Masaryk University
Faculty of Informatics

Using Flutter framework in multi-platform application implementation

Bachelor's Thesis

Miroslav Mikolaj

Brno, Spring 2020

Replace this page with a copy of the official signed thesis assignment and a copy of the Statement of an Author.

Declaration

Hereby I declare that this paper is my original authorial work, which I have worked out on my own. All sources, references, and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source.

Miroslav Mikolaj

Advisor: Ing. Lukáš Grolig

Acknowledgements

I would like to thank my supervisor Ing. Lukáš Grolig for his support and advice.

Abstract

The thesis aims to explore the possibilities of the Flutter, an open-source software development kit. In the theoretical part, analysis of cross-platform solutions is provided along with Flutter's description and its evaluation. Development of the Android mobile application takes part in the thesis, where it serves as a practical example of the Flutter's potential. The final version of the application provides a testing environment for Electronic evidence of sales with extended features.

Keywords

cross-platform, mobile, app, application, Dart, Flutter, EET

Contents

Introduction

In recent decades, mobile applications gained popularity as mobile phones started to be more common than they used to, primarily in developing countries such as India [1]. Businesses, as well as developers, have utilized the popularity of mobile devices to create useful tools and much more. However, applications on mobile phones entirely depend on a device's operating system. Mobile market consists of multiple operating systems (abbreviated as "OS"), with iOS and Android being the most popular ones. As the fragmented mobile OS market made reaching a broader audience harder soon after the mobile cross-platform development tools appeared such as Xamarin, Ionic, React Native or Flutter. Among the main advantages of developing with these solutions is the lesser need for native developers, lower time to market, unified design and more. In past years the new software development kit (abbreviated as "SDK") developed by Google brought a fresh approach with extraordinary results.

The thesis aims to describe and explore current possibilities of Flutter SDK, the new technology and an already popular tool for developing on multiple platforms with a single codebase. Part of the thesis is the Android and iOS mobile application aiming to deliver a simple yet useful tool for entrepreneurs and small businesses obligated with Electronic evidence of sales (abbreviated as "EET" in Czech). Businesses and entrepreneurs have to register every sale online via the Financial Authority servers to satisfy laws. The application intends to help users to perform tasks effectively providing a minimalistic UI and fast animations. The app works with a local storage which stores the data about a product list or the networking required during communication with the Financial Authority to test all of the capabilities of Flutter.

The beginning of the thesis individually describes the multi-platform development solutions such as Ionic, Progressive web applications (abbreviated as "PWA") and React Native with an emphasis on their positive features. The thesis continues with a paragraph containing a description of Flutter SDK and its crucial components. It discusses Dart, the programming language behind Flutter, Dart VM, the essential component in Flutter's high performance and the open-source

layered libraries architecture that forms Dart's framework called Flutter. The third paragraph of the thesis briefly mentions circumstances around EET and narrates how the system works. The last chapter sums up the phases of development such as User Interface (abbreviated as "UI") or the database design, the use case diagram and the programming itself. The text introduces the used technologies such as Dart, the programming language, Flutter, the Dart framework, NoSQL database Hive or SOAP protocol.

The thesis summarises various tools and methods for the multiplatform development and points out their strong and weak sides. However, the main topic is Flutter and its current position in the development and on the market. The work goes through the core of Flutter technologies and describes its engine, libraries, and other features to highlight how it differs from other solutions. Mobile enterprise application for reporting revenue to the Financial Authority is part of the project as a practical example of Flutter code. This work does not describe the final product as Flutter is a young project, thus leaving space for future research and a possible update of the work.

1 Multi-platform development options

It is a time and resources consuming task to develop a native application for every platform like Android, iOS and web for example. Each of them demands knowledge of different languages and technologies. Apps for Android are written either in Kotlin or Java, iOS apps in Swift or Objective-C and web developers should be familiar with JavaScript, HTML and CSS. Solutions for cross-platform development come handy as they reduce the development time by minimizing the amount of the unique code, otherwise being written on each platform individually, thus enable the code reusability. On the other hand, those applications may have lower performance than their native counterpart, or it might take longer for the non-native app to be able to implement new features provided after OS update.

Nowadays, we have a handful of options to choose from when we are going to develop a cross-platform application from which each solution targets to ease different aspects of development. Picking the right framework or technology depends on various factors but mainly on the type of application and the aim of the final product. In cases where the application has to work with primitive features provided by the OS or have to be precisely optimised, it is better to develop native.

1.1 Ionic

"Ionic Framework is an open-source UI toolkit for building performant, high-quality mobile and desktop apps using web technologies (HTML, CSS, and JavaScript)" [2]. Framework was initially released in 2013 by Drifty Co.. However, the first final version came out in 2015.

As a standalone tool, Ionic helps to create web applications by providing developers with its library of components built upon web technologies such as HTML, CSS and JavaScript. In a combination with Android and iOS implementation of a web view component, optionally a JavaScript framework and a mobile development framework Apache Cordova, is Ionic able to deliver a hybrid mobile application, thus being a real cross-platform solution.

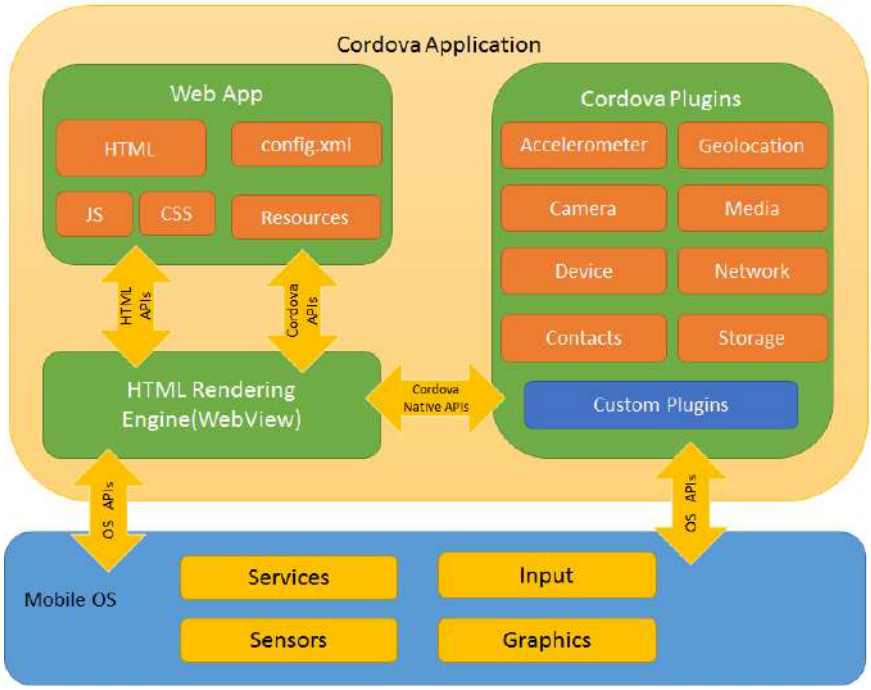
A web view is a component of native applications designated to render a web content within the application screen. This component

provides applications with the possibility of utilizing a web browser power inside independent applications.

A mobile app with an incorporated web view and a wrapper advancing native APIs for functionality like GPS or camera is known under the term a hybrid application.

Open-source framework Apache Cordova was developed under the pretext of making cross-platform development more straightforward. It wraps a native functionality such as geolocation or camera, separated in plugins, providing an option to incorporate only plugins necessary for the functionality. Access to the API via plugins is then distributed to the web view which contains packaged HTML, CSS, JavaScript and other sources [3].

Figure 1.1: Cordova's architecture[1]



In the past, Ionic used to be dependent on a JavaScript framework Angular, developed by Google. This framework's dependency

1. From `https://cordova.apache.org/docs/en/latest/guide/overview/index.html`

allowed wiring UI elements via business logic to the API functionality of Apache Cordova. Nowadays, framework succeeded in achieving not only to be able to cooperate with various JavaScript frameworks, but to operate completely without one [4].

Ionic aims to create a hybrid application with the feel of a native app. However, the web view used in the process is acting restrictive and is bringing slower performance due to the need for communications with native API via Cordova wrapper. It is not recommended to develop graphic extensive applications in Ionic as it lacks performance needed for a stable frame rate. Ionic appeals to the broad audience of developers as it utilizes already popular web platform technologies, even though it reaches lower performance than other cross-platform solutions. The framework is also supported by a stable team of developers and its Github repository has gained over forty thousand stars from developers interested in the progress of the project. Although, it has the least amount of stars from all open source options mentioned.

1.2 Progressive web application

The term progressive web application was coined for the first time by a duo of Google's designer and an engineer in 2015. A web application is an application developed with standard web technologies such as HTML, CSS and Javascript running on the remote server. To access the web app, the user needs to have a web browser and an internet connection necessary for communication between user and server. As time progressed, web apps started to be responsible and more comfortable for the mobile users. They have an advantage over native apps as they do not require installation and regular app updates download.

On the other hand, web applications have more disadvantages than benefits as they are dependent on the internet connection, cannot store data on local storage or be incapable of handling hardware APIs. With the arrival of progressiveness to web applications, offline mode via service workers, the capability to store data on the local cache, feel of full native app and last but not least access to hardware APIs like camera, NFC, GPS came. It is relatively simple to create PWA, if web application already exists. To generate PWA from web app developer

have to add a corresponding manifest.js file containing app name, icon and other essential information and implementation of the service worker along with secure communication via safe https protocol. After all this, the browser offers users to add web app on their home screen. The whole process is relatively simple, and the app takes a lot less space in device memory with speed after the initial load nearly as fast as native apps [5][6].

1.3 React Native

After an internal hackathon and months of development, Facebook released the first version of the framework for cross-platform development in 2015. React Native is a Javascript framework used to render native UI elements depending on the platform it is running on. Code for a button will generate a different native button element on Android and different on iOS. Internally, React Native application has two threads. One thread is the default for every native app. Default thread processes user's gestures and handles displaying UI elements. The second thread is an internal Javascript thread used to manage a virtual machine responsible for executing the business logic of the React Native application. Those two threads do not communicate directly. Instead, a bridge between transfers batched data to achieve efficiency. The bridge transfers data asynchronously in a serializable manner so both threads cannot have the same date at the same time. Because of the bridge, the application written with React Native is lacking the performance of the native app. However, the lack of performance speed compensates the speed of development. As it enables wiring Java, Swift or Objective-C code to Javascript code written it reduces the amount of duplicate code. With this, however, appears another problem es experienced Airbnb [7]. They have had to manage three teams of developers for React Native, Android and iOS as Airbnb wired their ongoing project with new technology. Even if development starts with Javascript and React Native framework, native code is necessary because of the platform-specific problem requiring individual treatment [8][9].

2 Flutter introduction

"Flutter is an app SDK for building high-performance, high-fidelity apps for iOS, Android, web (beta), and desktop (technical preview) from a single codebase" [10]. Initially released in May 2017, Flutter is a relatively young addition to the line of multi-platform development tools. While stable version 1.0 was released only on December 4, 2018 it gained a huge support from the community. So far Google itself have used Flutter for development of their own applications such as Google Ads, Google Assistant, Stadia. But not only Google, corporations like Tencent, Alibaba or BMW have their own apps developed in Flutter as well [11].

Figure 2.1: Flutter's architecture[1]

2.1 Dart

After an in-depth research, the team behind Flutter have chosen Dart as a programming language on which is the framework based. Dart is as well as Flutter developed under Google's supervision, that makes a symbiosis between two projects more fluid and stable. Even though it

1. From `https://flutter.dev/docs/resources/technical-overview`

is not a widely used language beside Flutter, it has numerous reasons to be the best match. Though it may seem to be discouraging for a developer to learn a new programming language pretty much not used outside Flutter, it serves well its purpose [11].

2.1.1 Dart VM

Dart's virtual machine (abbreviated as "VM") consists of components for executing Dart code natively. It includes a Runtime system such as Garbage Collection or Snapshots. Snapshots speed the application startup process by saving a snapshot of the heap to a simple file available for loading. Snapshot feature gives a speed boost, especially to slower devices with a restricted battery. Dart VM also includes Just-in-Time, Ahead-of-Time compilation pipelines, interpreter, ARM simulator and debugging, profiling, hot-reload components accessible via service protocol. A virtual machine, in this case, does not mean the code always interprets or compiles Just In Time (abbreviated as "JIT") as ahead of time (abbreviated as "AOT") pipeline can compile Dart code into machine code.

2.1.2 Compilation pipelines

One of the main features in Dart is that it is possible to compile a code AOT and also JIT. Dart code compiles JIT while iterating development cycles, this results in an overall slower application while developing but enables Flutter to gain an advantage of fast compiling and to utilize a hot reload feature. When the app is ready to be deployed, Dart compiles it AOH into a native code resulting into a performant end product. While dart2js compiler transforms dart code into Javascript thus code runnable in the web browser, compiling into a native code secures dart2native compiler which broadens possibilities of compiling into Android and iOS operating systems native code with Windows, Linux and macOS. However, it is not suitable to deploy an application for all of them as Windows and Linux are only in a technical preview and macOS is in beta so far.

2.1.3 Garbage Collecting

Another provided feature is garbage collection. Dart suits its generational garbage collector to be efficient, especially for a rapid creation and a destruction of objects and to provide a hook for Flutter engine to determine if the app is in the idle state. Collecting garbage consists of two phases: the young space scavenger and the parallel mark-sweep collectors. The first generation, the young space scavenger, is used for cleaning short-lasting objects such as stateless widgets. The process consists of dividing memory into two halves and when one half fills, the collector starts marking active objects and then copies them to the inactive half, resulting in leaving dead objects in the inactive part of memory. Mark-sweep, the second generation garbage collector manages objects with a longer lifespan which are scarce in Flutter app thanks to how widgets work. During two phases of mark-sweep, marking active objects and then recycling not marked ones are memory operations and the UI thread blocked.

2.1.4 Hot Reload

Dart VM is capable of inserting code changes to the running program. During this process, the program keeps its state and changes only the application's behaviour. It takes the Hot reload only a few milliseconds to apply the new code. Another possibility is to Hot restart the program, which takes more than ten seconds but also sets the state to default values.

2.2 Flutter

It was important for Eric Seidel, Flutter's co-founder, to build this project around three main attributes. It had to be fluid to make developer's work smooth and to eliminate waiting. The Hot reload feature accomplishes the elimination of waiting time for the project to compile and to make the programmers more focused on the developing part. Next attribute to be fulfilled was a flexibility that allows incorporating most of the designer's ideas into the final product. This aspect aims for a great variety of pre-made widgets and their high modifiability. Lastly, Mr. Seidel's vision is to have a faithful project. Faithfulness

of the project is that the final product should feel and look amazing in the hands of the user. On top of that, Flutter is the open-source providing full access to developers.

Flutter framework architecture consists of four simplified layers, as seen above. Those layers are dart:ui library granting lowest level access possible. Above dart:ui lies Render library, Widgets library and Material & Cupertino libraries mostly used on daily bases by app developers [11][12].

2.2.1 dart:ui library

Flutter exposes fundamental functions at the lowest layer of its framework. The dart:ui library communicates directly with the engine and Skia graphics library providing access to the canvas, textbox or paint classes with a crude functionality as a developer is supposed to calculate all coordinations and data for redrawing the screen.

2.2.2 Rendering library

The Rendering library builds a more sophisticated and a more comfortable to use interface on the previous library. This layer calculates coordinations and keeps them cached with the help of complex algorithms. To speed up the rendering process, during the layout and painting phase, the rendering pipeline uses one-pass algorithm traversing the whole rendering tree once for each phase.

2.2.3 Widget library

Every object created at the Rendering layer has its widget. Widgets are immutable and short-lived objects with a manageable dataflow keeping information about the user interface. Widgets layer provides ready-to-use UI components mostly separated into three categories. Developers usually compose layout, painting and hit/testing widgets from Widgets library into the custom and more complex widgets.

2.2.4 Material & Cupertino library

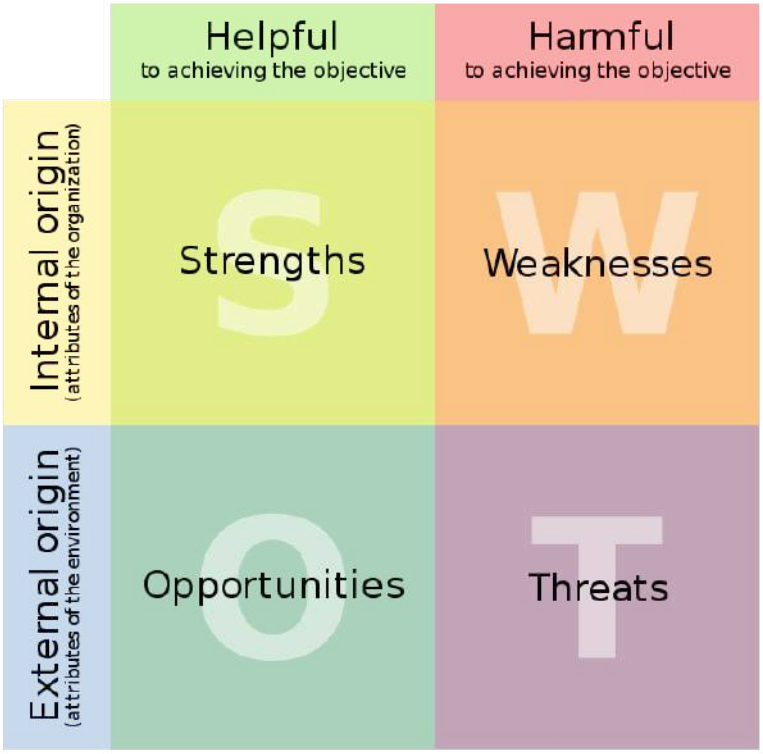
Last and the library on the top layer is Material & Cupertino library consisting of pre-made widgets looking and behaving almost exactly

like the natural Android or iOS components. Flutter team assembled this layer to make the developer's experience more enjoyable and straightforward.

2.3 SWOT analysis

The SWOT analysis helps with determining and adjusting the overall goal by assessing four aspects behind SWOT, an Strengths, Weaknesses, Opportunities and Threats. Diversity of aspects should force evaluators to look at the product and deeply evaluate it [13].

Figure 2.2: Four elements SWOT analysis matrix[2]



2. Image resourced from `https://en.wikipedia.org/wiki/SWOT_analysis`

2.3.1 Strengths

Products Strength is an essential and unique aspect that distinguishes it from its competitors. Flutter's strengths:

- stable rendering at 120 frames per second
- rendering the whole screen of the device
- reliable hot reload
- Dart
- Google backing

Flutter comes with a handful of strengths. Its main advantage is, the Flutter team took a risk and created something new, the product with a new approach to the problem of cross-platform development. Thanks to the utilizing Skia Graphics Engine, a C++ graphic library[3], it is possible for Flutter to render the whole screen, at 120 frames per second natively. Another piece that helps Flutter achieve its speed is Dart, a client-optimized language for fast apps on any platform. Both Flutter and Dart development teams closely cooperate as they are both backed by technological giant Google.

2.3.2 Weakness

Weakness is the exact opposite of strength. Successful analysis should consist of an honest critique of the weaknesses. Flutter's weakness:

- missing web and PC platforms support
- bigger app size than in native apps
- possible changes in API

Flutter's weak points come from not supporting common platforms like web, Windows, Linux or Mac as they are still in the development phase. Also, as flutter app ships with the engine, framework and other smaller parts final application's size is slightly bigger than the native app's.

3. Reference from `https://skia.org/dev/flutter`

2.3.3 Opportunities

Opportunities are beneficial situations or changes that arise from the external environment. Flutter's opportunities:

- a steep rise in popularity
- open-source nature of the project

The result of Flutter's strengths reflects in its popularity among developers. As a result, Flutter took over React Native's leadership in Github stars[4]. This increased interest may provide future chances for Flutter.

2.3.4 Threats

Situations possible harmful or negatively influencing assessed product rank among threats. Flutter's threats:

- over seven thousand open issues on Github
- youth of project
- lack of experienced developers

Flutter is still a young project. First stable release came out on December 2018[5]. Result of its youth is a relatively high amount of opened issues on Github, which is over seven thousand. However, the number might seem high it might not be real number as among issues are duplicates and also requests for future features [14].

4. Flutter's Github repository with an actual number of stars at `https://github.com/flutter/flutter`
5. Source from `https://en.wikipedia.org/wiki/Flutter_(software)`

3 EET - Electronic evidence of revenues

In 2016 Czech parliament launched the EET system to gain more control over business's revenue and to set equal conditions for the revenue taxpayers. It is common in economics to form such a part of the economy called the informal economy or grey economy. This branch contains activities unmonitored or untaxed by a government, thus leading to the leak of tax income for the state budget. In the last decade, similar solutions were applied by European countries such as Croatia, Germany, Slovakia, Poland and others. There is no standardized implementation of the EET system in the EU, leading to a different system application. The Czech solution is similar to the Croatian revenue evidence but with few modifications being made. The major one is not requiring customers to take every receipt.

The current law requires businesses to online register all money transactions via the Financial Authority, then handing the customer receipt as printed or as an electronic document. The system rolled out in four waves separated by a type of market position. The first wave contained restaurants and accommodations. The second wave set mandatory revenue reporting to retail and wholesale. The third wave merged with the fourth wave contains crafts, agricultural and other business activities.

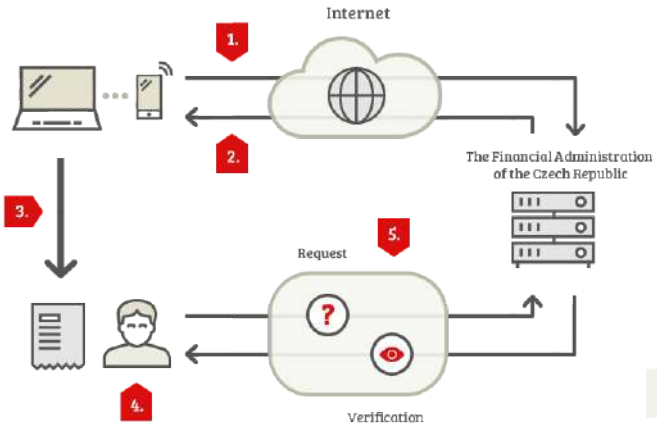
3.1 EET overview

The center point in EET are servers of the Financial Authority (abbreviated as "FA") as they issue an identity certificate to businesses[1], serve as a generator for unique code "FIK" on each receipt or let the customer check if their receipt is valid. Another essential point in the system is a device with an application capable of transmitting data about the sale online over the Internet. On the diagram below, communication happens between three entities, the device with EET software, FA server and customer.

---

1. Guide for generating the certificate, source in the Czech language available at `https://epodpora.mfcr.cz/cs/seznam-okruhu/eet--certifikaty/navod-jak-vygenerovat-certifikat-4418`

Figure 3.1: EET data flow[2]



The flow of the communication is straightforward and starts when a device designated to operate with the FA server prepares a data package containing various information about a sale, business identity and other crucial data into XML format. The XML package transmits over the Internet via SOAP protocol. After the FA receives the data, it checks if the file is formatted correctly, if the data are valid and if the business information matches. As a response, FA answers with another XML file containing either confirmation with a unique code called FIK or an error XML message [15].

3.2 XML message

Web Services Description Language[3] (abbreviated as "WSDL") is used in the description of the interface of the web service along with XSD[4], an XML schema definition determining the validity of the XML mes-

2. Original image source available at `https://www.etrzby.cz/cs/english-version-609`
3. WSD file provided by the Financial Authority at `https://www.etrzby.cz/assets/cs/prilohy/EETServiceSOAP.wsdl`
4. XSD file provided by the Financial Authority at `https://www.etrzby.cz/assets/cs/prilohy/EETXMLSchema.xsd`

sage. An XML message itself consists of only one element "Trzba" which embodies three elements holding data crucial to sale registration. The last element "KontrolniKody" is individual as it holds signed and hashed values mentioned in the next subsection [16].

Listing 3.1: Data message example

```
<Trzba xmlns:xsi="http://www.w3.org/2001/
   XMLSchema-instance" xmlns:xsd="http://www.w3.
   org/2001/XMLSchema" xmlns="http://fs.mfcr.cz/
   eet/schema/v3"/>
  <Hlavicka uuid_zpravy="UUID_VALUE" dat_odesl="
     2019-08-11T15:36:25+02:00" prvni_zaslani="
     true" overeni="false"/>
  <Data dic_popl="CZ00000019" id_provoz="141"
     id_pokl="1patro-vpravo" porad_cis="
     141-18543-05" dat_trzby="2019-08-11T15:36:14
     +02:00" celk_trzba="236.00" rezim="0"/>
  <KontrolniKody>
    <pkp xmlns="http://fs.mfcr.cz/eet/schema/v3"
        digest="SHA256" cipher="RSA2048" encoding
       ="base64">PKP VALUE</pkp>
    <bkp xmlns="http://fs.mfcr.cz/eet/schema/v3"
        digest="SHA1" encoding="base16">BKP VALUE
       </bkp>
  </KontrolniKody>
</Trzba>
```

3.3 Cryptography

Strict security measures surround messaging with web service. The first step in achieving security standards for the message to be accepted is to sign a string consisting of various attributes of the "Data" element connected with "|" [5]. To create the value of "pkp" element belonging in the element "KontrolniKody", a string from the previous step is hashed with algorithm SHA256[6] then signed, using the private key from

5. ASCII character with value 124
6. Specification at `https://tools.ietf.org/html/rfc6234`

the certificate, with algorithm RSASSA-PKCS1-v1_5[7] and encoded in Base64[8] format with a length of 344 characters.

Another verification code from element "KontrolniKody" a "bkp" is decoded "pkp" element hashed with the SHA1[9] algorithm, encoded in Base16[10] format with the character "-"[11] inserted after every eight positions in the prior encoded string.

After a successful computation of security codes and assembly of the valid XML message, constructing accurate header for SOAP request encapsulating the earlier message is essential. This step takes three values to fill. The first is the value of "BinarySecurityToken" element encoding in Base64Binary[12] format x509 certificate[13].

The last two security measurements are to sign "Body" element formatted in exclusive canonised XML format[14] with the RSA-SHA256[15] algorithm and to hash signed value with SHA256. Both values are used in SOAP header construction [16].

7. Specification at `https://tools.ietf.org/html/rfc3447`
8. Specification at `https://tools.ietf.org/html/rfc4648`
9. Specification at `https://tools.ietf.org/html/rfc3174`
10. Specification at `https://tools.ietf.org/html/rfc4648`
11. ASCII character with value 45
12. Specification at `https://tools.ietf.org/html/rfc4648`
13. Specification at `https://tools.ietf.org/html/rfc5280`
14. Specification at `https://www.w3.org/TR/2002/REC-xml-exc-c14n-20020718/#`
15. Specification at `https://tools.ietf.org/html/rfc6594`

4 Application design

Following sections contain a few major design schemes necessary for the smooth project's development process. Preparing schemes beforehand is a standard step in modern development, leading to a better understanding of a problem to be solved. To begin with, we assembled a list of functional and nonfunctional requirements in the first section. Presented requirements serve as the initial step providing a crude representation of the final product's possibilities. The next section narrates various user interactions with the application itself in a use case diagram. After understanding what the application should do and how the user workflow should look from both requirements and use case diagram, it is vital to design the user interface (abbreviated as "UI"). Well designed interface not only provides an aesthetical experience but what is more, makes it comfortable to use the application. The last section analyzes data and their influence over each other in database design.

As the previous chapter mentioned, the objective of this project is to provide a user-friendly interface for electronic evidence of revenue which is mandatory for businesses in the Czech Republic. In summary, the user has to create a data message and send it over to the servers of the financial authority. Following sections design the process necessary for dataflow and user workflow based on mentioned premises and the system requirements set by the provider of financial authority.

4.1 Requirements

Requirements specify what should developers implement, describe behaviour or attributes of the system. Defined earlier on in the system development, they might be specifying constraints on the system, defying general property of the system or describe the user-level facility. Writing the requirements is a very individual experience, as every situation or organization require a different level of detail or style of writing [17].

4.1.1 Functional requirements

As a part of software engineering, functional requirements take part in creating the structure of a future application. Functional requirements take the form of a list where each row contains a unique identifier and description of what tasks the system should perform. Here are listed up functional requirements for the electronic evidence of revenue application [17]:

- 1. The application let the user create a business profile.
- 2. The application provides a switch between a night time and a day time theme.
- 3. The application shows all valid products.
- 4. The application allows creating, editing and removing products.
- 5. The application lets users choose products and send them on servers of the Financial authority.
- 6. The application saves payment data locally when an internet connection is not present for later data resend.
- 7. The application provides a calculator feature for quick payments.

4.1.2 Non-functional requirements

Another set of requirements assemble from listing non-functional requirements (abbreviated as "NFR"). Unlike Functional requirements, it does not tell what application is or is not supposed to do. NFR instead describe how the system is supposed to be. After examining the documents provided for EET developers, I have assembled this list of non-functional requirements [17]:

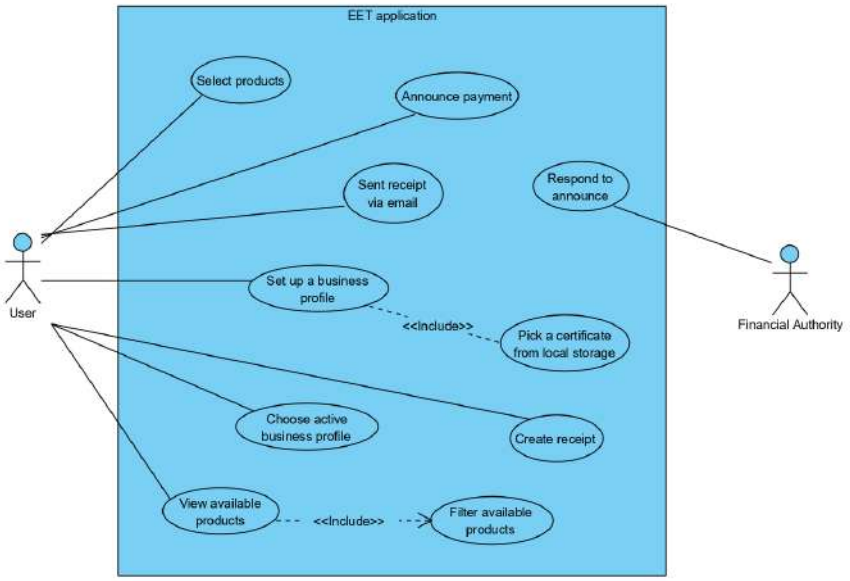
- 1. Mobile application shall be developed in the Flutter framework.
- 2. The message shall be sent to the Financial authority should be in XML format over SOAP, a messaging protocol.

- 3. The message shall be encrypted with the key pair provided by the Financial Authority to the user.

4.2 Use case diagram

Diagrams in Unified Modeling Language (abbreviated as "UML") divide into two categories, structural and behavioural either illustrate approaches to modelling a system differently. Both categories contain multiple diagrams, and this section narrates the use case diagram which belongs to the Behavioral UML diagram section.

Figure 4.1: Use case diagram



Use case diagram (abbreviated as "UCD"), as seen above in Figure 4.1, is a crude representation of users' interactions with a system, often not consisting of more than 20 interactions. UCD consists of four parts. The two principal parts are actors representing either users or other entities making interaction with the system and use cases which are available actions. Another two essential parts of the diagram are associations linking actors to use cases and system boundaries dividing actors from use cases [18].

4.3 User interface/experience design

From the very beginning of the user interface (abbreviated as "UI") and the user experience (abbreviated as "UX"), the design came a long way. It had to evolve either due to a growing competition in the technology field or continuously higher expectations of design consumers. It is not so uncommon to mistake one for each other as they work in a close relationship and depend on each other. Both UI and UX are essential for the final product because of their ability to immerse the user. Without a well-designed UI, the application would feel chaotic while without enough thought through UX design, using the application would feel strange and unnatural to the user. Maintaining the design clean and straightforward leads to a user's satisfaction.

Figure 4.2: Item menu concept

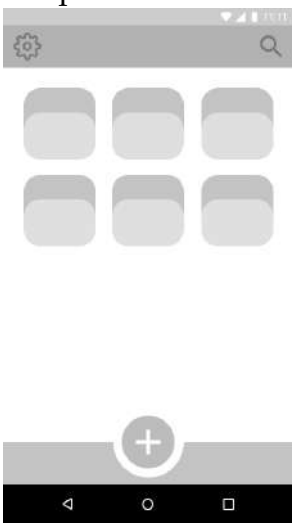
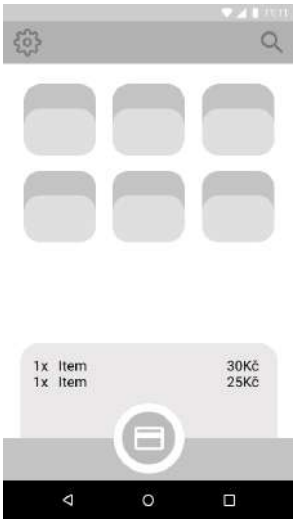


Figure 4.3: Concept of panel for selected items



The main principles behind designing the whole application's UI were clean design displaying only information currently required and intuitively positioned clickable components often changing their functionality and icon based on data during user workflow. Most application screens provide only one button positioned usually in the middle of a bottom bar with a self-explanatory icon. For example, concept designs provided in Figure 4.2 and Figure 4.3 present only a bottom bar with button, upper bar and product list consisting of clickable squares as that is the only interface that user needs in the given moment. Also, when a user has to add or edit data not to break the user workflow, a dialogue window 4.4 pops up overlaying only part of the current

screen, comfortably returning to the invocation point after the task completion.

Even though the application focuses more on not to clog the user with unimportant data, few optional components are present. As an example serves a switch between a light and a dark theme, which may not be crucial for the app functionality but rather provide convenience to the users. However, during typical user workflow, optional functionality is reduced to a minimum leaving only options that matter. Workflow is also kept straightforward and smooth with fast animations providing a clear representation of the application functionality in the background. For example, items clicked in the menu fly into the slider in Figure 4.3 catching attention and giving feedback about what happened. All those examples contribute to pleasant UX and users' satisfaction [19].

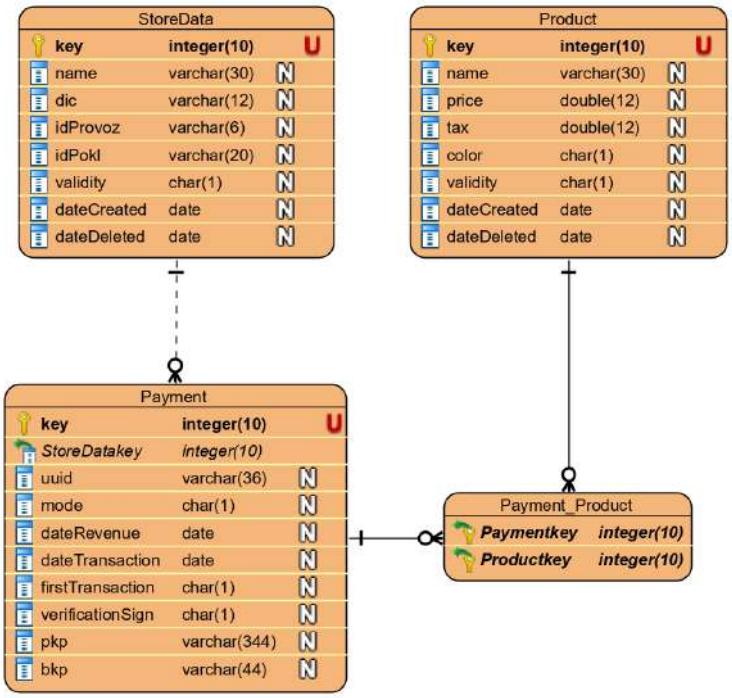
Figure 4.4: Design of the new product dialog

4.4 Database design

At last, it is crucial not to forget to outline the data structure as the whole application depends on data. A standard method to plan such a structure is to use Entity Relationship Diagram (abbreviated as "ERD"). This diagram consists of entities, their attributes and relationships between entities.

ERD in Figure 4.5 depicts relationships in the EET application between three data models of product, store and payment data. It is necessary to keep the store and product data saved on a local database as the application utilizes them between users sessions. On the other hand it is not essential to keep payment data in the local database. However, it is better to include future extensions that may include processing the data with machine learning or displaying them to the user in the form of a graph.

Figure 4.5: Entity relationship diagram



5 Application implementation

The last chapter describes the process of the implementation, and its details, such as available editors, different libraries included in the project and struggles or errors occurred.

5.1 Preparation

However, it is possible to develop a Flutter application without the Integrated Development Environment (abbreviated as "IDE") as it provides Command Line Interface (abbreviated as "CLI"). IDE's features are a welcome enhancement of developer's comfort. The popular tools to choose from for Flutter developers nowadays are IntelliJ IDEA or Android Studio[1], IDEs that provide full support for Dart and Flutter together with a lot of useful and unuseful features. On the other hand, there is VSCode, a text editor which users can modify with various extensions called plugins providing additional functionality on the top of a basic editor functionality. The development of this project takes a big part in the VSCode as it offers plugins for Flutter and Dart. VSCode does not include unimportant functionality for the Flutter development and provides a cleaner environment without much application overhead, contributing to the positive user experience.

5.2 Packages

With Flutter comes pub.dev[2], a platform for sharing packages. It lists user-developed libraries available to use, easing the need for developing everything from a scratch. Each package has an overall score assigned, consisting of three factors. Program for determining package score is still in development, and at the moment, the score consists of the package popularity, health and the maintenance. On top of the

---

1. Tutorial setup available at `https://flutter.dev/docs/development/tools/android-studio`

1. Tutorial setup available at `https://flutter.dev/docs/get-started/editor?tab=vscode`

2. pub.dev the main page for Flutter packages at `https://pub.dev/flutter/packages`

points assessment, Flutter Favorite program gives credit to the top solutions. Pub.dev provides a variety of different options to pick from, either advanced compositions of widgets with extensive functionality like carousels, sliders or solutions for data management or networking. The EET project consists of fifteen different flutter packages and one native Java library.

5.2.1 Provider with ChangeNotifier

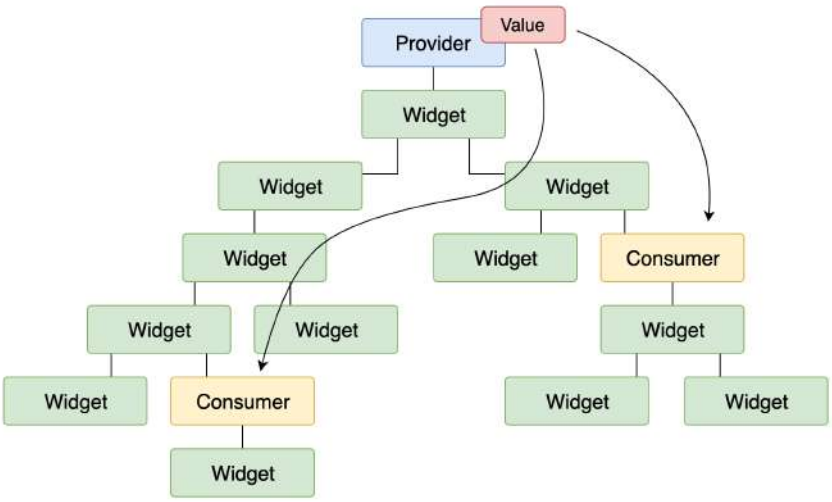
Flutter takes a different approach than imperative frameworks like Android SDK and iOS UIKit. Instead of modifying UI, it redraws UI based on the actual value of the app state. As Flutter composes widgets in the tree structure, Provider helps to propagate change of state to the consumers by occupying the position of their nearest common ancestor. When such a change of state happens, ChangeNotifier quickly notifies all subscribers, which as a result redraw themself and their descendants. Flutter team recommends Provider as an introduction to the declarative development by dedicating it official tutorial[3] and by listing this solution to the state management in their Flutter's Favorite program.

5.2.2 Hive, SharedPreferences and SecureStorage

To be able to use data, a fundamental building block of the application, between the app session, it is necessary to store them. Storing the data can take a lot of different approaches as data can be stored using remote service on a server such as Googles' Firebase Database or in the local storage. The project, which is a part of the thesis, utilize local NoSQL database called Hive[5] as the primary storage for data persistence. Among best Hive features belong fast operations, no native dependencies, its simple API and auto migration[6]. Another option for local data storage would be the classic SQL database, SQLite[7] with an

3. Official tutorial at `https://flutter.dev/docs/development/data-and-backend/state-mgmt/simple`
4. Image adopted from `https://www.raywenderlich.com/6373413-state-management-with-provider`
5. Package available at `https://pub.dev/packages/hive#-readme-tab-`
6. From package's Github repository `https://pub.dev/packages/hive`
7. Package available at `https://pub.dev/packages/sqflite`

Figure 5.1: Provider in Flutter's widget tree[4]



overly complicated query syntax and a slow performance compared to the Hive.

Platform support is another determining point. While Hive is a genuinely cross-platform plugin working on all platforms Dart compiles to, SQLite is restricted only to Android, iOS and macOS. While Hive currently does not support queries, it is an expected future feature[10] along with rewriting Hive's codebase in Rust with the help of upcoming Dart's feature dart:ffi library allowing to call native C APIs[11].

SecuredStorage[12] takes care of storing sensitive data like passwords and private keys with easy to use and understand API. The plugin utilizes native secure storage, Keychain in iOS and Android's Keystore, which encrypts its content. Nowadays SecureStorage plugin supports only mobile OS Android and iOS, leaving other platforms like web

9. Benchmark performed on Oneplus 6T with Android Q by author of Hive package. Resource from `https://pub.dev/packages/hive`
10. From the post of Hive's author at `https://github.com/hivedb/hive/issues/246`
11. Library is still in beta version, link to the documentation at `https://dart.dev/guides/libraries/c-interop`
12. Package available at `https://pub.dev/packages/flutter_secure_storage`

Figure 5.2: Benchmark comparing capabilities of popular data storage solutions for flutter.[9]

or PC OS supported. If this will be a case in the future, it might be reasonable to utilize the Hive's encrypted box[13] feature for sensitive data storage.

5.2.3 OpenEET

Java and the only platform-specific library used in the project, OpenEET[14] is an open-source and lightweight tool for operations related to the EET. Its API provides functionality for generating a request message from data given, creating security codes by hashing and signing data or sending a request to the FA servers. Native library functions are accessible via sending messages on platform channels that connect the Flutter application with its platform-specific host.

As a result of the Flutter frameworks relative youth, currently, there are not any existing packages satisfying the functionality needed for the strict FA cryptography policies. FA requires a specific type of

13. Documentation reference at `https://docs.hivedb.dev/#/advanced/encrypted_box`
14. OpenEET project Github library at `https://github.com/l-ra/openeet`

signing algorithm mentioned in the previous chapter, thus leading to the need for native libraries implementation. Such implementation would result in deflection from the focus of the thesis as it would bear developing native libraries in Swift for iOS and Javascript for the web. With a stable version of dart:ffi library implementation of signing algorithms in Dart might be possible.

5.3 Development

Flutter team created extensive documentation and tutorial to help with learning and understanding the framework, which is also open-source, enabling peaking at right into functions code. As mentioned before, Flutter comes with CLI, which is capable of an initialising project containing everything ready for development, thus providing a natural starting point[15]. Usually, all resources needed to developers programming cycle are the "lib" folder with all implementation and "pubspec.yaml" file managing package dependencies.

5.4 Result

The final product represents the Android application for registration of the revenue. For the application to work correctly, it is necessary to download certificates predetermined for the testing purpose by the Financial Authority[16]. After the initial setup, it is possible to use the application without the need of the internet as the app caches data about the receipt. During initial setup, the application prompts the user to enter information attached to the certificate as displayed in Figure 5.3. Other features provided in the application are product management 5.4 and filter, accessing cached receipt via bell button at the bottom app bar 5.5, displaying final receipt 5.6 or use mail application by the user preference to send the receipt. Screenshots of the final product referenced earlier:

15. Sourced from the tutorial at `https://flutter.dev/docs/get-started/test-drive?tab=vscode`
16. Testing certificates available at `https://www.etrzby.cz/cs/technicka-specifikace`

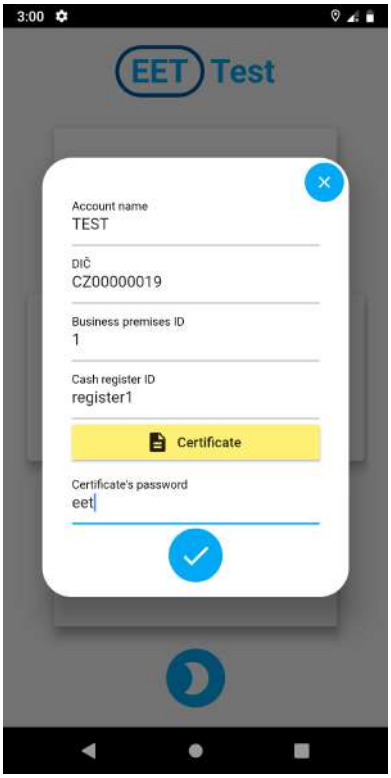


Figure 5.3: Screen with dialog for setting up a new profile.

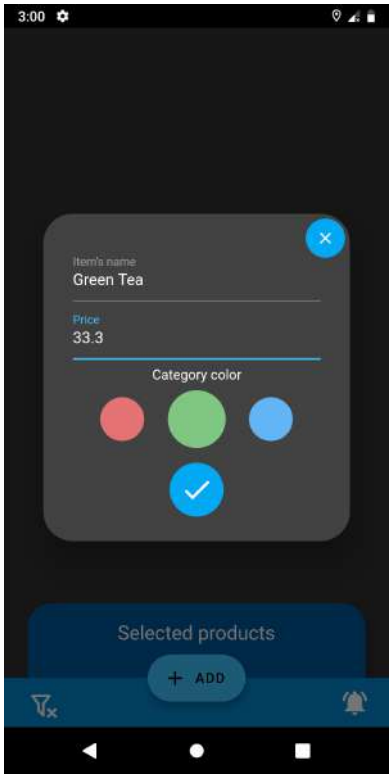


Figure 5.4: Screen with dialog for creating new instance of product.

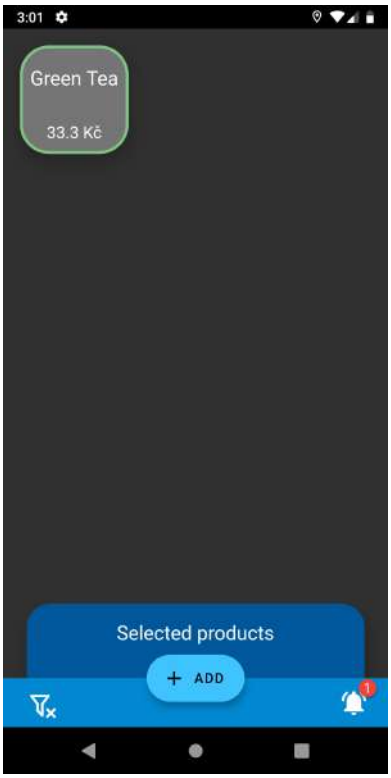


Figure 5.5: The screenshot shows the main screen with products grid and bottom app bar with a button for filtering products a the left and button for accessing locally cached receipt at the right.

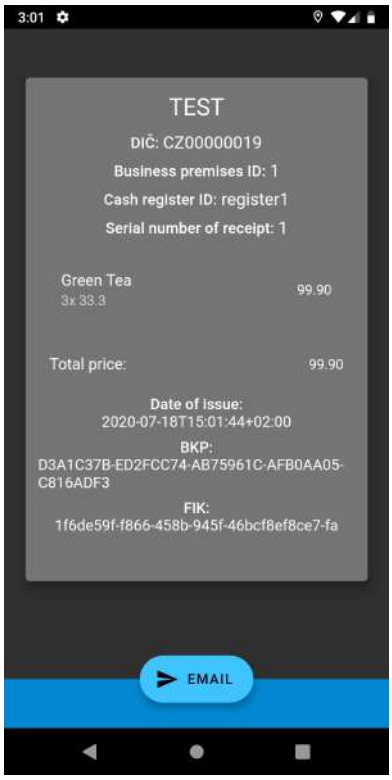


Figure 5.6: Screen displaying final receipt with a button for sending email.

Conclusion

The thesis sets the goal to evaluate the Flutter, describe its possibilities and test them on a practical example. Each chapter elaborates on the purposes of the thesis as the first assesses the strong and the weak point of competitors in the cross-platform development field. The next section introduces the Flutter and technology behind it. The last three chapters aim at the development of the application. The third section which is about Electronic evidence of revenue explains the reasons behind the need for such a product. Following chapter presents a phase of preparing for development via listing up functionality needed for the final product in requirement's lists, designing a user-friendly graphical interface and planning a logical database structure. In the last chapter the development process is described together with packages used and the issues encountered.

The product that arose from the development is still in the testing phase, as such a product operating with sensitive data has to be strictly correct. For now, it is possible to test on the playground environment to tune out potential flaws. Applications project will be open-source and available for future extensions. As the current project depends on the Android platform, the development of Dart package for EET functionality such as hashing and signing messages or generating a soap request would yield in a platform independence.

Bibliography

1. PRASAD, Rohit; SRIDHAR, Varadharajan. Optimal number of mobile service providers in India: Trade-off between efficiency and competition. *International Journal of Business Data Communications and Networking* (*IJBDCN*). 2008, vol. 4, no. 3, pp. 69–88.
2. *What is Ionic Framework?* [online]. Drifty, 2019 [visited on 2020-03-11]. Available from: `www.ionicframework.com/docs/intro`.
3. WARGO, John M. *Apache Cordova 4 Programming*. Pearson Education, 2015.
4. WILKEN, Jeremy. *Ionic in action: Hybrid mobile apps with Ionic and AngularJS*. Manning Publications, 2016.
5. HUME, Dean Alan. *Progressive web apps*. Manning Publications Co., 2017.
6. BIØRN-HANSEN, Andreas; MAJCHRZAK, Tim A; GRØNLI, Tor-Morten. Progressive web apps: The possible web-native unifier for mobile development. In: *International Conference on Web Information Systems and Technologies*. 2017, vol. 2, pp. 344–351.
7. *Building a Cross-Platform Mobile Team* [online]. Airbnb, 2018 [visited on 2020-03-24]. Available from: `https://medium.com/airbnb-engineering/building-a-cross-platform-mobile-team-3e1837b40a88`.
8. EISENMAN, Bonnie. *Learning react native: Building native mobile apps with JavaScript*. " O'Reilly Media, Inc.", 2015.
9. HANSSON, Niclas; VIDHALL, Tomas. *Effects on performance and usability for cross-platform application development using react native*. 2016.
10. *Technical overview* [online]. Google, 2018 [visited on 2020-03-11]. Available from: `www.flutter.dev/docs/resources/technical-overview`.
11. ZAMMETTI, Frank. *Practical Flutter*. Springer, 2019.

12. MAINKAR, Prajyot; GIORDANO, Salvatore. *Google Flutter Mobile Development Quick Start Guide: Get Up and Running with IOS and Android Mobile App Development*. Packt Publishing Limited, 2019.

13. PICKTON, David W; WRIGHT, Sheila. What's swot in strategic analysis? *Strategic change*. 1998, vol. 7, no. 2, pp. 101–109.

14. *FAQ* [online]. Flutter, 2017 [visited on 2020-07-17]. Available from: `https://flutter.dev/docs/resources/faq`.

15. *Information about registration of sales* [online]. Finanční správa, 2016 [visited on 2020-03-18]. Available from: `https://www.etrzby.cz/cs/english-version-609`.

16. *Formát a struktura údajů o evidované tržbě* [online]. Finanční správa, 2016 [visited on 2020-03-20]. Available from: `https://www.etrzby.cz/assets/cs/prilohy/EET_popis_rozhrani_v3.1.1.pdf`.

17. SOMMERVILLE, Ian; SAWYER, Pete. *Requirements engineering: a good practice guide*. John Wiley & Sons, Inc., 1997.

18. BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. *The Unified Modeling Language User Guide*. Addison-Wesley, 1998. ISBN 0-201-57168-4.

19. NORMAN, Don. *The design of everyday things: Revised and expanded edition*. Basic books, 2013.