39] [40]. As the name states, it is based on the Kotlin programming language. Kotlin is a statically-typed language and therefore an IDE can support more of its functionalities than

Gradle Groovy DSL's – for example: auto-completion, quick documentation, navigation to source, refactoring.

Gradle Groovy DSL and Gradle Kotlin DSL [41] are both used in this project. The disadvantage of Gradle Kotlin DSL is that it is early in its adoption state, and there are not enough supporting materials for it outside of official documentation.

## 3.4 Kotlin vs Java

### Java

Java is a class-based, object-oriented programming language which development started in 1995 at Sun Microsystems [42]. Java is a statically-typed, compiled language, and Java Virtual Machine is used to run Java code; therefore, Java belongs to the "write once, run anywhere" group of programming languages. In Android, the JVM is replaced with Dalvik Virtual Machine or Android Runtime (ART) [43].

### Kotlin

Kotlin is an inter-operable statically-typed programming language created by JetBrains in 2011. The language was fully open-sourced in 2012 under Apache License 2.0, and the Kotlin Foundation was created by JetBrains and Google to take over responsibilities for the language development in 2018 [44].

Kotlin can be compiled via different compilers for different environments. For Android applications, the vital part is Kotlin JVM which is inter-operable with JVM and therefore also DVM as it uses the same API as stated above [45].

### Comparison

Java has been an officially supported language for Android from the start of the OS development. Google added support for Kotlin in 2017, and at Google I/O 2019, they announced their first-class support of Kotlin for Android [46] and added a Kotlin plugin into their Android Studio IDE out of the box. The language was quickly spread amongst developers. They name mostly its null safety, extension functions and Java inter-operability as their fundamental reasons for switching to Kotlin.

The downsides of the Kotlin language are tightly connected to its novelty. There are not many Kotlin experts now, and the community around Kotlin is much smaller than the Java one. Also, Kotlin's compilation takes longer than Java compilation, but the difference is not significant for mobile applications as they are relatively small [47].

### Conclusion

I chose Kotlin for the new Slevomat Partner application for multiple reasons. The first one is that the Slevomat Client application is written in Kotlin and the other is a personal fondness for the language as the code is shorter than Java's and, from my point of view, more comfortable to read.

## 3.5 Android SDK

The Android software development kit includes different tools for increasing developers' productivity – specifically a debugger, libraries, an emulator, documentation, sample code, and tutorials. Android SDK is available for Windows, macOS and Linux. Solely the SDK can be used to develop applications, but programmers mostly use IDEs. Android Studio, the most popular IDE for Android, comes with Android SDK in its package [48].

## 3.6 Libraries

Android libraries are not so different from any other software library. They work as a software unit with API, allowing the developer to use methods instead of creating their own. Libraries can speed up the development of a software project, where new or independent implementation is not needed. Using a library makes the software dependent on the library, and if the library development stops, it might cause problems for software development. For this reason, projects, where independence is essential, are created without using libraries outside of the control of the developer [49].

In 2018, Google introduced Android Jetpack library set. This set of libraries contains all kinds of libraries and tools, from ones helping with component lifecycles to one making designing UI easier. Not all the libraries are currently in a stable state.

One of the most important libraries from the Jetpack is AndroidX library – a replacement for Android Support Libraries. They helped the developer use new methods even when targeting devices with an older operating system, AndroidX does the same, but is better organised internally.

In 2018, Google created a guide to app architecture – amongst other things it contains the usage of lifecycle-aware components. We can find LiveData and ViewModel libraries in Jetpack; they help the developer with creating the lifecycle-aware architecture.

The CameraX library needs to be mentioned because the camera is one of the main system resource used by the Slevomat Partner application. It aims to replace the Camera library and make using the device camera easier for the developer, which is quite a challenge because of the different device manufacturers with different specifications for each camera. The CameraX library is still in the alpha state though, and it is not suitable for production, for that reason, it is not used in the Slevomat Partner application [50]. Camera library from the Android's Hardware library is used instead.

An important library for this kind of project is a library that can convert JSONs into POJOs. The most popular libraries (by stars on Github) are Gson, Moshi and Jackson. All three have pros and cons. They differ in terms of speed, library size and built-in adapters for different types in Kotlin. In terms of the speed of deserialization, Gson falls short, and Jackson is larger in term of library size than the other two [51].

With REST API in mind, Retrofit needs to be mentioned. It is a hugely popular type-safe HTTP client. It connects the application with the API by generating methods according to a given interface, and it cooperates (have prepared adapters) with the Moshi and the Gson, its underlying layer for HTTP communication is OkHttp library from the same developer as Moshi and Retrofit [52].

## 3.7 Android project structure

Android applications can consist of multiple modules. There are different types of modules: Android app module, dynamic feature module, library module and Google Cloud module.

Android app module consists of source files, resource files, and app level settings, module build files, and others. You can see a sample project structure in a Figure 3.1. It does not show all files and folders but suits as an example.

Android uses different folder endings to differ language variants [53]. Build types need to be configured in the Gradle folder, a sample can be seen in Listing 1.

The `main` folder is the `release` build type, and it contains the default files for the project. The other build type's source files overwrite the default ones when another build type is chosen. The developer chooses a build variant – build variant is a combination of build type and product flavour (explained in Section 5.4).

```
│ build.grade ...................................... Gradle configuration file
├─ proguard-rules.pro ......................... ProGuard configuration rules
└─ src ........................................... directory of source codes
   ├─ main ........................................... default build variant
   │  ├─ java ..................................... Java/Kotlin source codes
   │  └─ res ............................................... other resources
   │     ├─ drawable ..................... graphical data, vector and raster
   │     ├─ layouts .................................... layouts in XML
   │     ├─ values
   │     │  ├─ strings.xml ......................... default text resources
   │     │  ├─ strings-en.xml ...................... English text resources
   │     │  └─ colors.xml .............................. colour definitions
   │     └─ AndroidManifest.xml ..................... application manifest
   └─ debug ......................................... different build variant
```

Figure 3.1: Sample project structure

```
buildTypes {
    release {
        minifyEnabled true
        proguardFiles getDefaultProguardFile
                      ('proguard-android.txt'),
                      'proguard-rules.pro'
    }

    debug {
        applicationIdSuffix ".debug"
        minifyEnabled false
    }
}
```

Listing 1: Build type sample

Chapter 4

# Analysis and design

In this chapter, the new application design is described from both the user experience point of view but also from the technical point of view. The chapter starts with a definition of use cases and a data model. Then the application is designed including graphical wireframes and their validation. The application architecture and the multi-language architecture close the chapter.

## 4.1 Use cases

Slevomat Partner's goal is to simplify the process of voucher validation for the business partners of the company. It is designed to help them serve customers coming through the Slevomat platform. It needs to be able to connect the application with a partner account via a partner code. When the application is connected, it is listed in the list of devices at the website. After the onboarding, the app serves as a voucher scanner for most of its users.

Use case diagram in Figure 4.1 covers all use cases the application offers; it is followed by a subset of detailed use cases. In the diagram, we can see the Manager and the User as separate actors. The use cases are prepared for a permission setup, but it is a part of future development in the designs and development.

The use case is a written description of a sequence of actions for an actor (user, another application or time) when the actor wants to perform some task in the application. The use case starts with the goal of the actor and continues with the actions leading to accomplishing the goal. The last action should complete the goal of the use case. Use cases are a vital part of modern application development. Use cases can be in written in plain-text form, as a table or as a UML diagram.

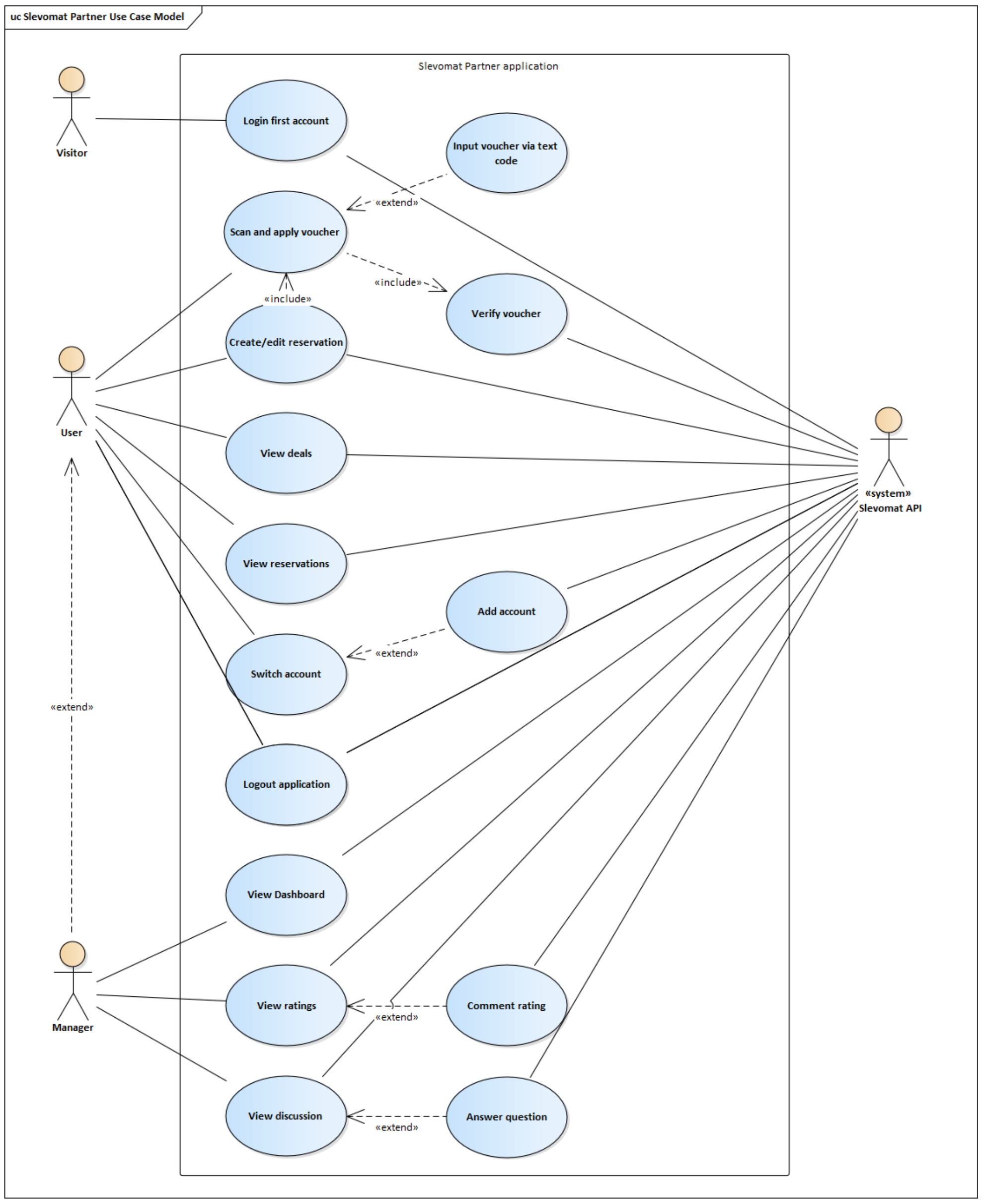


Figure 4.1: Use case diagram

| **Use Case #1** | **Login with the first account** |
|---|---|
| *Description* | Every new user needs to log into at least one account. |
| *Actors* | Visitor, API |
| *Preconditions* | Application in “fresh after installation” state, User is online. |

*Basic flow*

1. Application shows a welcome screen and explains its purpose
2. User taps Continue
3. Application asks for camera permission with explanation
4. User accepts it
5. Application shows QR code scanner with info where to search for partner code
6. User scans QR code from the website
7. Application asks for a name of the device
8. User fills in the name of the device or confirms the pre-generated one
9. Application sends partner code and name to the API
10. Application sends information about the device to the API
11. API receives information about the device being paired with a specific partner account from the application, sends back access token for the device
12. Application shows the user a success login screen, and the user can continue into the application

*Alternative flows*

3.a User does not grant the camera permission
   1. Application shows failure message; camera permission is mandatory
   2. Applications ends itself

7.a User inputs the partner code via text input
   1. User taps text input in the corner of QR code scanner
   2. User inputs the code in the input field and taps OK

9.a Invalid partner code
   1. API sends back information about the invalidity of the partner code
   2. Application shows a failure message
   3. User returns to step 5

| **Use Case #2** | **Validate and redeem a voucher** |
|---|---|
| *Description* | User wants to validate and redeem a voucher of a client. |
| *Actors* | User, API |
| *Preconditions* | User has at least one account logged in, User is online. |

*Basic flow*

1. (optional) User taps Scanner in the bottom bar
2. Application shows the Scanner screen
3. User aims the device at the QR code of the voucher
4. Application scans the voucher code and sends it to the API
5. API sends back voucher validity status and information about the voucher
6. Application shows a screen with voucher details and its validity
7. User taps a button to redeem the voucher
8. API receives the redeem request, redeems it and sends back OK status

*Alternative flows*

3.a User inputs the voucher code via text input
   1. User taps text input in the corner of QR code scanner
   2. User inputs the code in the input field and taps OK

6.a Invalid voucher
   1. Application shows that the voucher is invalid
   2. User taps Up/Back button
   3. User is taken back to the Scanner

| **Use Case #3** | **View deals** |
|---|---|
| *Description* | User sees running and past deals of an active account. |
| *Actors* | User, API |
| *Preconditions* | User has at least one account logged in, User is online. |

*Basic flow*

1. User taps Other in the bottom bar
2. Application shows a list of items
3. User taps Deals from a list of items
4. Application asks API for a list of deals for the specific partner account
5. Application shows a list of deals (running first, past second)

| **Use Case #4** | **Create a reservation from a text code** |
|---|---|
| *Description* | Create a reservation from a voucher text code received from the client. |
| *Actors* | User, API |
| *Preconditions* | User has at least one account logged in, User is online. |

*Basic flow*

1. User taps Reservation from the bottom bar
2. Application goes to the Reservation page
3. User taps Plus button
4. Application opens up a screen to insert a voucher code
5. User inputs the voucher code
6. Application sends the voucher code to the API to check status
7. API sends back information about the voucher – if it is available for reservation
8. Application shows form to fill in info about the reservation
9. User fills in the information about the reservation.
10. Application sends the data to the API
11. API received the information and checks it, if successful, it sends back confirmation
12. Application shows an animation of a successful reservation and shows a list of reservations

*Alternative flows*

7.a Voucher is not available for reservation

   1. Application shows Error that the voucher is not available for reservation
   2. Application returns the user to the code input screen

11.a Incorrect reservation info

   1. API returns error code
   2. Application shows the user which part of the information is incorrect
   3. User corrects the information and confirms
   4. Use case continues with step 10

| **Use Case #5** | **Adding another account** |
|---|---|
| *Description* | User adds another account in application with at least one account already logged in. |
| *Actors* | User, API |
| *Preconditions* | User has at least one account logged in, User is online. |

*Basic flow*

1. User taps Other in the bottom bar
2. Application shows a list of items
3. User taps Switch account from the list of items
4. Application shows a list of logged-in accounts
5. User taps the Add another account button
6. Application shows QR code scanner with info where to search for partner code
7. User scans QR code from the website
8. Application asks for the name of the device
9. User fills in the name of the device or confirms the pre-generated one
10. Application sends partner code and name to the API
11. Application sends information about the device to the API
12. API receives information about the device being paired with a specific partner account from the application, sends back access token for the device
13. Application shows an animation of success and shows the list of the accounts with the one added as the one logged in

*Alternative flows*

7.b User inputs the partner code via text input
1. User taps text input in the corner of QR code scanner
2. User inputs the code in the input field and taps OK

11.a Invalid partner code
1. Application shows a failure message
2. User returns to step 5

| **Use Case #6** | **React to a rating** |
|---|---|
| *Description* | User searches for new rating from a client and reacts to it. |
| *Actors* | User, API |
| *Preconditions* | User has at least one account logged in, User is online. |

*Basic flow*

1. User taps Other in the bottom bar
2. List of items is shown
3. User taps Ratings from list of items
4. Application asks API for a list of ratings
5. API sends back the list of ratings in JSON
6. List of rating is shown
7. User scrolls to rating he/she wants to react to
8. User taps the rating
9. Application shows details of the rating
10. User fills in the reaction and taps Save
11. Application sends the reaction to the API
12. API saves the reaction and publishes it on the website

*Alternative flows*

1.a User uses Rating card in Dashboard
    1. User taps Dashboard in the bottom bar
    2. User taps Rating card
    3. use case continues from Step 4

## 4.2 Data model

The data model is an abstraction of data organisation in the application. It shows how data relate to one another and how they are structured [54]. The Slevomat Partner application has its data model derived from the data model of API. It uses the data description diagram, sample seen in figure 4.2.

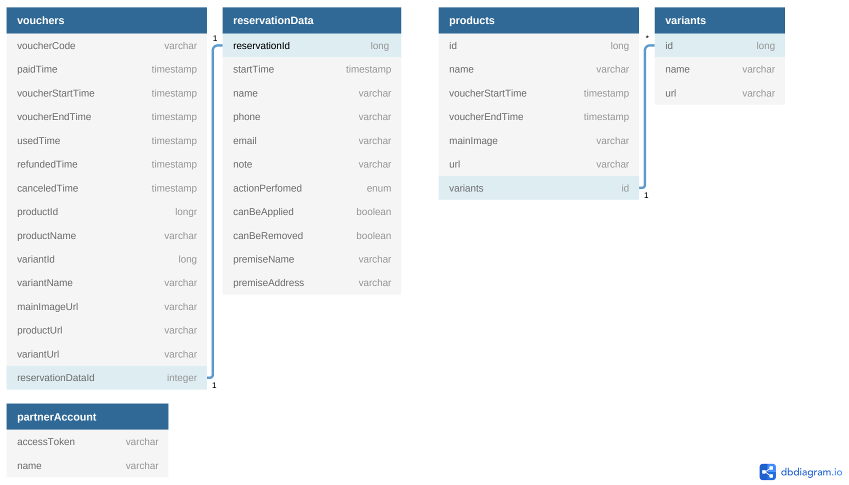


Figure 4.2: Sample data model diagram

## 4.3 User experience analysis

There are many definitions of what “user experience” with the product is. They can be summarised into how the user feels about the product when interacting with it.

One of the fundamentals of mobile application development is a good user experience. Users expect an intuitive and self-explanatory application; they do not have much patience with the application and need to be able to finish their goal without the need for a guide. This can be achieved with a thorough analysis of users’ needs.

The analysis started with going through the data logs of the old application. There are limitations with the old logging system. However, we could read that 92,5 % of screens in the application visited by the users were the Main Menu and screens directly connected with voucher scanning (the QR code scanner and scanning middle screen). Therefore; it is crucial to make this process as easy, fast and intuitive as possible. Other parts of the application were designed following Material guidelines for user interface design.

For good UX design, it is essential to follow basic rules when designing wireframes. One of the most important steps is to test the wireframes with real users after the design process; this is called validation. The validation of wireframes is described in the Validation section.

## 4.4 Wireframes

This is section shows examples of how the application looks and can be used. Wireframes for this application were created in a desktop application called Adobe XD [55]. Adobe XD is a vector-based design tool created by Adobe Inc. It can be used for designing user interfaces and prototyping mobile applications and websites. All of the designs were created with a resolution 640 × 360 px as this is the most wide-spread Android density-independent pixel (DP) resolution with aspect ratio of 16:9.

Application wireframes are more complex than the usual wireframes used in production. Lack of designers in the Slevomat company forced me to elevate my wireframes from simple ones to complete ones so they can also be used as a base for UI implementation.

### 4.4.1 Navigation

The application makes use of Material Bottom Navigation [56] as the primary navigation component. It displays five destinations next to each other and displays a text label for each. Badges are not part of the design, but the wireframe is prepared for their later addition. The navigation does not disappear when scrolling.

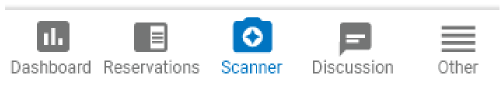
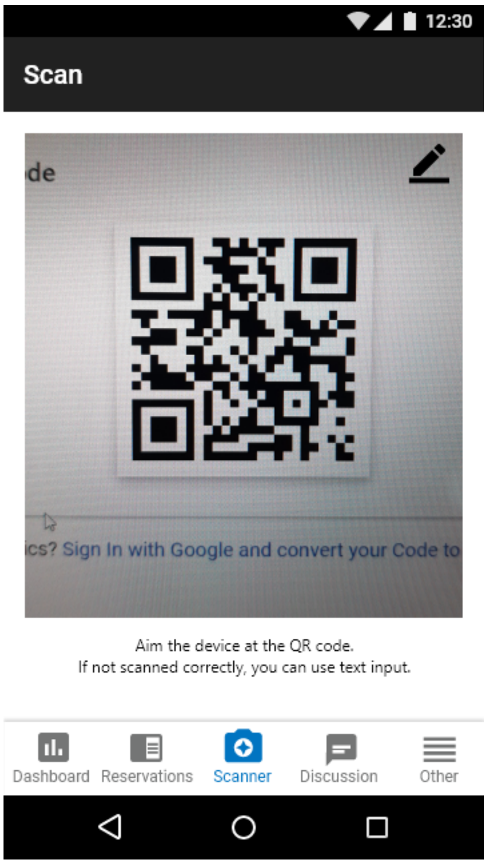


Figure 4.3: Full bottom navigation

### 4.4.2 QR code scanner

The scanner is an essential part of the application. As stated before, more than two-thirds of the app usage is linked to voucher validation and scanning. It was designed with speed in mind. When the user enters the Scanner screen, the voucher scanner automatically starts scanning. When it scans a voucher, it goes into the Voucher Detail screen. The user does not need to tap any button.

There is a possibility to input the code via text using the text input button in the corner of the QR scanner. When tapped, a dialogue box is shown with a text field.

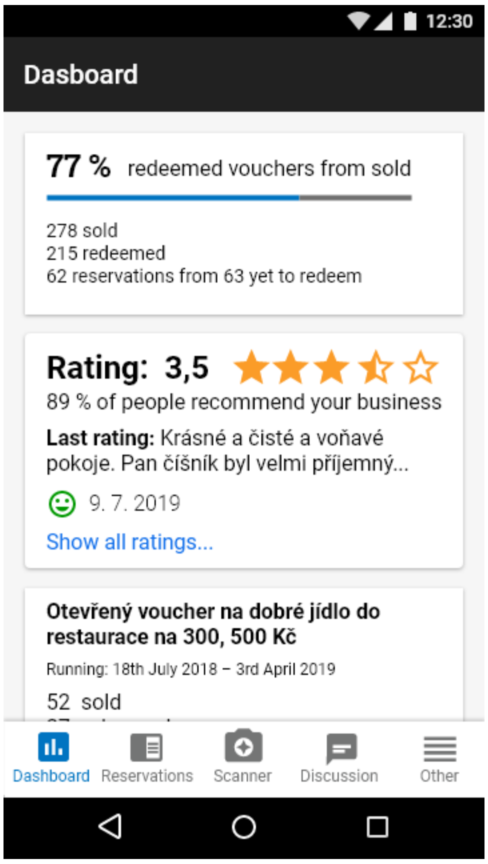


Aim the device at the QR code.
If not scanned correctly, you can use text input.

Figure 4.4: Scanner screen

### 4.4.3 Dashboard

A dashboard screen shows the user general information about their account. It shows percents of redeemed vouchers from all vouchers sold by the partner. It shows an overall rating of the partner, the last received rating and gives the possibility to tap-through into the Rating screen. The dashboard also shows a list of running deals with basic info about their performance (sold vouchers, redeemed).


12:30

Dasboard

77 % redeemed vouchers from sold

278 sold
215 redeemed
62 reservations from 63 yet to redeem

Rating: 3,5

89 % of people recommend your business

**Last rating:** Krásné a čisté a voňavé pokoje. Pan číšník byl velmi příjemný...

9. 7. 2019

Show all ratings...

**Otevřený voucher na dobré jídlo do restaurace na 300, 500 Kč**

Running: 18th July 2018 – 3rd April 2019

52 sold

Figure 4.5: Dashboard screen

### 4.4.4 Account switch

A new feature of the application is the ability to switch between accounts without being forced to log out of the other one. Account switcher has a straightforward design using simple radio buttons to switch accounts. The ability to log out of an account or register another account is always shown.

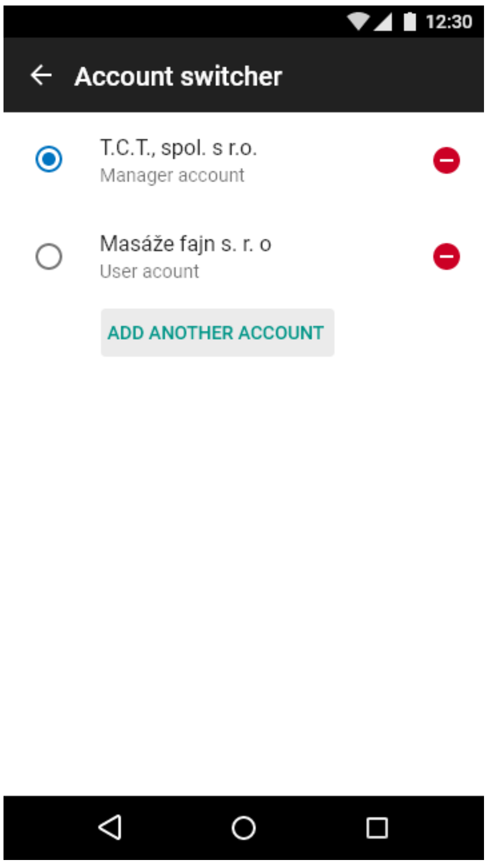
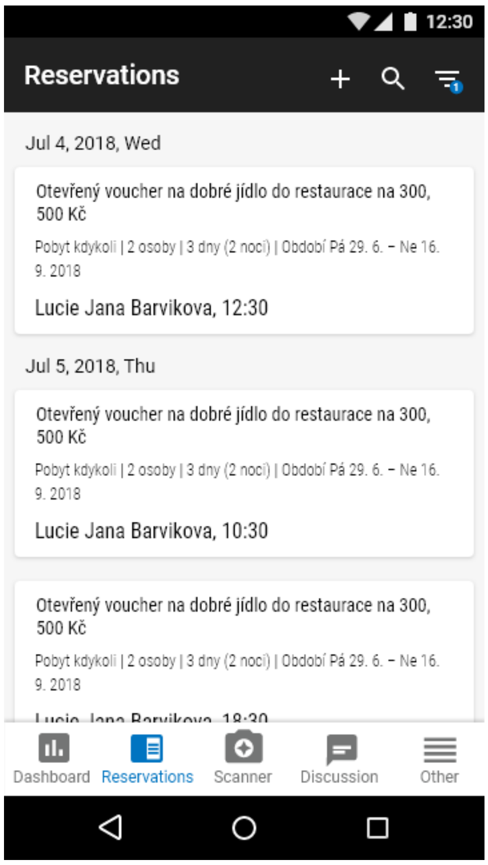


Figure 4.6: Account switcher screen

### 4.4.5 Reservations

Reservations are an essential part of restaurant and service agenda; therefore, it is necessary that they are easily accessible. As the application follows the Material Design guide, a list of reservations is created using card components. Each card contains basic information and can be tapped through to show reservation details with complete information.


Reservations

Jul 4, 2018, Wed

Otevřený voucher na dobré jídlo do restaurace na 300, 500 Kč

Pobyt kdykoli | 2 osoby | 3 dny (2 noci) | Období Pá 29. 6. – Ne 16. 9. 2018

Lucie Jana Barvikova, 12:30

Jul 5, 2018, Thu

Otevřený voucher na dobré jídlo do restaurace na 300, 500 Kč

Pobyt kdykoli | 2 osoby | 3 dny (2 noci) | Období Pá 29. 6. – Ne 16. 9. 2018

Lucie Jana Barvikova, 10:30

Otevřený voucher na dobré jídlo do restaurace na 300, 500 Kč

Pobyt kdykoli | 2 osoby | 3 dny (2 noci) | Období Pá 29. 6. – Ne 16. 9. 2018

Lucie Jana Barvikova, 18:30

Figure 4.7: List of reservations

## 4.5 User validation of the prototype

User validation tries to unite user expectations with the product design. In accordance with E. Petrášová [57], during validation, users are asked to do some task, and the researcher observes their actions.

The first task is to select the respondents. For this validation, two people were selected. It is less than the ideal number, but only two Slevomat partners accepted our offer. One of the partners never used the old version of the application, and the other uses it regularly.

The validation consisted of five tasks and a discussion afterwards:

- Onboarding process. Connect the newly installed application with a partner account.
- A client came to you with a voucher on his phone. Serve him.
- Find out how is your company rated on Slevomat.

- You have received an SMS from a client. He/She wants to create a reservation for a date specified in the SMS where is also the voucher code. Create a reservation for him/her.

- Colleague told you that your business received a review. You want to read it and answer.

Both validations came off successfully. The gathered feedback was turned into feature changes and was integrated into the application: Dashboard card click-through action was added to the whole surface of the card, voucher status in voucher detail changed to look less like a button.

The most visible change was the change of the Scanner button in the bottom bar. The original highlighted one by a circle was replaced with a classic menu item to make it look less like a floating-action button. With the original button, the responders tried to tap it multiple times even though they were already in the Scanner.

The afterwards discussion also confirmed that the Discussion should be placed in the bottom bar as it is. The other option was to put the Rating (hidden other tab Other) in the position of the Discussion and vice-versa. The discussion is more important for the users because clients ask questions before buying, and if they have to wait for the answer for too long, they buy a deal from a different business.

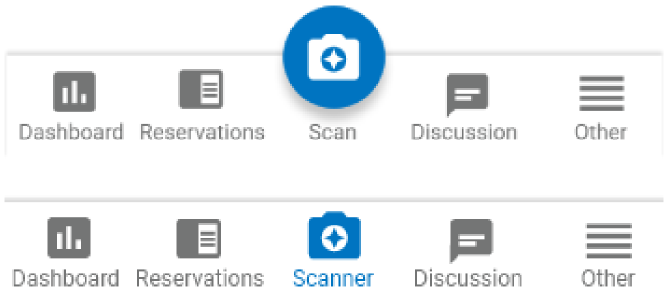


Figure 4.8: Bottom navigation comparison (top before, bottom after)

## 4.6 Application architecture

Application architecture tries to define how the classes are structured amongst each other. There are many different design patterns, but the three most popular are Model-View-Controller, Model-View-Presenter and Model-View-View model [58]. In the following paragraphs, the advantages and disadvantages of all of the three are summarised.

### Model-View-Controller (MVC)

In MVC, the model contains the application data and rules for data validity and the valid states of the data. The controller holds business logic, receives input from the user and sends requests/orders to the model and the view. The view contains classes and files representing UI, including XML files.

In reactive programming, the view can be an observer of the model and update itself accordingly [59]. The disadvantage of MVC is the size of the controller. The controller in Android is the Activity itself. The Activity is also the view in a way, and that makes its complexity rise quickly even with smaller applications.

### Model-View-Presenter (MVP)

The MVP is a derived pattern from the MVC, but the presenter replaces the controller. The critical difference between the MVC and MVP is that in the MVP, the view and the presenter are not tied to each other in any way and communicate via API. In the MVP, the view receives requests from the user and passes them to the presenter. The model holds the same data rules and data as in the MVC.

In Android, the View is Activity as well as it is in MVC, but the presenter is a stand-alone class. That makes the complexity of the classes and Activities reasonable. The disadvantage of MVP is that the count of the classes and interfaces grows abruptly [60].

### Model-View-View model (MVVM)

The MVVM pattern tries to use binding to replace the controller or presenter. It two-way binds data between the view and the view model. The view model exposes actions and data that can be called and observed by the view. It also interacts with the model to persist the data. The view binds itself to the view model, but the view model should not know about the view. Many views can be bound to one view model.

A concern with the MVVM is that the developer puts presentation logic in the view as he/she is often used to, but it should not be there. The view should be bound directly to presentable data which the view model holds [61].

General advantages of the MVVM are that interfaces are not needed as the view are updated through the binding and that the view model is not tied to the view in any way. In the Android OS, there is the extra advantage of Google support. Many classes that the developer does need for the MVVM architecture are included in the Jetpack libraries.

## Conclusion

For this application, the MVVM pattern was chosen. It is pattern recommended by Google, and for the MVVM Google introduced the Android Jetpack package, which includes libraries for the lifecycle-aware view model, the LiveData data holder, and the declarative data binding library, which helps the developers easily create MVVM architecture.

Chapter 5

# Realisation

This chapter describes the process of the software implementation of the application. It consists of tools selection, libraries used and description of an implementation of interesting parts of the application – what problems occurred and how they were solved. Analysis and design from previous chapters were used.

## 5.1 Used tools

It was easy to choose IDE because Android Studio (version 3.5 in time of development) is the official application for Android development. It is developed by JetBrains and Google and is based on IntelliJ IDEA Community IDE.

The second option would be using IntelliJ IDEA Community directly, but it doesn't provide any advantages for Android development. The third option is using Eclipse IDE with plugins for Android, but that would suit a long-term Eclipse user, not me as I am used to JetBrains' IDEs.

Git was used for project version control. It supports offline work, is better at branching the code and mainly as the complete project is downloaded to the local machine, the local builds used during the project are possible. *GitHub* is a code repository used by Slevomat company for all of its projects; therefore, it was used for this project too. Android Studio has integrated Git support, which easily connects with GitHub.

## 5.2 Libraries used in the project

All of the project dependencies, including libraries, are organised in a file called `Dependencies.kt`. It is a Kotlin file in `buildSrc` package that is accessible from Gradle. The list of dependencies separates names and versions of libraries which makes maintaining the dependencies more easier. Sample can be seen in Listing 2.

In this project, a few libraries are used. The most important one is the already mentioned combination of Retrofit [52], Moshi [62] and OkHttp libraries which provide API communication (Retrofit) and convert JSONs into Kotlin objects (Moshi). These are the key libraries for the project as it is heavily dependent on its API. All of these three libraries come from the Square company that keeps updating its libraries for years now and is seen as a dependable developer.

The Koin library is another library used in the project. It is “pragmatic lightweight dependency inejction framework for Kotlin” according the Koin website [63]. It is an open-source framework that provides DI using Kotlin domain-specific language. During the building process, it generates code needed for the object to be provided where injected. I chose Koin over Dagger, another dependency injection library, for its ease of use while maintaining the same functionalities for a project of this size.

ML Kit for Firebase library from Google is another library used in the project. It is a machine learning library that provides the barcode scanning functionality for the application. The ML Kit library contains other modules – facial recognition, text recognition, image labelling [64]. There are many alternatives to ML Kit library in terms of QR code scanning, but there are none that are still in active development and are backed by a trustworthy developer. ZXing library used by many is publicly in maintenance mode, and others are not updated for months or years without official notice.

The Jetpack library set contains libraries that help with the MVVM development – namely the Architecture components. The ViewModel and the LiveData library are used throughout the whole project. An alternative for the LiveData is RxJava, but the project is developed in Kotlin; therefore, the Kotlin option (LiveData) is used.

```
object Libs {
  const val retrofit =
    "com.squareup.retrofit2:retrofit:${LibsVersion.retrofit}"
  const val retrofitMoshiConverter =
    "com.squareup.retrofit2:converter-moshi:${LibsVersion.retrofit}"
  const val moshi =
     "com.squareup.moshi:moshi:${LibsVersion.moshi}"
}

object LibsVersion {
  const val retrofit = "2.6.2"
  const val moshi = "1.6.0"
}
```

Listing 2: Sample from `Dependencies.kt` file

## 5.3 Project setup

The project was started with Bottom navigation activity template. It automatically prepares Main Activity with 3 different Fragments and adds Bottom navigation component. Gradle is used as the default build tool, Gradle Groovy DSL and Gradle Kotlin DSL are combined throughout the project.

Application is very much dependent on the API connection; therefore, the first thing was to configure the Retrofit library to connect the application to given API endpoints. Retrofit needs interface and objects for requests and responses. It also supports query parameters using a map, in Listing 3 is one sample method and its request, and response object. It requests information about a voucher from the Slevomat REST API using the POST method.

## 5.4 Multi-language setup

One of the non-functional requirements is to prepare the Slevomat Partner application for creating different language mutations, such as Zľavomat Partner is used for a Slovak market. The application uses the same code for the application logic, but it has different language resources, it is connected to different API, and it can have differences in logic when needed.

Basic multi-language setup is simple in Android. The developer can set up a project folder structure that has one default resource file or folder while others are optional and are only used when the device locale is set to one matching the folder/file name ending.

This application has more requirements than just different text resources. It needs a different API connection, a different application name, a different icon, and some functionalities are limited. These can be achieved using flavours.

Flavours work similarly to build types – they differ code used when building the application. Build type specifies how the application is built, what certificates are included and other build-related settings. In contrary, flavours are used for creating a different version in terms of functionalities. They are used when creating free and paid versions of the application, and in this project, they are used to differ Slevomat and Zľavomat Partner application. The flavour setting of the project can be seen in Listing 4.

Every flavour needs to have dimension specified. There can be more than one dimension in one application, but each flavour can only belong to one dimension [65].

## 5.5 User interfaces

Android layouts have a poser for designers and developers that Android devices come with many different screen sizes and resolutions. The layout

```
@POST("voucher-info")
fun getVoucherInfo(@Body voucherRequest: VoucherRequest):
    LiveData<ApiResponse<VoucherResponse>>

data class VoucherRequest(val code: String)

@JsonClass(generateAdapter = true)
data class VoucherResponse(
    @Json(name = "code") var voucherCode: String,
    val paidTime: String,
    val voucherStartTime: OffsetDateTime,
    val voucherEndTime: OffsetDateTime,
    val usedTime: OffsetDateTime?,
    val refundedTime: OffsetDateTime?,
    val canceledTime: OffsetDateTime?,
    val serverTime: OffsetDateTime,
    val canBeApplied: Boolean,
    val voucherUsingClosed: Boolean,
    val productId: Int,
    val productName: String,
    val variantId: Int,
    val variantName: String,
    val mainImageUrl: String,
    val productUrl: String,
    val variantUrl: String,
    val voucherUsageMethod: String?,
    val usagePremiseName: String?,
    val usagePremiseAddress: String?,
    val reservationData: ReservationDataItem?
)
```

Listing 3: Sample of method definition from API interface

implementation needs to be prepared for different aspects of the screen. It is important to mention that Android uses density points (DPs) instead of pixels. 1 DP is calculated from the pixel density of the screen with formula: $dp = (width\,in\,pixels * 160)\,/\,screen\,density$.

This application uses a combination of Linear Layouts and Constraint Layouts. The Voucher Detail layout uses both layouts, but the Linear Layout only aligns the views next to each other vertically or horizontally. The example shows how the constraints are defined.

The layout contains a thumbnail of the voucher product and information about it below. It also holds a button area with different background colour and two buttons and text about the voucher performance. The design can be

```
flavorDimensions "language"
productFlavors {
    cz {
        dimension "language"
        applicationId "cz.slevomat.partner"
    }
    sk {
        dimension "language"
        applicationId "sk.zlavomat.partner"
        manifestPlaceholders = [appName: SK_APP_NAME]
    }
}
```

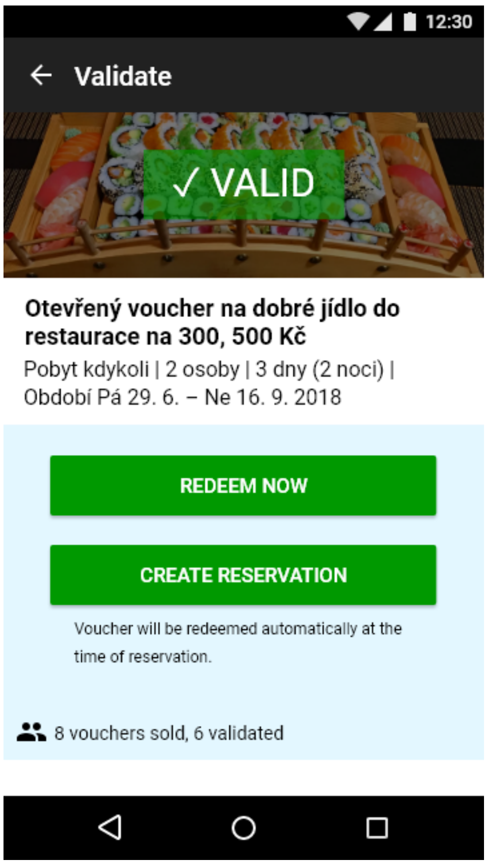
Listing 4: Slevomat Partner's flavour settings

seen in the Figure 5.1 and the code in Listing 5. For a moderate length of the thesis, some attributes were omitted in the sample.

## 5.6 QR code scanner

The crucial implementation problem with the QR code scanner is that it needed to work in a Fragment that can be included in different parts of the application without the need to start new Activity, for example, the Main Activity holds only a Fragment of the activity. The QR code scanner is based on the ML Kit Showcase application, but there is the scanner used in an Activity and has many features not needed for the scanning. The Scanner uses Camera library from Android and FirebaseVision package from the ML Kit.

The scanner starts an infinite loop when the Scanner fragment is active. It analyses every frame and tries to find a QR code in the picture. If a QR code is found, it is returned into the view model. Although the infinite loop might seem like an inappropriate approach, the Android device turns the display off when inactive, thus breaking the loop.



Otevřený voucher na dobré jídlo do restaurace na 300, 500 Kč

Pobyt kdykoli | 2 osoby | 3 dny (2 noci) | Období Pá 29. 6. – Ne 16. 9. 2018

Voucher will be redeemed automatically at the time of reservation.

8 vouchers sold, 6 validated

Figure 5.1: Voucher detail for a valid voucher design

```xml
<androidx.constraintlayout.widget.ConstraintLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="@color/lightBlue">

    <Button
        android:id="@+id/btnValidate"
        ...
        android:layout_marginTop="@dimen/offset_base"
        android:layout_marginBottom="@dimen/offset_small"
        app:layout_constraintBottom_toTopOf="@+id/btnCreateReservation"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/btnCreateReservation"
        ...
        android:layout_marginTop="@dimen/offset_small"
        android:layout_marginBottom="@dimen/offset_small"
        app:layout_constraintBottom_toTopOf="@id/tvReservationHelp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/btnValidate" />

    <TextView
        android:id="@+id/tvReservationHelp"
        android:layout_width="300dp"
        android:layout_height="wrap_content"
        android:layout_marginTop="@dimen/offset_small"
        android:layout_marginBottom="@dimen/offset_small"
        android:text="@string/reservation_help_text"
        app:layout_constraintBottom_toTopOf="@id/tvVoucherStats"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/btnCreateReservation" />

    <TextView
        android:id="@+id/tvVoucherStats"
        ... />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Listing 5: Constraint Layout sample

Chapter 6

# Testing

In a book *The Art of Software Testing* from G. J. Myers and C. Sandler [66] they say: “Software testing is a process, or a series of processes, designed to make sure computer code does what it was designed to and, conversely, that it does not do anything unintended.”

There are different views on how to separate software testing into groups. Based on approach (static/dynamic, exploratory, white-box/black-box), level of testing (unit, integration, system, acceptance) or type of test (e.g., smoke, regression, functional/non-functional, destructive, security). In this chapter, the view is narrowed to mobile application testing and also is specified how the new Slevomat Partner application is tested [67].

## 6.1 Separating release and test versions

The release version is different from the test versions. In this project, two test versions are prepared: The Debug version and the Alpha version. Their differences can be seen in the Gradle file sample in Listing 6.

The Debug version is set up to have the shortest build time and to retain the debug symbols. Its purpose is for the developer to quickly see the code running when changing the code and to be able to see the debug information when needed. If this version would be released, an advanced user could decompile the application and exploit the code.

The Alpha version is as close to the release version as possible. It is not debuggable, it is obfuscated, and its resources are shrunk. This process takes time and can cause bugs in the application. The advantage is the similarity to the release version – the same process is done during the build of the release version of the application. For the testing (unit, automated and usability) the Alpha version is used.

```
buildTypes {
    release {
        minifyEnabled(true)
        shrinkResources(true)
        debuggable(false)
        productFlavors.cz.signingConfig signingConfigs.releaseCZ
        productFlavors.sk.signingConfig signingConfigs.releaseSK
        proguardFiles(getDefaultProguardFile(
            "proguard-android-optimize.txt"),
            "proguard-rules.pro")
    }

    alpha {
        applicationIdSuffix(".alpha")
        minifyEnabled(true)
        shrinkResources(true)
        debuggable(false)
        proguardFiles(getDefaultProguardFile(
            "proguard-android-optimize.txt"),
            "proguard-rules.pro")
        matchingFallbacks = ['release']
        signingConfig signingConfigs.debug
    }

    debug {
        applicationIdSuffix(".dev")
        minifyEnabled(false)
        shrinkResources(false)
        debuggable(true)
        signingConfig signingConfigs.debug
    }
}
```

Listing 6: Separating release and test versions via Gradle

## 6.2 Unit testing

When a small unit of software (module, class) is tested, it is called unit testing. Unit tests for Android do not differ from other software unit tests.

Android SDK includes an extension to the JUnit framework and its android.test package includes testing tools specifically designed for Android use. The package includes single activity testing, utilities for generating events like a touch or services testing or an extended set of asserts (e.g., assignables, regular expression) [68].

The amount of unit test in the application is minimal because of the application state. It is also caused by the limited amount of logic included in the application.

## 6.3 Automated UI testing

Automated UI tests are part of instrumented tests. They require an instance of Android system, either on a physical device or an AVD (Android Virtual Device). Automated UI tests simulate input of the user to the application and check if its behaviour is correct.

Google developed the Espresso framework for Android that is relying on the mentioned JUnit framework. It provides interfaces for writing automated UI tests. It uses the grey-box testing approach as the test designer needs to know how the view tree is built. The framework is capable of checking the status of activities, whether they are loading or are in a ready state [69].

The application is not covered by automated UI tests as the user interface is still in the development state and will change throughout the app finishing.

## 6.4 Usability testing

Usability testing or user testing is a tool to evaluate mobile application with real users. The most common pattern is giving the users tasks to complete. In combination with thinking aloud, it can give the researcher much information on how the users feel and think about the application. If there are any problems, and if they are, how severe they are. They can be divided into high (failure in task execution), medium (task can be executed with difficulties) and low (minor problems) [70].

The user testing is designed from the base of the wireframe validation. It has the same 5 tasks covering most of the screens of the application and 1 extra task – editing existing answer to a question in a discussion. These six tasks cover all important parts of the application. The user change is a minor use case in terms of quantity of users, but it is important for the few that need it. That is why it is not tested with general users.

Though the testing was not carried out yet, it is prepared for the execution. It is important to use the Alpha version of the application. If the release version would be used, the information would be sent to the real API including payments.

# Conclusion

The main goal of a “Slevomat Partner – Android application” thesis was to design and implement a new version of the application. The goal also included the analysis of the current version of the application and conducting a market recherche of mobile applications with a similar purpose.

The goal of the analysis of the current application was fulfilled. Problems of the current application were pointed out, and they were taken into account during the design. The thesis also contains an analysis of other applications in the market; thus, the analysis goal was completed.

The crucial part of this thesis is the design of the new application. The original goal was to design wireframes only, from which a designer should have designed the user interface in the details. The Slevomat company failed to provide the designs. Therefore, the goal was extended to include creating the whole UI with all the details. The extended goal was accomplished, and real users of the application validated the application design.

The extended design goal created a significant delay in the implementation goal which lead to a incomplete implementation goal. Albeit the application is not yet ready for production, the project has a solid foundation. My work on the application will continue outside the scope of the thesis.

# Bibliography

[1] Holst, A. Global smartphone penetration rate as share of population from 2016 to 2020 [online]. Available from: `https://www.statista.com/statistics/203734/global-smartphone-penetration-per-capita-since-2005/`

[2] Guide to background processing | Android Developers. 2019. Available from: `https://developer.android.com/guide/background`

[3] Mobile Operating System Market Share Worldwide. 2019. Available from: `http://gs.statcounter.com/os-market-share/mobile/worldwide/`

[4] Stieglitz, S.; Lattemann, C.; et al. Mobile Applications for Knowledge Workers and Field Workers. *Mobile Information Systems*, volume 2015, no. 2015, 2015: pp. 1–8, ISSN 1574-017X, doi:10.1155/2015/372315. Available from: `http://www.hindawi.com/journals/misy/2015/372315/`

[5] Bartoňková, E. Rozšiřitelná mobilní herní aplikace. 2019.

[6] Xamarin.Forms | .NET. 2019. Available from: `https://dotnet.microsoft.com/apps/xamarin/xamarin-forms`

[7] Tillu, J. Flutter Compilation Process [online]. 2019. Available from: `https://dev.to/jay_tillu/flutter-compilation-process-41k0`

[8] Dunn, C.; Schonning, N.; et al. Part 1 – Understanding the Xamarin Mobile Platform - Xamarin | Microsoft Docs [online]. Available from: `https://docs.microsoft.com/en-us/xamarin/cross-platform/app-fundamentals/building-cross-platform-applications/understanding-the-xamarin-mobile-platform`

[9] Terebetska, A. Flutter vs Xamarin: The Complete 2019 Developer's Guide [Infographics Included] [online]. Available from:

https://apiko.com/blog/flutter-vs-xamarin-the-complete-2019-developers-guide-infographics-included/

[10] Bartík, M. Proč a jak psát progresivní webové aplikace [online]. 2017. Available from: https://www.ackee.cz/blog/proc-a-jak-psat-progresivni-webove-aplikace/

[11] GoOut Scanner [software]. 2019. Available from: https://play.google.com/store/apps/details?id=net.goout.scanner

[12] RIS [software]. 2018. Available from: https://play.google.com/store/apps/details?id=com.restu.ris

[13] Groupon Merchants [software]. 2019. Available from: https://play.google.com/store/apps/details?id=com.groupon.redemption

[14] Qerko [software]. 2019. Available from: https://play.google.com/store/apps/details?id=com.itispaid

[15] Robertson, A. The Supreme Court will hear Google and Oracle's nearly decade-long copyright fight [online]. 2019. Available from: https://www.theverge.com/2019/11/15/20946398/oracle-google-java-copyright-lawsuit-trial-supreme-court-request

[16] Android Open Source Project. 2019. Available from: https://source.android.com/

[17] Brahler, S. Analysis of the Android Architecture. 2010.

[18] Frequently Asked Questions | Android Open Source Project. 2019. Available from: https://source.android.com/setup/start/faqs

[19] Activity | Android Developers. 2019. Available from: https://developer.android.com/reference/android/app/Activity

[20] Services | Android Developers. 2019. Available from: https://developer.android.com/guide/components/services

[21] Farrel, S. Choosing the Right Background Scheduler in Android [online]. 2016. Available from: https://www.bignerdranch.com/blog/choosing-the-right-background-scheduler-in-android/

[22] Broadcast overview | Android Developers. 2019. Available from: https://developer.android.com/guide/components/broadcasts

[23] BroadcastReceiver | Android Developers. 2019. Available from: https://developer.android.com/reference/android/content/BroadcastReceiver

[24] Intent | Android Developers. 2019. Available from: `https://developer.android.com/reference/android/content/Intent`

[25] Content Providers | Android Developers. 2019. Available from: `https://developer.android.com/guide/topics/providers/content-providers`

[26] Fragments | Android Developers. 2019. Available from: `https://developer.android.com/guide/components/fragments`

[27] Layouts | Android Developers. 2019. Available from: `https://developer.android.com/guide/topics/ui/declaring-layout`

[28] View | Android Developers. 2019. Available from: `https://developer.android.com/reference/android/view/View`

[29] TextView | Android Developers. 2019. Available from: `https://developer.android.com/reference/android/widget/TextView`

[30] EditView | Android Developers. 2019. Available from: `https://developer.android.com/reference/android/widget/EditText`

[31] Button | Android Developers. 2019. Available from: `https://developer.android.com/reference/android/widget/Button`

[32] TextView | Android Developers. 2019. Available from: `https://developer.android.com/reference/android/widget/ImageView`

[33] ViewGroup | Android Developers. 2019. Available from: `https://developer.android.com/reference/android/view/ViewGroup`

[34] FrameLayout | Android Developers. 2019. Available from: `https://developer.android.com/reference/android/widget/FrameLayout`

[35] LinearLayout | Android Developers. 2019. Available from: `https://developer.android.com/reference/android/widget/LinearLayout`

[36] ConstraintLayout | Android Developers. 2019. Available from: `https://developer.android.com/reference/android/support/constraint/ConstraintLayout`

[37] RecyclerView | Android Developers. 2019. Available from: `https://developer.android.com/reference/androidx/recyclerview/widget/RecyclerView`

[38] Gradle vs Maven Comparison. 2017. Available from: `https://gradle.org/maven-vs-gradle/`

[39] Gradle Kotlin DSL 1.0 RC3 Release Notes. 2018. Available from: https://github.com/gradle/kotlin-dsl-samples/releases/tag/v1.0-RC

[40] Beams, C. Kotlin Meets Gradle [online]. 2016. Available from: https://blog.gradle.org/kotlin-meets-gradle/

[41] Gradle Kotlin DSL Primer. 2018. Available from: https://docs.gradle.org/current/userguide/kotlin_dsl.html

[42] Binstock, A. Java's 20 Years Of Innovations [online]. 2015. Available from: https://www.forbes.com/sites/oracle/2015/05/20/javas-20-years-of-innovation/

[43] Williams, M. Java vs. Kotlin - The No-nonsense Comparison of Android Programming Languages [online]. 2019. Available from: https://www.promptbytes.com/blog/java-vs-kotlin-the-no-nonsense-comparison-of-android-programming-languages

[44] Kotlin Foundation. 2018. Available from: https://kotlinlang.org/foundation/kotlin-foundation.html

[45] Kukla, G. Differences between .class and .dex files in Java & Android [online]. Available from: https://www.boldare.com/blog/differences-between-class-and-dex-files-in-java-android/

[46] Lardinois, F. Kotlin is now Google's preferred language for Android app development [online]. 2019. Available from: https://techcrunch.com/2019/05/07/kotlin-is-now-googles-preferred-language-for-android-app-development/

[47] Team, A. E. Kotlin vs Java: What is Better for Android Development? [online]. 2017. Available from: https://applikeysolutions.com/blog/kotlin-vs-java-what-is-better-for-android-development

[48] What is Android SDK? Available from: https://www.techopedia.com/definition/4220/android-sdk

[49] What is a Software Library? Available from: https://www.techopedia.com/definition/3828/software-library

[50] Android Jetpack | Android Developers. 2019. Available from: https://developer.android.com/jetpack

[51] Damsky, D. Here's why you probably shouldn't be using the Gson library in 2018 [online]. 2018. Available from: https://medium.com/@dannydamsky99/heres-why-you-probably-shouldn-t-be-using-the-gson-library-in-2018-4bed5698b78b

[52] Retrofit. 2013. Available from: `https://square.github.io/retrofit/`

[53] Limantara, M. Overview of Android Project Structure and Naming Conventions [online]. 2016. Available from: `https://medium.com/@mikelimantara/overview-of-android-project-structure-and-naming-conventions-b08f6d0b7291`

[54] Brodie, M. L. On the development of data models. In *On conceptual modelling*, Springer, 1984, pp. 19–47.

[55] Adobe XD [software]. Available from: `https://www.adobe.com/cz/products/xd.html`

[56] Bottom Navigation - Material Design. 2019. Available from: `https://material.io/components/bottom-navigation/`

[57] Petrášová, E. Uživatelské testování #1 [online]. 2017. Available from: `https://medium.com/house-of-rezac/uzivatelske-testovani-1-fb4ad9b5de6e`

[58] Sinhal, A. MVC, MVP and MVVM Design Pattern [online]. 2017. Available from: `https://medium.com/@ankit.sinhal/mvc-mvp-and-mvvm-design-pattern-6e169567bbad`

[59] Davis, I. What Are The Benefits of MVC? [online]. 2008. Available from: `https://blog.iandavis.com/2008/12/what-are-the-benefits-of-mvc/`

[60] Ramsdale, C. Building MVP apps: MVP Part I [online]. 2010. Available from: `http://www.gwtproject.org/articles/mvp-architecture.html`

[61] Guide to app architecture | Android Developers. 2019. Available from: `https://developer.android.com/jetpack/docs/guide`

[62] square/moshi: A modern JSON library for Kotlin and Java. 2013. Available from: `https://github.com/square/moshi`

[63] insert-koin.io · a smart Kotlin dependency injection framework. 2019. Available from: `https://insert-koin.io/`

[64] ML Kit for Firebase | Firebase. 2019. Available from: `https://firebase.google.com/docs/ml-kit`

[65] Configure build variants | Android Developers. 2019. Available from: `https://developer.android.com/studio/build/build-variants`

[66] Myers, G. J.; Sandler, C.; et al. *The art of software testing*. John Wiley & Sons, 2011.

[67] Everett, G. D.; McLeod, R., Jr. *Software Testing: Testing Across the Entire Software Development Life Cycle*. John Wiley & Sons, 2007.

[68] Annuzzi Jr., J.; Darcey, L.; et al. *Advanced Android Application Development*. Addison-Wesley Professional, 2014.

[69] Coppola, R.; Raffero, E.; et al. Automated Mobile UI Test Fragility: An Exploratory Assessment Study on Android. In *Proceedings of the 2Nd International Workshop on User Interface Test Automation*, INTUITEST 2016, New York, NY, USA: ACM, 2016, ISBN 978-1-4503-4412-8, pp. 11–20, doi:10.1145/2945404.2945406. Available from: `http://doi.acm.org/10.1145/2945404.2945406`

[70] Kaikkonen, A.; Kekäläinen, A.; et al. Usability Testing of Mobile Applications: A Comparison Between Laboratory and Field Testing. *J. Usability Studies*, volume 1, no. 1, Nov. 2005: pp. 4–16, ISSN 1931-3357. Available from: `http://dl.acm.org/citation.cfm?id=2835525.2835527`

Appendix A

# Acronyms

| | |
|---|---|
| **GUI** | Graphical User Interface |
| **XML** | Extensible Markup Language |
| **UX** | User Experience |
| **UI** | User Interface |
| **API** | Application Programming Interface |
| **REST** | Representational State Transfer |
| **PWA** | Progressive Web Application |
| **SDK** | Software Development Kit |
| **NDK** | Native Development Kit |
| **MVC** | Model-View-Controller |
| **MVP** | Model-View-Presenter |
| **MVVM** | Model-View-ViewModel |
| **ART** | Android Runtime |
| **VM** | Virtual machine |
| **DSL** | Domain specific language |
| **UML** | Unified Modeling Language |

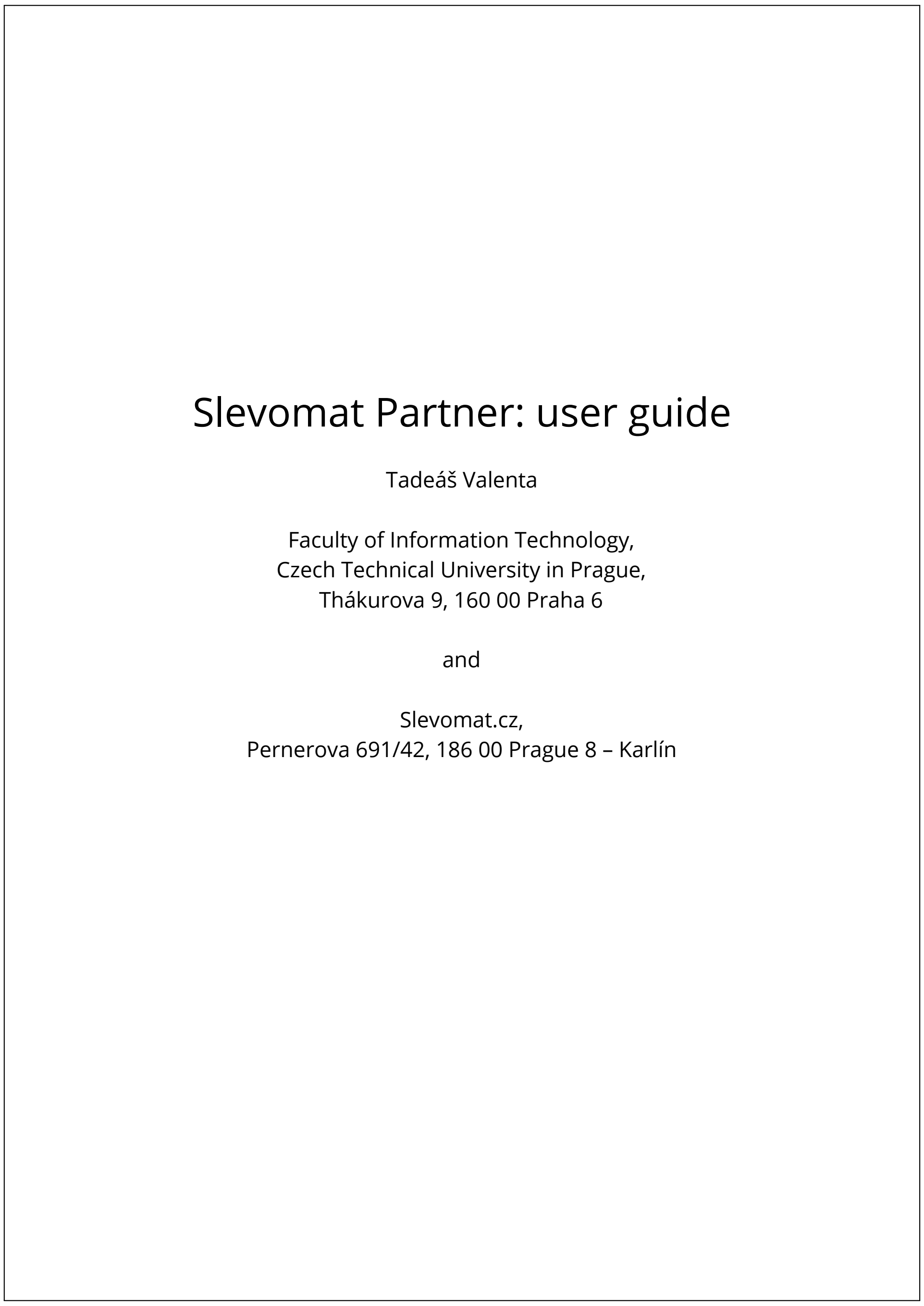
Appendix B

# Contents of enclosed USB drive

```
readme.txt ..................... file with USB drive contents description
src ........................................ directory of source codes
  thesis .................. directory of LaTeX source codes of the thesis
SlevomatPartnerUI.xd ..................... wireframe design source file
SlevomatPartnerUI.pdf ....................... exported wireframe design
text .............................................. thesis text directory
  thesis.pdf ............................... thesis text in PDF format
```

Appendix C

# User guide



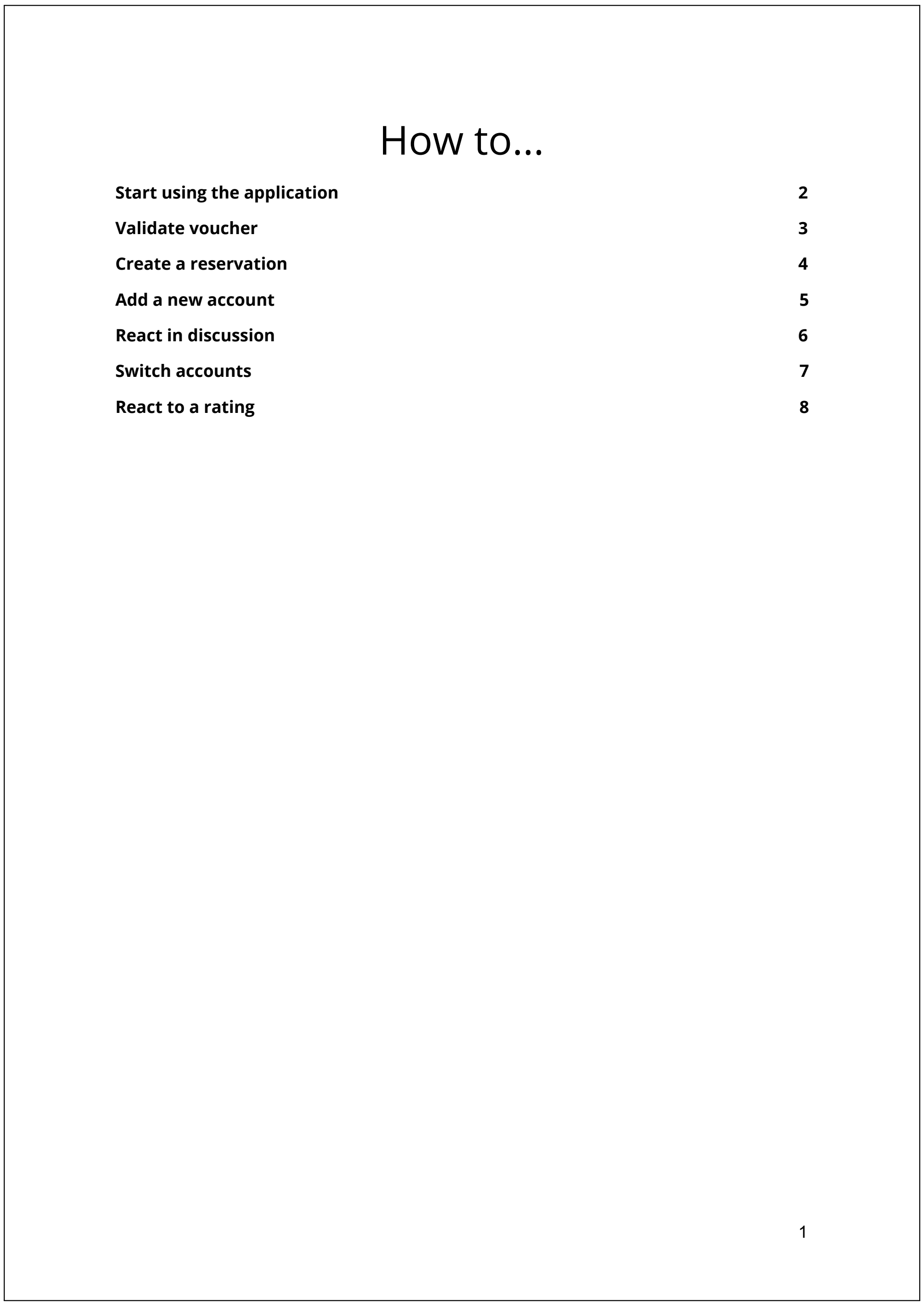
# Slevomat Partner: user guide

Tadeáš Valenta

Faculty of Information Technology,
Czech Technical University in Prague,
Thákurova 9, 160 00 Praha 6

and

Slevomat.cz,
Pernerova 691/42, 186 00 Prague 8 – Karlín

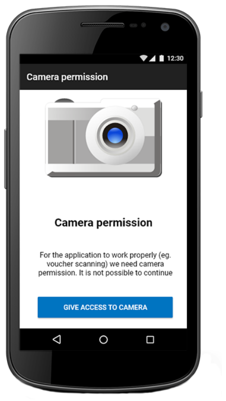


## How to...

1

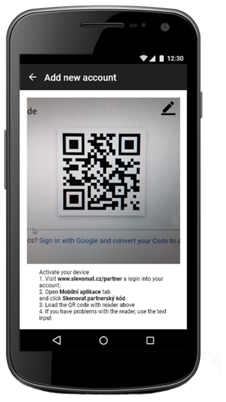
## Start using the application

1. Go to the link https://play.google.com/store/apps/details?id=cz.slevomat.partner and tap Install. This will install the application into your phone. If you have the application already installed, you will see an Uninstall button.
2. Find the application on your phone and open it.
3. Read the welcome message
4. Give the application the camera access (it is used for voucher scanning only)



For the application to work properly (eg. voucher scanning) we need camera permission. It is not possible to continue

5. Open administration of your partner account: www.slevomat.cz/partner
6. Go to tab **Mobile application**
7. Click **Add a new device**
8. Scan the code with the application. Or tap the Pen button to type in the code.
9. Fill in the name for your device or use the generated one



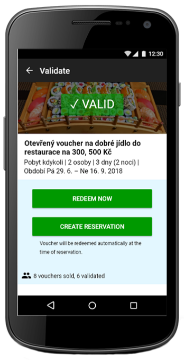
Activate your device
1. Visit **www.slevomat.cz/partner** a login into your account.
2. Open **Mobilní aplikace** tab
and click **Skenovat partnerský kód**
3. Load the QR code with reader above
4. If you have problems with the reader, use the text input

2

## Validate voucher

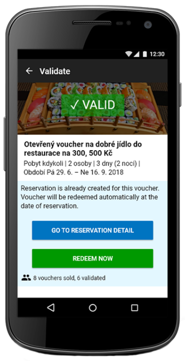
1. Tap the Scanner in the bottom bar
2. Aim the device at the QR code of the voucher you want to scan
3. Voucher status is automatically shown
4. Available action options are shown in the detail. You can always go back with the Back button, it will not cause any changes to the voucher
5. Below you can see a subset of voucher states that might occur

1) Voucher is valid and it has no reservations created



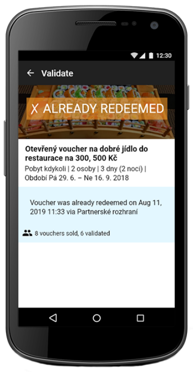
12:30
Validate
✓ VALID
**Otevřený voucher na dobré jídlo do restaurace na 300, 500 Kč**
Pobyt kdykoli | 2 osoby | 3 dny (2 noci) | Období Pá 29. 6. – Ne 16. 9. 2018
REDEEM NOW
CREATE RESERVATION
Voucher will be redeemed automatically at the time of reservation.
8 vouchers sold, 6 validated

2) Voucher is valid and it already has reservation created



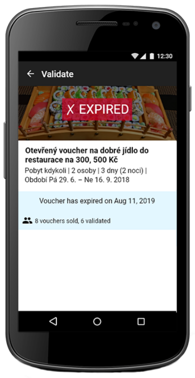
12:30
Validate
✓ VALID
**Otevřený voucher na dobré jídlo do restaurace na 300, 500 Kč**
Pobyt kdykoli | 2 osoby | 3 dny (2 noci) | Období Pá 29. 6. – Ne 16. 9. 2018
Reservation is already created for this voucher. Voucher will be redeemed automatically at the date of reservation.
GO TO RESERVATION DETAIL
REDEEM NOW
8 vouchers sold, 6 validated

3) Voucher was already redeemed



12:30
Validate
X ALREADY REDEEMED
**Otevřený voucher na dobré jídlo do restaurace na 300, 500 Kč**
Pobyt kdykoli | 2 osoby | 3 dny (2 noci) | Období Pá 29. 6. – Ne 16. 9. 2018
Voucher was already redeemed on Aug 11, 2019 11:33 via Partnerské rozhraní
8 vouchers sold, 6 validated

4) Voucher is expired, it can't be redeemed

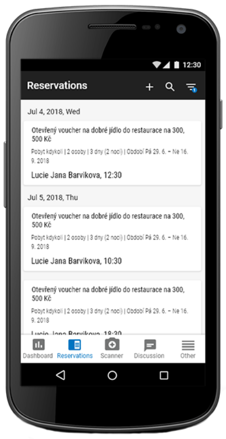


12:30
Validate
X EXPIRED
**Otevřený voucher na dobré jídlo do restaurace na 300, 500 Kč**
Pobyt kdykoli | 2 osoby | 3 dny (2 noci) | Období Pá 29. 6. – Ne 16. 9. 2018
Voucher has expired on Aug 11, 2019
8 vouchers sold, 6 validated

3

## Create a reservation

1. Tap Reservations in the bottom bar
2. Tap + sign in the top menu to create a new reservation
3. Fill in the code of the voucher you want to create the reservation for



Jul 4, 2018, Wed

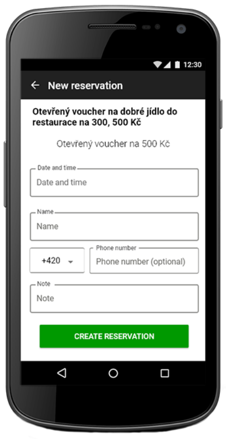
Otevřený voucher na dobré jídlo do restaurace na 300, 500 Kč
Pobyt kdykoli | 2 osoby | 3 dny (2 noci) | Období Pá 29. 6. – Ne 16. 9. 2018
Lucie Jana Barvikova, 12:30

Jul 5, 2018, Thu

Otevřený voucher na dobré jídlo do restaurace na 300, 500 Kč
Pobyt kdykoli | 2 osoby | 3 dny (2 noci) | Období Pá 29. 6. – Ne 16. 9. 2018
Lucie Jana Barvikova, 10:30

Otevřený voucher na dobré jídlo do restaurace na 300, 500 Kč
Pobyt kdykoli | 2 osoby | 3 dny (2 noci) | Období Pá 29. 6. – Ne 16. 9. 2018
Lucie Jana Barvikova, 18:30

4. Fill in the details of the reservation into the form
5. Confirm the reservation by tapping the Create reservation button
6. Reservation is created, you can see it in your list of reservations



Otevřený voucher na dobré jídlo do restaurace na 300, 500 Kč

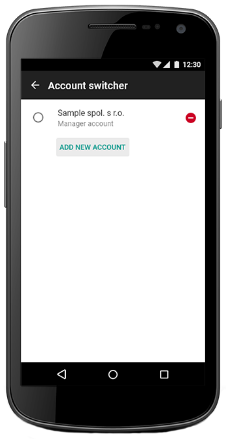
Otevřený voucher na 500 Kč

Date and time: Date and time
Name: Name
+420 | Phone number: Phone number (optional)
Note: Note

4

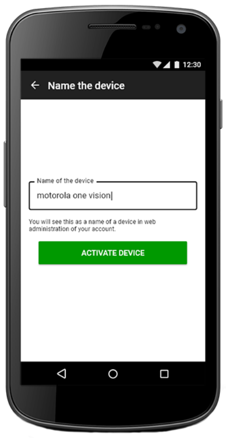
## Add a new account

1. Go to the Others tab
2. Tap Account switch
3. Now you are in the Account switcher menu
4. Tap Add another account to add another account
5. Login into www.slevomat.cz/partner
6. Go to tab **Mobile application**
7. Click **Add a new device**
8. Scan the code with the application. Or tap the Pen button to type in the code.



Sample spol. s r.o.
Manager account

9. Fill in the name for your device or use the generated one
10. Tap Activate the device
11. Your another account was activated



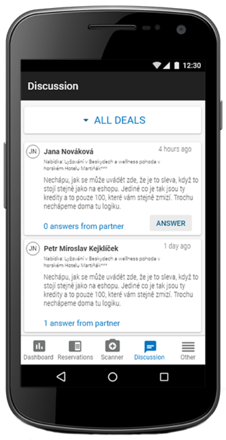
Name of the device: motorola one vision

You will see this as a name of a device in web administration of your account.

5

## React in discussion

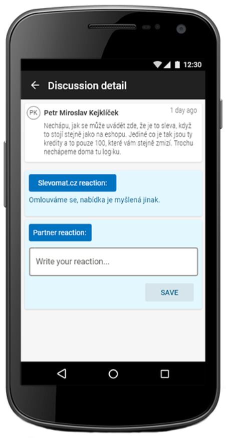
1. Tap Discussion in the bottom bar
2. Choose the question you want to answer by tapping on the Answer button



JN Jana Nováková 4 hours ago
Nabídka: Lyžování v Beskydech a wellness pohoda v horském Hotelu Martiňák***
Nechápu, jak se může uvádět zde, že je to sleva, když to stojí stejně jako na eshopu. Jediné co je tak jsou ty kredity a to pouze 100, které vám stejně zmizí. Trochu nechápeme doma tu logiku.
0 answers from partner

JN Petr Miroslav Kejklíček 1 day ago
Nabídka: Lyžování v Beskydech a wellness pohoda v horském Hotelu Martiňák***
Nechápu, jak se může uvádět zde, že je to sleva, když to stojí stejně jako na eshopu. Jediné co je tak jsou ty kredity a to pouze 100, které vám stejně zmizí. Trochu nechápeme doma tu logiku.
1 answer from partner

3. Type in your answer to the question
4. Tap Save to send the answer



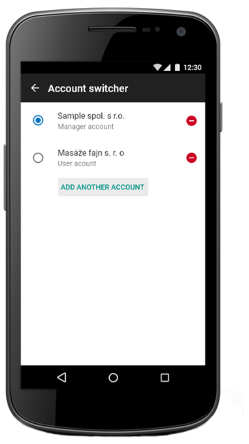
PK Petr Miroslav Kejklíček 1 day ago
Nechápu, jak se může uvádět zde, že je to sleva, když to stojí stejně jako na eshopu. Jediné co je tak jsou ty kredity a to pouze 100, které vám stejně zmizí. Trochu nechápeme doma tu logiku.

Omlouváme se, nabídka je myšlená jinak.

6

## Switch accounts

1. Tap Others in the bottom bar
2. Tap Account switcher from the list
3. List of your accounts is shown



12:30

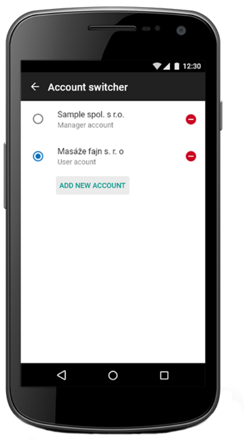
Account switcher

Sample spol. s r.o.
Manager account

Masáže fajn s. r. o
User acount

ADD ANOTHER ACCOUNT

4. Tap the name of the account you want to use
5. The blue radio button is shown next to the active account



12:30

Account switcher

Sample spol. s r.o.
Manager account

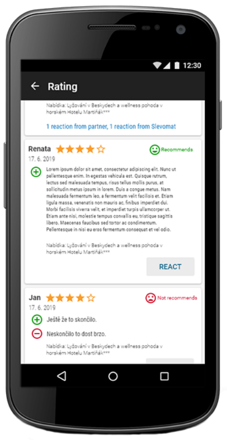
Masáže fajn s. r. o
User acount

ADD NEW ACCOUNT

7

## React to a rating

1. Tap Others in the bottom bar
2. Tap Rating from the list of options
3. Choose the rating you want to react to by tapping on the React button



Nabídka: Lyžování v Beskydech a wellness pohoda v horském Hotelu Martiňák***

1 reaction from partner, 1 reaction from Slevomat

Renata ★★★★☆ Recommends

17. 6. 2019

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc ut pellentesque enim. In egestas vehicula est. Quisque rutrum, lectus sed malesuada tempus, risus tellus mollis purus, at sollicitudin metus ipsum in lorem. Duis a congue metus. Nam malesuada fermentum leo, a fermentum velit facilisis et. Etiam ligula massa, venenatis non mauris ac, finibus imperdiet dui. Morbi facilisis viverra velit, et imperdiet turpis ullamcorper ut. Etiam ante nisi, molestie tempus convallis eu, tristique sagittis libero. Maecenas faucibus sed tortor ac condimentum. Pellentesque in nisi eu eros fermentum consequat et vel odio.

Nabídka: Lyžování v Beskydech a wellness pohoda v horském Hotelu Martiňák***

Jan ★★★★☆ Not recommends

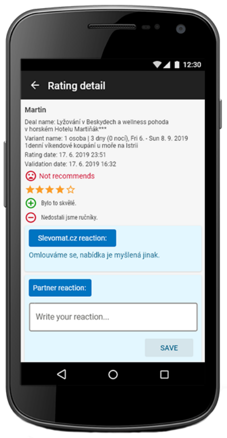
17. 6. 2019

Ještě že to skončilo.

Neskončilo to dost brzo.

Nabídka: Lyžování v Beskydech a wellness pohoda v horském Hotelu Martiňák***

4. Type in your reaction
5. Tap Save to send the reaction



Martin

Deal name: Lyžování v Beskydech a wellness pohoda v horském Hotelu Martiňák***

Variant name: 1 osoba | 3 dny (0 nocí), Fri 6. - Sun 8. 9. 2019 1denní víkendové koupání u moře na Istrii

Rating date: 17. 6. 2019 23:51

Validation date: 17. 6. 2019 16:32

Not recommends

★★★★☆

Bylo to skvělé.

Nedostali jsme ručníky.

Slevomat.cz reaction:

Omlouváme se, nabídka je myšlená jinak.

Partner reaction:

8