VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA PODNIKATELSKÁ
FACULTY OF BUSINESS AND MANAGEMENT

ÚSTAV INFORMATIKY
INSTITUTE OF INFORMATICS

NÁVRH A VÝVOJ POKLADNÍ APLIKACE PRO ANDROID

DESIGN AND DEVELOPMENT OF CASH REGISTER APPLICATION FOR ANDROID

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — Dalibor Mrňávek
AUTHOR

VEDOUCÍ PRÁCE — Ing. Petr Dydowicz, Ph.D.
SUPERVISOR

BRNO 2021

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ FAKULTA PODNIKATELSKÁ

Zadání bakalářské práce

Ústav:	Ústav informatiky
Student:	**Dalibor Mrňávek**
Studijní program:	Systémové inženýrství a informatika
Studijní obor:	Manažerská informatika
Vedoucí práce:	**Ing. Petr Dydowicz, Ph.D.**
Akademický rok:	2020/21

Ředitel ústavu Vám v souladu se zákonem č. 111/1998 Sb., o vysokých školách ve znění pozdějších předpisů a se Studijním a zkušebním řádem VUT v Brně zadává bakalářskou práci s názvem:

Návrh a vývoj pokladní aplikace pro Android

Charakteristika problematiky úkolu:

Úvod

Vymezení problému a cíle práce

Teoretická východiska práce

Analýza problému a současné situace

Vlastní návrh řešení, přínos práce

Závěr

Seznam použité literatury

Cíle, kterých má být dosaženo:

Cílem práce je návrh a vytvoření mobilní aplikace pro firmu. Aplikace má sloužit jako součást účetního a pokladního systému, který firma bude nabízet svým zákazníkům (firmám). Hlavní úlohou aplikace je zaznamenávání běžných pokladních činností (prodej, reklamace, slevy...) a komunikace se serverem (odesílání a přijímání dat o transakcích a produktech).

Základní literární prameny:

GARGENTA, M. Learning Android. Sebastopol, Calif.: O'Reilly, 2011. 245 p. ISBN 14-493-9050-1.

LEE, W. M. Beginning Android application development. Indianapolis, IN: Wiley Pub., 2011. 428 s.

ISBN 978-111-8087-800.

MARTIŠEK, D. Algoritmizace a programování v Delphi. Brno: Littera, 2007. 230 s. ISBN 978-8-85763-37-9.

UJBÁNYAI, M. Programujeme pro Android. Praha: Grada, 2012. 187 s. ISBN 978-80-247-3995-3.

VELTE, A., T. VELTE a R. ELSENPETER. Cloud Computing: praktický průvodce. Brno: Computer Press, 2011. 344 s. ISBN 978-80-251-3333-0.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2020/21

V Brně dne 28.2.2021

L. S.

Mgr. Veronika Novotná, Ph.D. doc. Ing. Vojtěch Bartoš, Ph.D.

ředitel děkan

Abstrakt

Bakalářská práce se zaměřuje na vývoj aplikace pro konkrétní společnost. Zabývá se návrhem uživatelského prostředí, vnitřní databáze aplikace a rozvržením kódové základny. Obsahuje taktéž analýzu trhu, pro který je aplikace vyvíjena. Závěr je věnován zhodnocení nákladů potřebných k vývoji.

Klíčová slova

Android, datové modelování, čistá architektura, ER diagram, třídní model, Flutter

Abstract

Bachelor thesis is focused on an application development for a specific company. It deals with user interface design, design of inner database of the application and with designing a structure of the codebase of the application. It also contains an analysis of the market, for which the application is meant. The conclusion is devoted to the evaluation of development costs.

Key words

Android, data modelling, clean architecture, ER diagram, class diagram, Flutter

Bibliografická citace

MRŇÁVEK, Dalibor. *Návrh a vývoj pokladní aplikace pro Android.* Brno, 2021. Dostupné také z: https://www.vutbr.cz/studenti/zav-prace/detail/135303. Bakalářská práce. Vysoké učení technické v Brně, Fakulta podnikatelská, Ústav informatiky. Vedoucí práce Petr Dydowicz.

Čestné prohlášení

Prohlašuji, že předložená bakalářská práce je původní a zpracoval jsem ji samostatně. Prohlašuji, že citace použitých pramenů je úplná, že jsem ve své práci neporušil autorská práva (ve smyslu Zákona č. 121/2000 Sb., o právu autorském a o právech souvisejících s právem autorským).

V Brně dne 10. května 2021

.....

podpis autora

OBSAH

ÚVOD

První zařízení, které bychom mohli nazývat termínem „smartphone“ vzniklo už v roce 1992. Ačkoli tehdy nezaznamenalo velký úspěch a bylo spíše míněno jako demonstrace tehdejších technologických schopností, nyní, bezmála o 30 let později, se dotykem ovládaná zařízení stala díky svojí intuitivnosti téměř neodmyslitelnou součástí života v zemích prvního světa. Z toho důvodu jsem si vybral návrh mobilní aplikace jako téma své bakalářské práce pro konkrétní společnost.

První kapitola bude věnována teoretickým východiskům, ze kterých bude vycházet řešení popsané ve třetí kapitole. Jedná se o popis frameworku pro vývoj aplikací Flutter a jazyku Dart, který je využíván při vývoji v něm a vývojových prostředí, ve kterých lze tuto technologii použít. Taktéž se bude věnovat popisu metod datového modelování a technologií, které jsou využívány při ukládání dat. Dále obsahuje podkapitolu o principech, podle kterých je nutné rozvrhnout kódovou základnu aplikace, aby byl její vývoj udržitelný. Na závěr je obsažena aktuální analýza trhu s mobilními zařízeními.

Druhá kapitola se zaobírá představením firmy B2C Support, pro kterou bude tato aplikace vyvíjena, služby, které nabízí a popis platformy informačních toků ve firmě. Dále budou představeny její silné a slabé stránky na trhu, bude provedena analýza konkurence a bude zmapován trh operačních systémů v souvislosti s primárním druhem mobilního zařízení, na kterém bude tato aplikace distribuována zaměřeným na Evropské prostředí.

Poslední kapitola je určena konkrétnímu návrhu aplikace. V úvodu bude stručně představena a v další části bude poskytnutý kompletní přehled uživatelského prostředí s popisem ovládacích prvků. V další části bude navržena databáze s popisem jednotlivých entit a atributů a v poslední části bude navržena kódová základna pro vybranou funkcionalitu. Závěr této kapitoly je určen ekonomickému zhodnocení a přínosům práce.

VYMEZENÍ PROBLÉMU A CÍLE PRÁCE

Cílem této bakalářské práce je navrhnout mobilní aplikaci pro firmu B2C Support, která se zaobírá vývojem a poskytováním IT řešení svým odběratelům. Nejdříve bude nutné analyzovat firmu, její pozici na trhu, služby, které dodává svým zákazníkům a poté samotný trh z hlediska konkurence a profilu možných zákazníků. Poté bude navržena samotná aplikace, konkrétně uživatelské prostředí, databáze a kódová základna. Přínosem aplikace bude snadnější integrace již vyvinutých řešení a poskytnutí systému odběratelské firmě, který bude moci naimplementovat do svých prodejen pro zkvalitnění svých služeb.

V aplikaci bude implementovaný systém pro přihlášení konkrétní firmy a zaměstnance, dále bude aplikace obsahovat systém pro výběr zboží, správu zákaznických účtů, správu otevřených účtů a platbu. Taktéž bude obsahovat modul pro účetní uzávěrku a správu již vystavených dokladů.

Mezi vedlejší cíle patří analýza firmy a jejích nabízených služeb analýza trhu s pokladními aplikacemi a analýza trhu s operačními systémy na zařízeních a regionu, pro který je vyvíjená aplikace určena, aby bylo patrné, jaké technologie a postupy bude vhodné zvolit, aby mohla být aplikace vyvinuta co nejrychleji a zároveň aby bylo minimalizováno riziko časových prodlev při vývoji a nutnosti přepisovat kód a plýtvat tak zdroji.

1 TEORETICKÁ VÝCHODISKA PRÁCE

V teoretické části jsou popsané informace potřebné k vyjasnění pojmů v této bakalářské práci. Bude představena sada vývojových nástrojů Flutter, která je určena mimo jiné k vývoji mobilních aplikací, jazyk Dart, který Flutter používá. Dále bude popsaná metoda analýzy SWOT, jazyk UML, metoda na tvorbu diagramů a databázová teorie. Bude představena metoda čisté architektury a důležitost barev a rozmístění prvků pro snadnost používání, dále jen UI Design.

1.1 Flutter

Flutter je sada nástrojů vytvořená firmou Google, určená pro vývoj nativně kompilovaných aplikací pro Android, iOS, web a desktop z jednoho společného kódu. Verze 1.0.0 byla představena na konci roku 2018 [1] a od té doby obrovským tempem narostl na popularitě mezi vývojáři. V porovnání s jazykem Kotlin, vytvořeným českou společností zaměřenou na vývoj softwaru JetBrains, který je určen taktéž k vývoji multiplatformních aplikací, a který je na trhu o téměř tři roky déle [2], má k datu psaní tohoto textu na programátorském fóru GitHub 110 tisíc hvězd a 15,6 tisíc forků [3](vlastní úprava cizího kódu, nebo jeho využití pro vlastní projekt), kdežto Kotlin má hvězd pouze 34,5 tisíc a 4,3 forků.[4]

Obrázek 1: Logo frameworku Flutter
(Zdroj: [1])

1.1.1 Architektura Flutteru

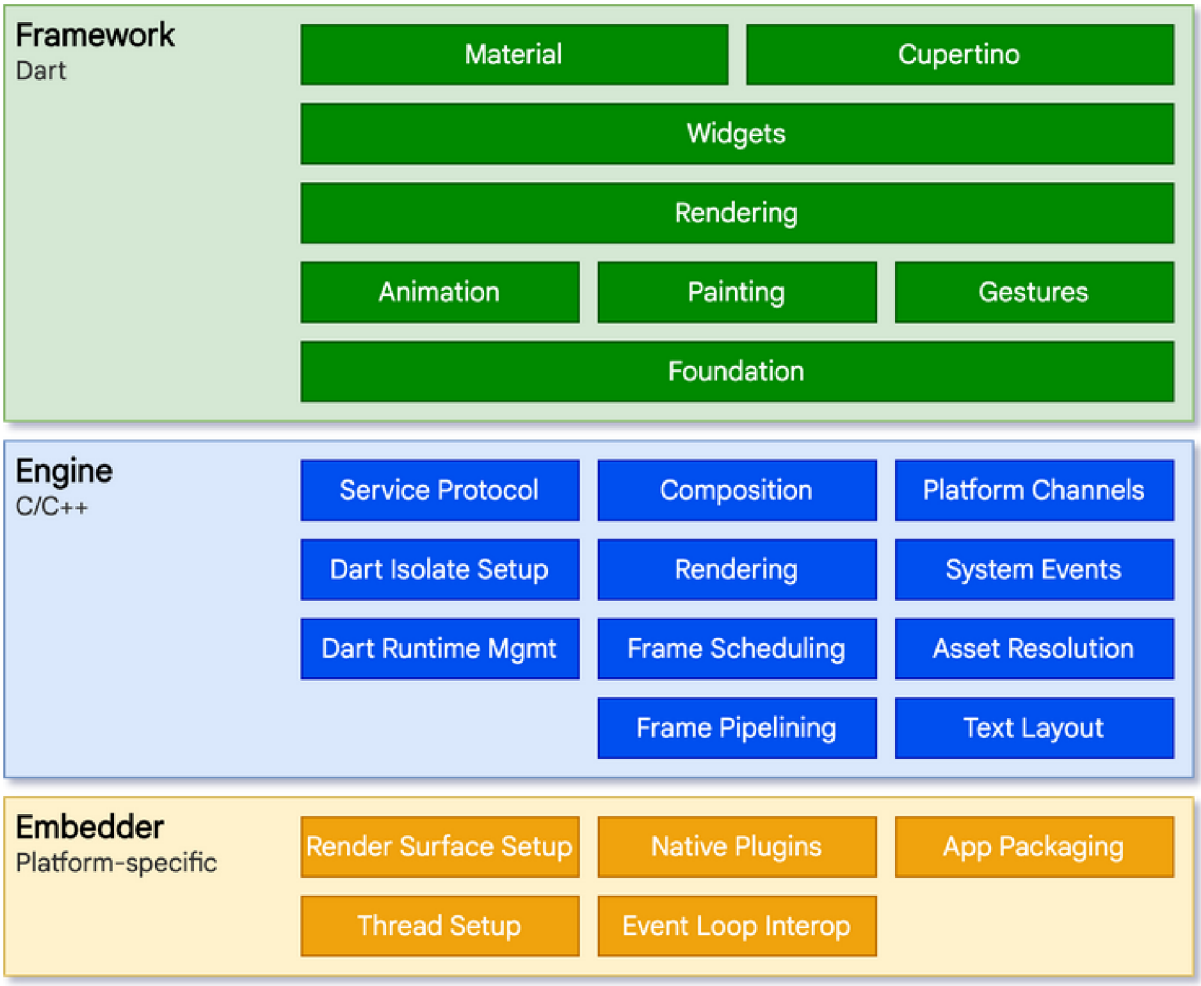
Flutter je framework beroucí inspiraci z javascriptového frameworku *React*. Hlavní myšlenka spočívá v tom, že všechny součásti UI jsou widgety, které se dají do sebe nořit navzájem. [5]

```dart
import 'package:flutter/material.dart';

void main() {
  runApp(
    Center(
      child: Text(
        'Hello, world!',
        textDirection: TextDirection.ltr,
      ),
    ),
  );
}
```

Obrázek 2: Příklad Hello, world! programu ve Flutteru
(Zdroj: [5])

Podle níže uvedeného obrázku je patrné, že kódem psaným v jazyce Dart může vývojář ovládat pouze některé funkce zařízení, pro které je vývoj určen a o zbytek se postará kód psaný v C/C++, který je obsažen v instalačním balíčku Flutter.



Obrázek 3: Přehled vrstev Flutteru
(Zdroj: [6])

Pro kód, který se u každé platformy liší, a jehož účelem má být interakce s nějakou hardwarovou částí telefonu, například požadavek na zjištění stavu baterie, jsou používány *platform channels* – kód napsaný v jazyce specifickém pro danou platformu (Android – Kotlin, iOS – Swift), který je vložen do kódu psaném v Dart.[6]

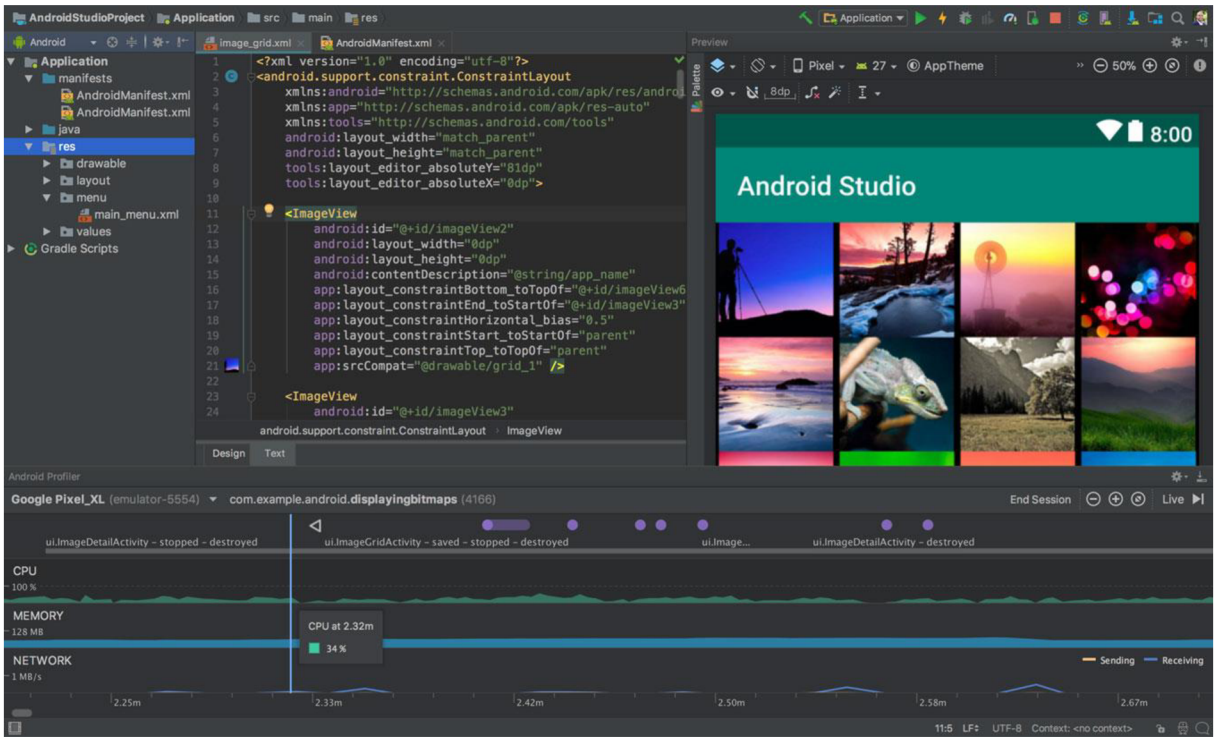
1.2 Vývojové prostředí

Při vývoji aplikací na mobilní platformy ve Flutteru můžeme volit mezi mnoha vývojovými prostředími, které se liší mírou usnadnění psaní kódu a jeho implementací na mobilní zařízení. Mezi nejkvalitnější nástroje patří Android Studio od Google a IntelliJ IDEA od JetBrains. Obě tato prostředí umožňují po instalaci Flutter zásuvných modulů využití jeho přednosti, kterými jsou *Hot Reload* (okamžité promítnutí změn, které nevyžadují opětovnou kompilaci celého programu, provedených v kódu, přímo do běžící aplikace) a *Hot Restart* (stejné jako Hot Reload, jenom s tím rozdílem, že se celá aplikace restartuje).[7]

1.2.1 Android Studio

Android Studio je oficiální vývojové prostředí od firmy Google, které je určeno specificky pro vývoj na Android a je zdarma [8]. První stabilní verze vyšla v prosinci 2014 [9]. Mezi nabízené funkce Android Studio patří:

- Podpora vývoje pro Android Wear
- Integrovaný emulátor OS Android pro spouštění a ladění aplikací
- Zvýrazňování, doplňování a návrhy pro kód [8]



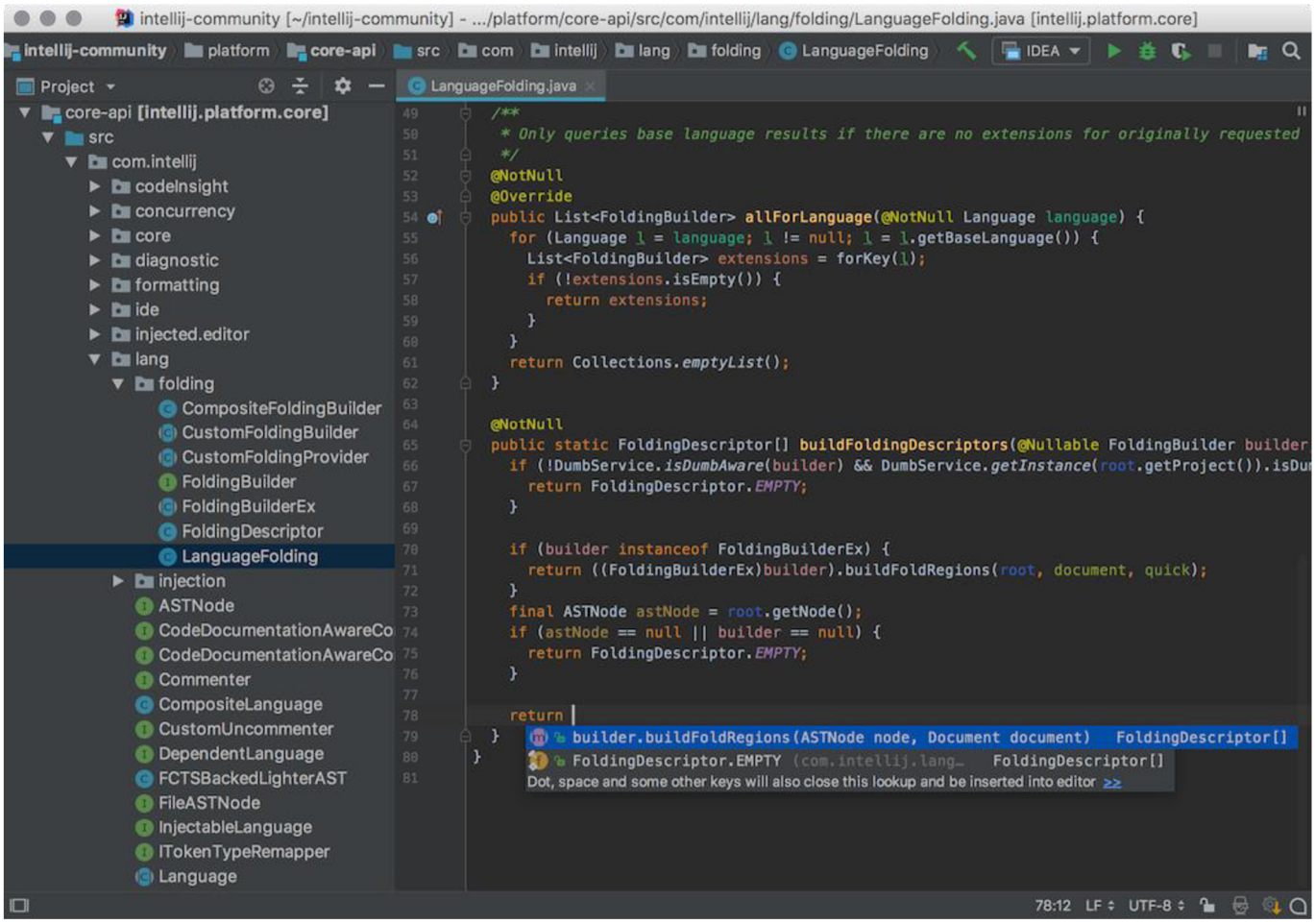
Obrázek 4: Prostředí Android Studio
(Zdroj: [8])

1.2.2 IntelliJ IDEA

Intellij IDEA je vývojové prostředí od české firmy JetBrains. První stabilní verze byla vydána v roce 2001 [10]. Je dostupné pod Apache 2 License [11] a poskytuje i placenou alternativu Intellij IDEA Ultimate, která je nutná pro vývoj aplikací s pomocí Flutter. Tato placená verze je poskytována zdarma studentům a akademickým institucím ke studijním účelům a byla využita pro psaní této práce. [12]
IntelliJ IDEA poskytuje následující funkce pro usnadnění vývoje:

- Chytré doplňování kódu
- Asistence specifická pro Framework

- Predikce kódu
- Zvýrazňování kódu [13]



Obrázek 5: Prostředí IntelliJ IDEA
(Zdroj: [13])

1.3 Dart

Dart je programovací jazyk vyvinutý firmou Google. Je to objektově orientovaný, jazyk, který má syntaxi podobnou jazyku C [14]. První zmínka o něm se objevila v roce 2011 [15]. Je používaný při vývoji pro aplikace na mobilní platformy, pro desktop, server a web.[16]

1.3.1 Vlastnosti

Seznam některých vlastností jazyka Dart:

- Je to objektově orientovaný jazyk založený na třídách – všechno, co se dá dosadit do proměnné, je *objekt*, a každý objekt je instance *třídy*. Čísla, funkce, i *null* jsou objekty, které dědí z třídy Object.
- Dart podporuje generické typy, jako například List<int> (pole integerů).

- Dart se dá přímo kompilovat na Javascript.
- Podporuje top-level funkce, funkce spojené se třídou nebo objektem, a funkce ve funkcích (*nested*, nebo *local* funkce).
- Podporuje aritmetické (sčítaní, odčítání, násobení, dělení), logické (konjunkce, disjunkce, exkluzivní disjunkce), porovnávací (rovná se, nerovná se), přiřazovací (+=, -=, *=…) operátory, dále podmínkový operátor „?" a kaskádový operátor „..".
- Podporuje *null safety*.
- Je to *case sensitive* jazyk – při kompilaci rozlišuje mezi velkými a malými písmeny. [14]

```
void main() {
  print('Hello, World!');
}
```

Obrázek 6: Příklad Hello, World! programu
(Zdroj: [17])

1.3.2 Proměnné a datové typy

Dart podporuje dva různé způsoby typování proměnných – dynamické a statické, a umožňuje sdružovat proměnné do souhrnných objektů, označovaných jako pole. [14]

1.3.2.1 Dynamické typování

Pomocí klíčových slov *var* a *val* se dají vytvářet proměnné, které nemají explicitně definovaný typ a je jim přiřazen automaticky při kompilaci. [14]

1.3.2.2 Statické typování

Pomocí předdefinovaných klíčových slov můžeme vytvořit proměnné s předem daným datovým typem. [14]

Datový typ	Rozsah hodnot	Velikost
int	-2^63 až 2^63–1	64 bitů
BigInt	Limitovaný RAM	Limitovaný RAM

double	-1.7976931348623157e+308 až 1.7976931348623157e+308	64 bitů
String	Hodnoty v UTF-16 kódování	Limitovaný RAM
bool	*true* nebo *false*	

Tabulka 1: Datové typy jazyku Dart
(Zdroj: [14])

1.3.2.3 Pole

V Dartu existují tři typy souhrnných objektů obsahujících více samostatných objektů, které se v jiných jazycích označují slovem *array*. Každý z nich má jiné vlastnosti a s každým se dají provést jiné operace. [14]

Souhrnný typ	Popis	Příklad definování
List	Seřazená skupina objektů	List<int> list = [1,2,3]
Set	Neseřazený soubor unikátních hodnot s jedním datovým typem. Při vložení nesprávného datového typu nastane chyba při překladu.	<String> ovoce = {'jablko', 'kiwi', 'mandarinka'}

Map	Soubor dvojic s předem definovanými datovými typy. Při vložení nesprávného datového typu nastane chyba při překladu.	Map<String, String>
		hlavniMesta = {
		‚CZ' : ‚Praha',
		‚SK' : Bratislava
		}

Tabulka 2: Souhrnné datové typy jazyku Dart
(Zdroj: [14])

1.4 SWOT Analýza

SWOT analýza je nástroj, který je využívaný k ilustrování situace, ve které se organizace nachází předtím, než se rozhodne pro aplikování nějaké strategie. SWOT analýza má čtyři parametry, které jsou vzhledem k organizaci analyzovány. Jsou to věci, které má možnost firma přímo ovlivnit a které souvisí s jejím fungováním – její silné a slabé stránky. Dále to jsou vnější faktory, které jí mohou pomoci nebo uškodit při dosahování jejích cílů – vnější příležitosti a hrozby. [18]

Zkratka SWOT je vytvořena z prvních písmen anglických názvů metrik, které jsou měřeny:

- S – *Strengths* – Silné stránky
- W – *Weaknesses* – Slabé stránky
- O – *Opportunites* - Příležitosti
- T – *Threats* – Hrozby

Sledované faktory se při tvorbě SWOT analýzy zapisují do matice 2x2, každé pole je vyhrazeno jedné sledované vlastnosti firmy. Při analýze SWOT je vhodné využít i jiné analýzy, které poskytnou přesnější obrázek o dílčích veličinách. Je to například model 7 S při analýze interního prostředí, nebo PESTLE analýza při analýze vnějšího prostředí.[18]

Interní	Strengths	Weaknesses
	Silné stránky	Slabé stránky
Externí	Opportunities	Threats
	Příležitosti	Hrozby

Obrázek 7: Vizualizace SWOT analýzy
(Zdroj: Vlastní vypracování podle [18])

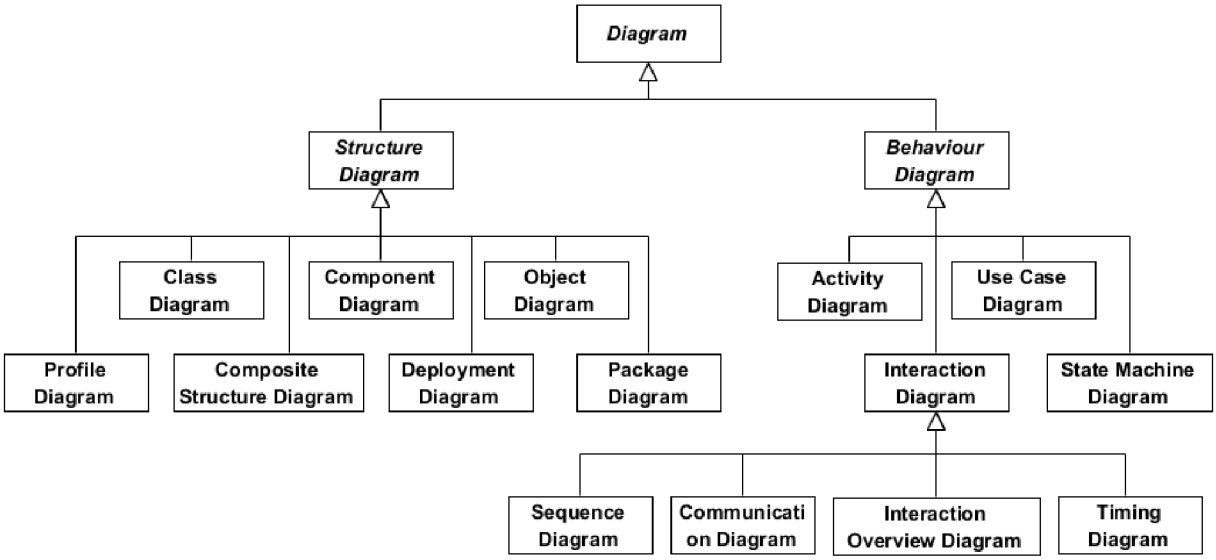
1.5 Datové modelování

Pod názvem Datové modelování se při vývoji softwaru rozumí procesy vytváření datového modelu informačního systému použitím předepsaných technik. Tyto techniky vychází z předpokladu, že informační systém má odrážet reálný svět. Informační systém slouží ke zprostředkování informací, které vznikly v reálném světě, a v systému samotném informace nevzniká. [19]

1.5.1 Unified Modelling Language

UML je jednotný jazyk určený pro použití plánování všech částí informačního systému. Jeho smyslem je dostatečně jasně znázornit popisovanou součást systému i členům organizace, kterých se konkrétní návrh netýká. Dal by se popsat jako všeobecný vizuálně modelovací

jazyk určený pro vizualizaci, specifikaci, konstrukci a dokumentaci softwarového systému. Přes tuto definici se používá i pro popis procesů, které neprobíhají přímo v softwaru, ale i mimo (např. popis prodeje výrobku na pokladně). Jeho způsoby použití se dají rozdělit na dvě skupiny – strukturální a behaviorální. Strukturální diagramy popisují neměnné části systému, např. software, nebo relační databázi. Behaviorální diagramy popisují procesy, ve kterých hraje nějakým způsobem roli lidský faktor. [20]



Obrázek 8: Přehled způsobů použití UML
(Zdroj: [21])

1.5.1.1 Prvky UML

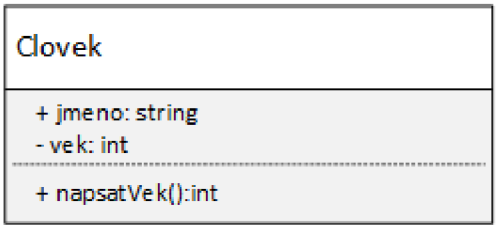
V UML diagramech lze nalézt dva druhy prvků – **Věci,** (things) a **Vztahy** (relationships). Věci označují objekty a koncepty, které jsou součástí systému. Dají se rozdělit na čtyři podkategorie:

- Strukturální
  - Třída (Skupina objektů zodpovědných za stejnou věc)
  - Interface (Definuje soubor operací, které specifikují zodpovědnost tříd)
  - Kolaborace (Interakce mezi prvky)
  - Příklad použití (Soubor akcí směřujících k určitému cíli)
  - Komponent (Fyzická součást systému)
  - Uzel (Fyzická věc existující v systému)
- Behaviorální
  - Interakce
  - State machine

- Seskupující
    - Balíček
- Anotační
    - Poznámka

Vztahy znázorňují, jak spolu dva prvky souvisí a popisují funkcionalitu aplikace. Druhy vztahů jsou následující:

- Závislost
- Asociace
- Generalizace
- Realizace [22]



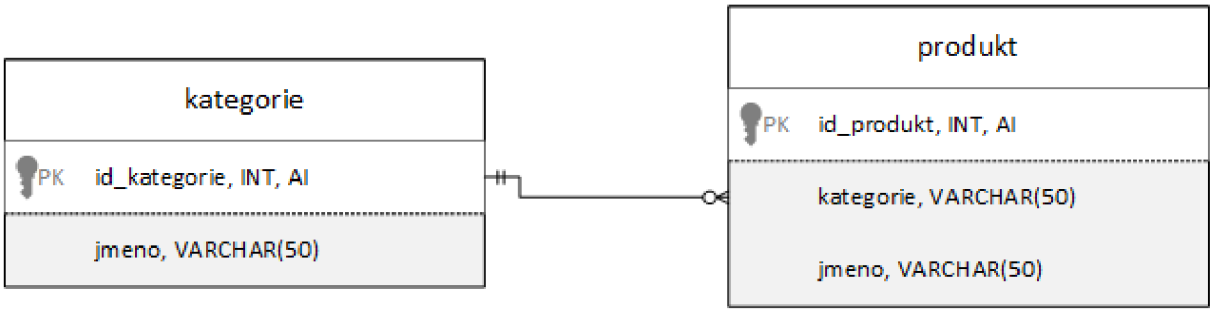
Obrázek 9: Ukázka UML notace
(Zdroj: Vlastní vypracování podle [22])

Na obrázku výše je pro ukázku použití UML znázorněna entita člověk. Do horní části se píší atributy třídy dolní část obsahuje metody, které instance třídy obsahuje. Znaménko před daným atributem nebo metodou značí, jestli je atribut či metoda *public* (je přístupná z kódu mimo třídu), nebo *private*. U metod se do závorek píší argumenty a jejich datový typ. U atributů následuje za dvojtečkou jeho datový typ, u metod je za dvojtečkou datový typ návratové hodnoty.

1.5.2 Entitně-relační diagram

Entitně-relační diagram slouží ke grafickému návrhu databáze, který lépe umožňuje její pochopení pro člověka a tím pádem je možné včas odhalit chyby, které by vznikly při vývoji databáze bez tohoto nástroje. Existují nástroje, které obsahují různé druhy notace pro ERD, a

některé, které jsou schopné na základě diagramu nakresleném člověkem vygenerovat SQL kód, který vygeneruje navrhnutou databázi.[23]



Obrázek 10: Příklad ER Diagramu
(Zdroj: Vlastní vypracování podle [23])

1.5.2.1 Prvky ERD

ER Diagramy se vytváří pomocí symbolů a prvků, mezi které patří:

- **Entita** – V SQL databázi jde o tabulku. Entita značí nějaký objekt, který může mít různé vlastnosti.
- **Atribut** – Jedná se o sloupec tabulky. Každý sloupec nese informaci o jeho názvu, datovém typu, a zdali se jedná o primární nebo cizí klíč.
- **Relace** – Jde o vztah mezi dvěma entitami, které jsou spolu navzájem propojeny. Produkt může spadat do určité kategorie, a kategorie může mít více produktů k ní přiřazené.
- **Kardinalita** – Definuje možný výskyt v jedné entitě, který má vazbu na výskyt v jiné entitě. Například, jedna kategorie může mít víc produktů, a produkt náleží právě jedné kategorii.
- **Primární klíč** – Také zkracovaný na PK, primární klíč je unikátní identifikátor záznamu v tabulce. Jinými slovy, více záznamů než jeden, nemůže mít stejnou hodnotu tohoto atributu.
- **Cizí klíč** – Značí odkaz na primární klíč v tabulce. Je využíván pro identifikaci vztahu mezi entitami. Na rozdíl od primárního klíče nemusí být unikátní, a mohou se vyskytnout v tabulce víckrát, pokud to umožňuje kardinalita vazby.[23]

1.5.3 Jazyk SQL

SQL (vyslovováno „es kjů el“) je jazyk určený pro manipulaci, popis a kontrolu dat. Na rozdíl od procedurálních jazyků (jako je C, nebo Dart), které dávají instrukce počítači krok za

krokem, SQL funguje na principu dotazů, které vrací výsledek z databáze podle určených kritérií. [24]

1.5.3.1 Datové typy

Relační databáze, se kterými SQL pracuje, jsou založeny na tabulkách (také entitách), které obsahují sloupce (atributy) a existují mezi nimi vztahy (relace). Tyto atributy mají daný datový typ, který specifikuje druh metriky použité u atributu a také kolik paměti tento atribut zabere. Kompletní přehled datových typů je zde:

Data Category	Data Type	Size	Value Range
Exact numeric	Bit	1	1, 0, or NULL.
	Tinyint	1	0 to 255
	Smallint	2	-2^15 (-32,768) to 2^15-1 (32,767)
	Int	4	-2^31 (-2,147,483,648) to 2^31-1 (2,147,483,647)
	Bigint	8	-2^63 (-9,223,372,036,854,775,808) to 2^63-1 (9,223,372,036,854,775,807)
	Smallmoney	4	- 214,748.3648 to 214,748.3647
	Money	8	-922,337,203,685,477.5808 to 922,337,203,685,477.5807
	numeric[(p[,s])]	5-17	
	decimal [(p[,s])]	5-17	
Approximate numeric	Float	4-8	- 1.79E+308 to -2.23E-308, 0 and 2.23E-308 to 1.79E+308
	Real/float(24)	4	- 3.40E + 38 to -1.18E - 38, 0 and 1.18E - 38 to 3.40E + 38
Character strings	char [(N)]	N	N = 1 to 8000 non-Unicode characters bytes
	varchar [(N or max)]	N or 2^31-1	N = 1 to 8000 non-Unicode characters bytes Max = 2^31-1 bytes (2 GB) non-Unicode characters bytes
	Text	2^31-1	1 to 2^31-1 (2,147,483,647) non-Unicode characters bytes
Unicode character strings	nchar [(N)]	N	N = 1 to 4000 UNICODE UCS-2 bytes
	nvarchar [(N | max)]	N or 2^31-1	N = 1 to 4000 UNICODE UCS-2 bytes 1 to 2^31-1 (2,147,483,647) UNICODE UCS-2 bytes
	Ntext	2^30-1	Maximum size 2^30 - 1 (1,073,741,823) bytes
Binary strings	binary [(N)]	N	N = 1 to 8000 bytes
	varbinary [(N | max)]	N or 2^31-1	N = 1 to 8000 bytes Max = 0 to 2^31-1 bytes
	Image	2^31-1	0 to 2^31-1 (2,147,483,647) bytes
Other data types	Uniqueidentifier	16	xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx (hex decimal)
	Timestamp	8	binary(8) or varbinary(8)
	rowversion	8	binary(8) or varbinary(8)
	xml	2^31-1	xml([CONTENT | DOCUMENT] xml_schema_collection)
	sql_variant	8016	data type that stores values of various SQL Server-supported data types
	Hierarchyid	892	6*logA*n* bits where n is child node
	Cursor		
	Table		
	Sysname	256	
Date and time	Date	3	0001-01-01 through 9999-12-31
	time [(fractional second precision)]	3 to 5	00:00:00.0000000 through 23:59:59.9999999
	Smalldatetime	4	Date: 1900-01-01 through 2079-06-06 Time: 00:00:00 through 23:59:59
	Datetime	8	Date: January 1, 1753, through December 31, 9999 Time: 00:00:00 through 23:59:59.997
	datetime2 [(fractional seconds precision)]	6 to 8	Date: 0001-01-01 through 9999-12-31 Time: 00:00:00 through 23:59:59.9999999
	datetimeoffset [(fractional seconds precision)]	8 to 10	Date: 0001-01-01 through 9999-12-31 Time: 00:00:00 through 23:59:59.9999999 Time zone offset: -14:00 through +14:00
Spatial	Geography	2^31-1	
	Geometry	2^31-1	

Obrázek 11: Datové typy jazyka SQL
(Zdroj: [25])

1.5.3.2 Ukázka syntaxe

Následující příklad syntaxe vrátí všechny položky z databáze, které odpovídají kritériím.

```sql
SELECT `productName` FROM `kacsel` WHERE -- vyber všechny produkty z tabulky 'kacsel', které:
`price` > 100 -- stojí více než 100 korun
AND -- a (logický operátor, musí platit obě podmínky)
`stock` < 5 -- na skladě je méně než 5 kusů
GROUP BY `sortiment` -- a sdruž je do skupin podle sortimentu
```

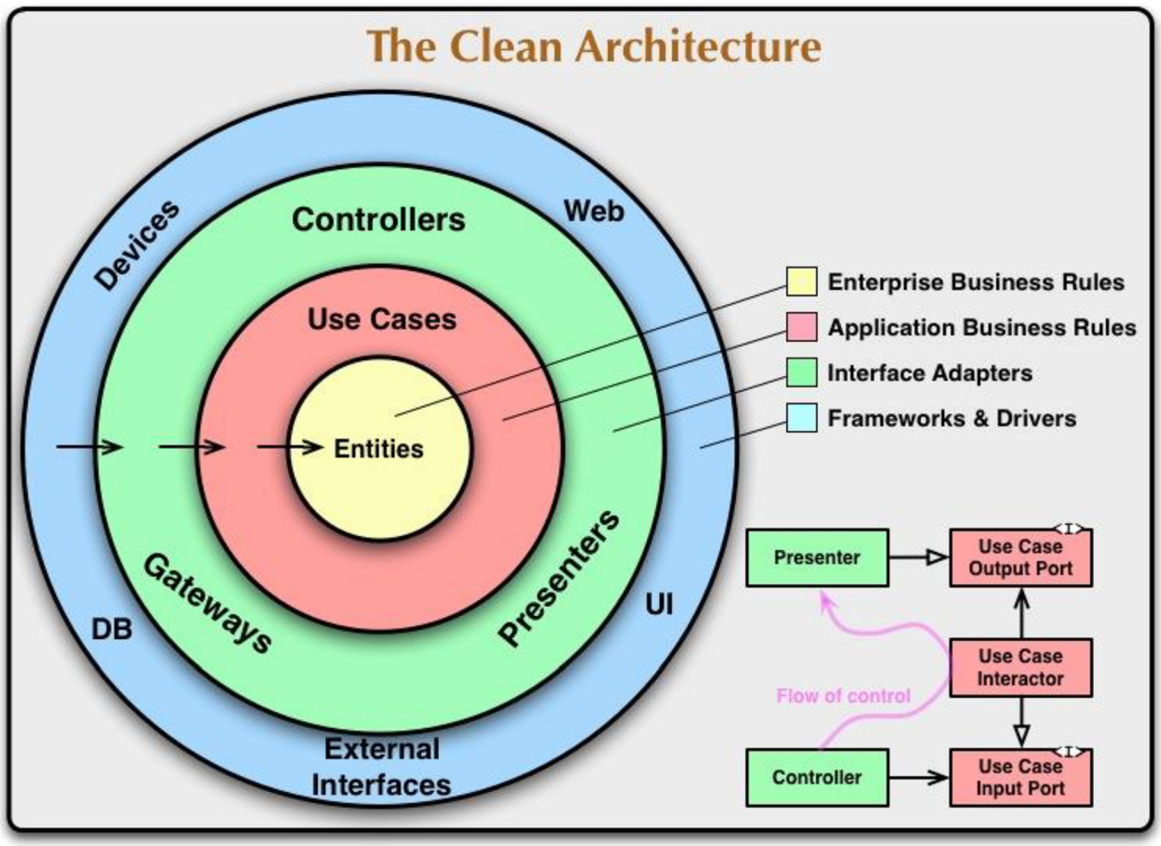
Obrázek 12: Ukázka syntaxe jazyka SQL
(Zdroj: Vlastní vypracování)

1.6 Čistá architektura

Během let vývoje softwaru mnoho firem narazilo na problém, kdy stále rozšiřovaly svoji základnu vývojářů, ale čistý počet nových řádků kódu se snižoval s každým dosaženým milníkem ve vývoji, až nakonec dosáhl asymptoty. Tato situace je naprosto nevhodná z hlediska udržitelnosti, jelikož rostoucí mzdové náklady na vývoj a klesající rychlost vývoje nakonec dostanou produkt do situace, kdy nebude mít smysl do něj dále investovat a firma bude muset draze vyvinout produkt nový.

Příčinou tohoto problému bylo to, že byly kladeny velké nároky na rychlost vývoje, zatímco byla přehlížena uspořádanost a struktura kódu, a v pozdějších fázích vývoje se vývojáři v napsaném kódu ztráceli a vývoj se zpomaloval. Měli totiž za to, že se ke kódu vrátí později a upraví ho tak, aby byl přehledný a strukturovaný. To se ale nedělo a množství narůstalo a kódová základna se stávala stále více nepřehledná.

Tento problém řeší implementace principů čisté architektury do návrhu softwaru ještě předtím, než se začne s vývojem. **Čistá architektura** je soubor pravidel, podle kterých se má navrhovat každý software, aby byl dobře otestovaný, funkční a škálovatelný bez nutnosti exponenciálně zvětšovat nároky na výrobní prostředky firmy. [26]



Obrázek 13: Schéma čisté architektury
(Zdroj: [27])

Při vývoji se podle Čisté architektury uplatňuje princip SOLID, který je složený z pěti pravidel:

1.6.1 SRP – Single Responsibility Principle

Single responsibility principle je pravidlo, jež je popsáno slovy *„A module should be responsible to one, and only one actor"* a značí rozdělení kompetencí a příslušností jednotlivých modulů. Každá třída by měla obsahovat funkce, které jsou spjaty s činností právě jedné pozice. Zdůvodněním tohoto pravidla jsou problémy, které vzniknou, když více druhů zaměstnanců používá stejné funkce. Změna funkcionality na popud jedné strany ovlivní všechny subjekty, které ji používají, a to často aniž by se o tom dozvěděli a organizace pak pracuje s nesprávnými daty. [26]

1.6.2 OCP – Open-Closed Principle

OCP je jeden ze základních kamenů architektury softwaru. Cílem je udělat systém snadný na rozšíření, bez nutnosti provádět rozsáhlé změny v již napsaném kódu. Je ho dosaženo

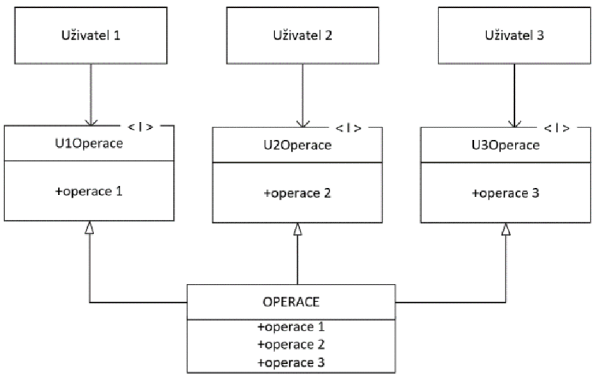
rozdělením systému do komponentů a seřazení těchto komponentů do hierarchie závislosti která chrání komponenty vyšší úrovně před změnami v komponentech nižší úrovně. [26]

1.6.3 LSP – Liskov Substition Principle

Podle LSP se pro speciální případy použití určité třídy vytváří podtřída, která z ní dědí, místo toho, aby se jednoduše ošetřil tento speciální případ podmínkovou klauzulí *if*. Ošetření každého takového případu by nevyhnutelně zahltilo zdrojový kód a učinilo by jej nepřehledným. [26]

1.6.4 ISP – Interface Segregation Principle

Podle tohoto principu je nutné vytvořit interface, které zajistí komunikaci mezi moduly, přičemž se zároveň postará o to, aby na sobě zároveň nebyly přímo závislé. Jednotlivé moduly by neměly přímo záviset na funkcích, které nepoužívají. Pokud bychom měli Systém S, který by přímo závisel na frameworku F, a ten by byl spjatý s databází D, čekaly by nás problémy. Předpokládejme, že D obsahuje funkcionality, které F nepoužívá a tím pádem nejsou důležité pro S. Změna těchto funkcionalit by znamenala nutnost znovu zavést F a tím pádem i S. V horším případě by chyby v D způsobily selhání F a poté S. [26]



Obrázek 14: Příklad segregace pomocí interface (UML)
(Zdroj: Vlastní vypracování podle [26])

1.6.5 DIP – Dependency Inversion Principle

Tento princip znamená, že nejvíce flexibilní systémy jsou ty, jejichž závislosti zdrojového kódu jsou složeny z abstrakcí. V praxi to znamená, že *import* klauzule by měly odkazovat pouze na třídy, které obsahují interface, abstraktní třídy nebo jiné druhy abstraktních deklarací. [26]

1.7 UI Design

UI (zkratka z anglického *User Interface*) Design je obor zabývající se návrhem uživatelského prostředí softwaru, aby bylo zajištěno co nejsnadnějšího používání ze strany uživatele. UI Design do sebe zahrnuje všechny formy ovládání softwaru uživatelem. Patří mezi ně:

- Grafické uživatelské prostředí (GUI) – Uživatelé interagují s vizuální reprezentací programu skrze ovládací prvky na obrazovce. Příkladem GUI může být uživatelské prostředí desktopové aplikace.
- Ovládání hlasem – Uživatelé ovládají software pomocí hlasových příkazů. Nejvíce je tento styl ovládání použit u chytrých asistenčních aplikací – Siri na iPhony, Google Assistant u Androidu.
- Ovládání gesty – Uživatel interaguje skrze gesta provedená v prostoru. Tento styl ovládání se používá u her ve virtuální realitě.[27]

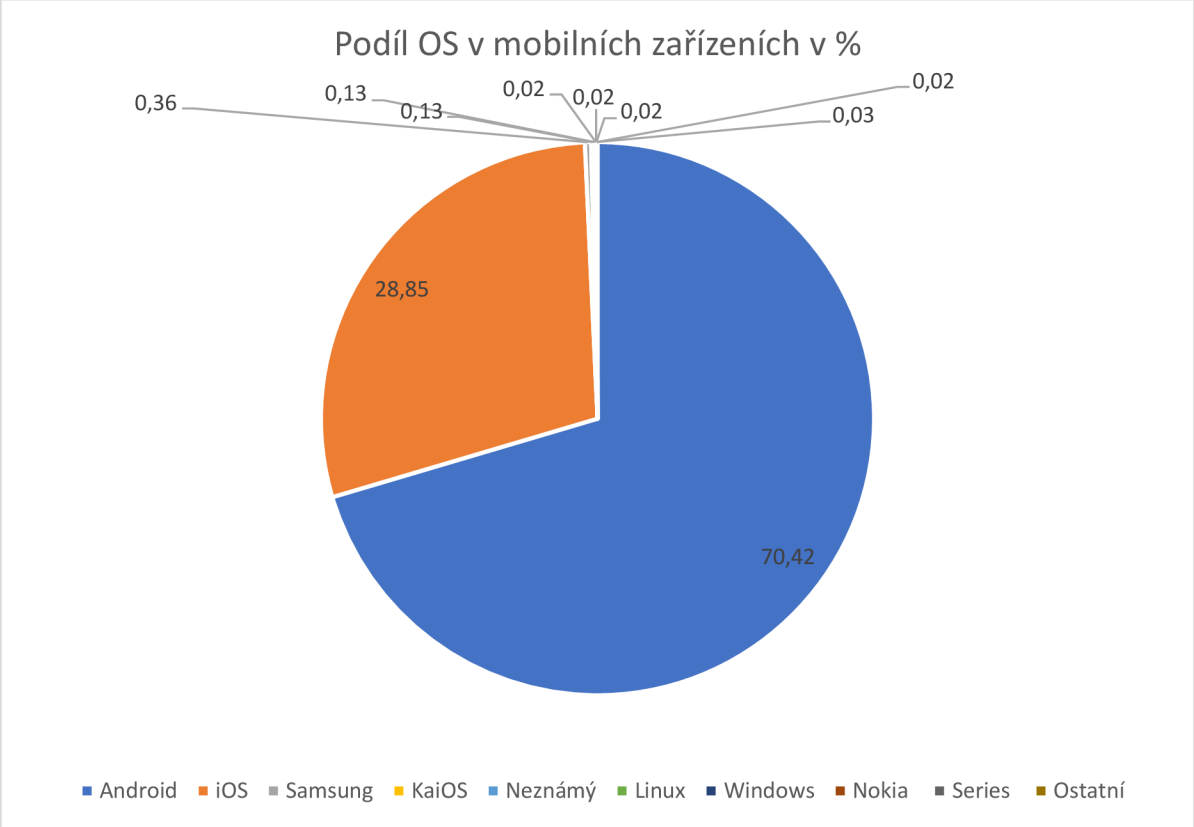
1.7.1 Adobe XD

Program Adobe XD vyvinutý firmou Adobe slouží k návrhu a prototypování uživatelského prostředí webových a mobilních aplikací. Je k dispozici pro macOS a Windows v podobě, ve které je možné uživatelské prostředí navrhnout. Existují i aplikace pro iOS a Windows, které slouží k zobrazení funkčního prototypu vytvořeného v desktopové aplikaci na mobilních zařízeních. [28]

Obrázek 15: Logo aplikace Adobe XD
(Zdroj: [28])

1.8 Operační systémy v mobilních zařízeních

Na trhu operačních systémů mobilních zařízení dominují především operační systémy iOS od firmy Apple a Android od konkurenčního Googlu, který je oblíbený mezi výrobci telefonů díky své podpoře ze strany výrobce a modifikovatelnosti. Až na výjimky takřka všichni distributoři mobilních telefonů vytváří nadstavby Androidu, kterými se snaží odlišit na trhu s mobilními zařízeními od konkurence. Na grafu níže je vyobrazený podíl operačních systémů celosvětově ve všech mobilních zařízeních (mobilech i tabletech) k březnu 2021. Je z něj patrná jasná dominance Androidu na trhu s mobilními zařízeními. Jediný důvod, proč je iOS schopný alespoň částečné konkurence je to, že je výhradně využíván pro produkty firmy Apple.[29]



Graf 1: Podíl operačních systémů v mobilních zařízeních celosvětově k březnu 2021
(Zdroj: Vlastní vypracování podle [29])

1.8.1 Android

Operační systém Android je založen na Linuxovém jádru (Linux kernel) a je vytvořený a spravovaný firmou Google. Je navržen především pro mobilní zařízení, jako jsou mobilní telefony a tablety. Aplikace na Android jsou distribuovány především skrze službu Google

Play. Díky tomu, že je open-source se stal nejrychleji se rozšiřujícím operačním systémem a těší se oblíbenosti u vývojářů i zákazníků. Vývojáři ho mohou jednoduše upravovat a vytvářet vlastní nadstavby Androidu anebo ho upravovat tak, aby odpovídal měnícím se požadavkům technologií. Mezi klíčové funkce Androidu se řadí: Aplikační Framework, virtuální stroj Dalvik, integrovaný prohlížeč, optimalizovaná grafika, podpora SQLite, podpora médií, technologie GSM, Bluetooth, Edge, 3G, Wi-fi, fotoaparát a GPS atd. Aby Google usnadnil vývoj aplikací pro Android, poskytuje Android Software Development Kit. [30]

Vývoj aplikací může být realizován pomocí programovacích jazyků Java, Kotlin, nebo Dart.

1.8.1.1 Architektura Androidu

Operační systém Android je soubor softwarových komponentů. Hlavními prvky jsou:

Linux kernel – Linuxové jádro je úplným základem celého softwarového balíčku. Celý operační systém je postaven nad touto vrstvou. Obsahuje ovladače hardwaru, spravuje paměť, síťové operace a napájení. Android interaguje s hardwarovými komponenty zařízení (fotoaparát, klávesnice, displej atd) skrze tuto vrstvu. Dále zajišťuje zpracování procesů. [31]

Native libraries – Tato komponenta je napsaná v jazyce C, nebo C++. Umožňuje Androidu zpracovat různé typy dat. Mezi důležité knihovny se řadí:

- SQLite – Relační databáze přístupná všem aplikacím
- WebKit – Engine prohlížeče, který zpracovává HTML obsah
- Media Framework – Umožňuje vytváření a přehrávání souborů v audio a video formátu (např. MP3, AAC, JPG…) [31]

Android Runtime – Pracuje na stejné vrstvě jako Native libraries. Android Runtime se skládá z virtuálního stroje Dalvik a ústředních knihoven Javy. Dalvik VM je typ Java Virtual Machine a jeho využití spočívá v poskytování prostředí pro chod aplikací, kde každá aplikace může spouštět své vlastní procesy, každá ve své instanci tohoto virtuálního stroje. [31]

Application Framework – Aplikační framework obsahuje knihovny, které mohou být vývojáři přímo využívány v kódu, a které poskytují přístupový interface aplikacím ve formě Java tříd. [31]

1.8.2 iOS

Podobně jako u Androidu, i iOS slouží jako prostředník mezi hardwarem a aplikacemi. Je využívaný na zařízeních iPhone, iPad, iPod Touch a také zařízeních Apple TV. Aplikace na iOS jsou distribuovány skrze službu App Store.

Vývoj aplikací na iOS může být realizován pomocí jazyků Objective-C, nebo Swift.

1.8.2.1 Architektura iOS

Architektura iOS je podobně jako Android rozdělena do více vrstev, které dohromady fungují jako prostředník mezi hardwarem a aplikacemi. Každá vrstva obsahuje jasně definovaná systémová rozhraní, skrze které může tato komunikace probíhat. Nižší vrstvy poskytují základní systémové služby, které jsou využívány aplikacemi a vyšší vrstvy poskytují služby grafického rázu.[32] Nejdůležitější vrstvy od nejnižší jsou:

Core OS

Tato vrstva obsahuje přístupové rozhraní k funkcím nejnižší úrovně, jako jsou například funkce ovládání Bluetooth, Accelerate Framework, Framework bezpečnostních služeb, Framework externích zařízení. [33]

Core Services

Technologie v Core Services poskytují nezbytné služby aplikacím, ale nemají žádný vliv na uživatelské rozhraní. Všeobecně se dá říct, že tyto technologie závisí na frameworku a technologiích použitých v nižší vrstvě – Core OS. Mezi důležité služby poskytované touto vrstvou se řadí Address Book, Core Data, Core Foundation, Foundation, Quick Look, Social, Security a WebKit. [34]

Media Layer

Media Layer se stará o poskytování grafických efektů, které mohou být využívány vývojáři ke zkvalitnění uživatelského zážitku aplikací. Obsahuje 2 D a 3 D grafiku, obrázkové efekty, a audio a video funkcionality, kterých můžou využívat aplikace. [35]

Cocoa Application Layer

Cocoa Application Layer je vrstva odpovědná za vzhled aplikací a jejich chování na základě uživatelského vstupu. Mnoho funkcionalit důležitých pro OS X uživatelský zážitek, jako například Centrum oznámení, režim celé obrazovky a automatické ukládání, je implementováno v této vrstvě. [36]

2 ANALÝZA SOUČASNÉHO STAVU

V této kapitole se budu věnovat představení firmy, službám, které nabízí svým zákazníkům, do kterých budu implementovat navrhovanou aplikaci, a analýze informačního toku ve firmě. Druhá kapitola bude zaměřena na trh s podobnými službami, a na postavení firmy z hlediska tohoto trhu.

2.1 Charakteristika společnosti

2.1.1 Informace o společnosti

Název společnosti: B2C Support s.r.o.
Adresa: Hodslavice 143, 742 71
Právní forma: společnost s ručením omezeným
IČO: 29448948
DIČ: CZ29448948
Rok založení: 2012

Obrázek 16: Logo B2C Support s.r.o.
(Zdroj: [37])

2.1.2 Nabízené služby

Firma B2C Support vznikla v roce 2012 v Pardubicích, a soustředí se především na poskytování služeb na trhu informačních technologií. Mezi její nabízené služby patří:

- Grafický design
- 3D Vizualizace staveb a bytů
- Produktová videa
- Poskytování business intelligence služeb
- Vývoj a správa informačních systémů pro firmy [37]

V oblasti IT jsou hlavním produktem firmy nástroje Sybila a SiD. Které budou přiblíženy v následujících podkapitolách.

2.1.2.1 SYBILA

Business Intelligence (BI) nástroj SYBILA, je nástroj vyvíjený firmou za účelem sledování a vyhodnocování dat generovaných činnostmi odběratelských firem. Je zaměřen na specifika zásilkového obchodu a jeho cílem je poskytovat ucelenou řadu BI nástrojů, které umožní analytické výpočty a automatizaci rozhodování. Bude se skládat ze čtyř modulů:

- Integrator
- Monitor
- Testace
- Prediktor

Jako celek bude tato služba zaručovat sběr dat z činností zákazníků, předpovídání potřeb zboží a chování klientů. Výstup bude formou služby Web Services a konzumentem těchto služeb budou různé moduly stávajícího systému uživatele, například ERP, e-shop, e-mailový klient, nebo CRM[38]

2.1.2.2 SiD

SiD je informační systém navržený speciálně pro zásilkové firmy. Je určen ke zpracovávání velkého množství objednávek denně, podporuje CRM, umožňuje optimalizaci logistických procesů od příjmu zboží od dodavatelů až po expedici, zpracování nepřevzatých zásilek a reklamací včetně vrácení peněz na účet. Mimo to poskytuje i mnoho dalších funkcí, které usnadňují řízení podniku.[39]

2.2 Analýza trhu

2.2.1 Analýza konkurenčních služeb a aplikací

Na trhu s EET aplikacemi se od dob jejího zavedení pohybuje spousta podnikatelských subjektů, kteří vyvíjí různá řešení mající různé funkcionality. Některé z nich jsou aplikace zaměřené čistě jen na vystavování účtenek a odesílání dat na EET, jiné poskytují i různé další možnosti využití.

2.2.1.1 Dotykačka

Jedním z prvních na trhu a nejvíce rozšířených EET řešení je aplikace Dotykačka. Prodává se ve formě mobilního zařízení s tiskárnou, malého i velkého statického terminálu s oddělenou tiskárnou, nebo přenosného platebního terminálu s vlastní tiskárnou účtenek. Na všech těchto zařízeních běží upravená verze operačního systému Android. Aplikace Dotykačka umožňuje prodej na pokladně, správu skladových zásob, rezervační systém stolů, pokojů, nebo služeb. [40]

Obrázek 17: Logo společnosti Dotykačka
(Zdroj: [40])

2.2.1.2 ProfiÚčtenka

ProfiÚčtenka umožňuje pořízení přenosného terminálového zařízení vlastní tiskárnou, nebo použití na vlastním zařízení s tiskárnou účtenek připojenou odděleně. Oproti Dotykačce sází na jednoduchost použití, což ocení převážně méně technicky zdatní uživatelé, nebo ti, kteří nevyužijí všechny funkce složitějšího pokladního systému. Nabízí tisk účtenek, odeslání do EET, tvorbu uzávěrek a podporuje platby kartou.[41]

2.2.1.3 PlayStore

Kromě výše zmíněných aplikací se po vyhledání klíčového slova EET na Google Play Store objeví 44 relevantních výsledků nabízejících EET pokladny od různých vývojářů. Tyto aplikace jsou patrně omezeny pouze na funkci vystavování účtenek a odesílání údajů do centrálního systému České republiky a nenabízí žádné další funkce.

2.2.2 SWOT analýza firmy na trhu

V této části jsem se rozhodl analyzovat trh s aplikacemi na mobilní zařízení z pohledu společnosti B2C Support a jejich vstupu na tento trh. Tato analýza by měla sloužit k ustanovení jasné představy o podobě možností, které firma při vstupu na nový trh má, a k usnadnění dalšího rozhodování.

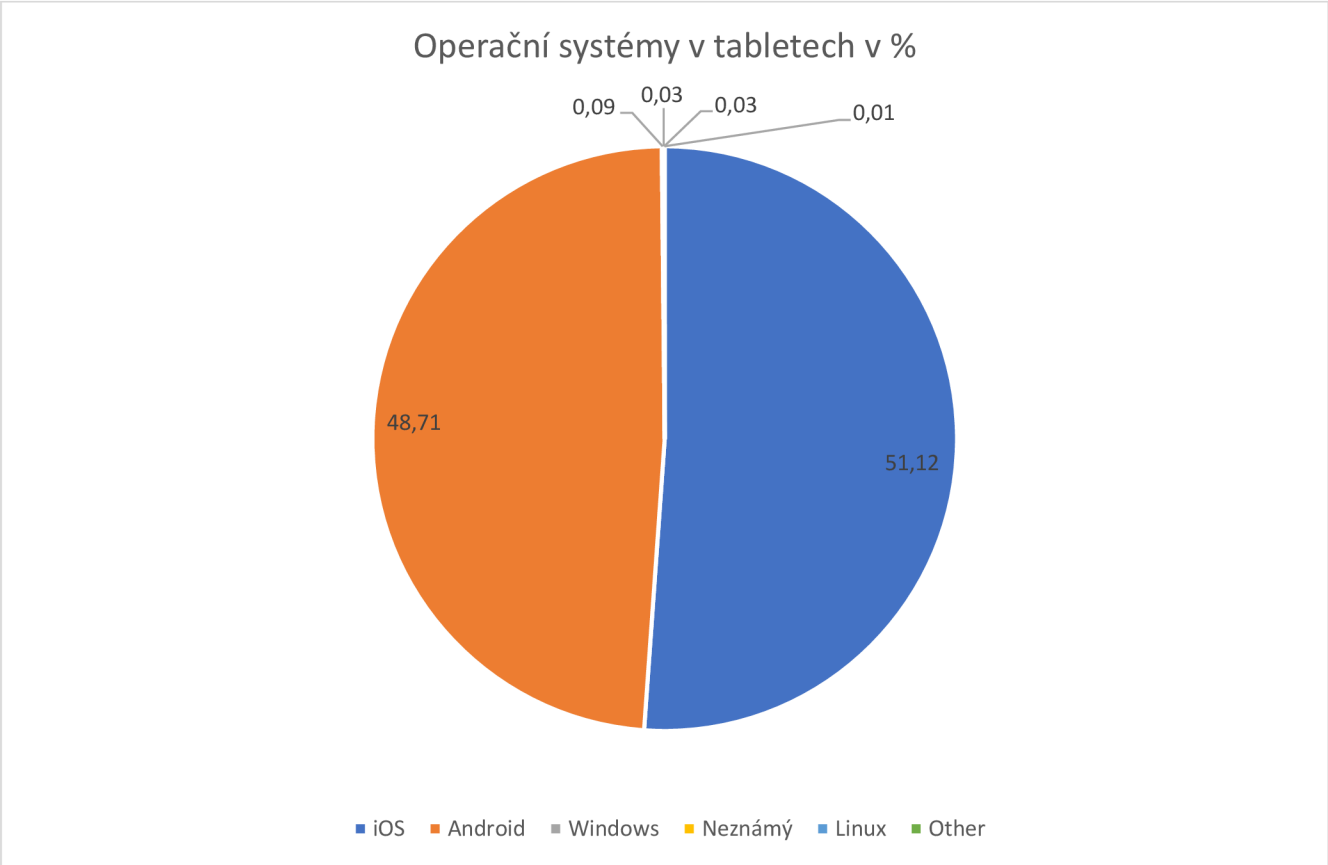
Silné stránky	Slabé stránky
Stálí zákazníci Nabídka netradičních služeb Know-how Dobré technické zázemí	Vysoké náklady na tvorbu aplikace Velikost firmy Slabý marketing
Příležitosti	**Hrozby**
Vstup na velký trh ČR Snadný vstup na trh Prudký rozvoj technologií v době koronavirové krize	Velká konkurence Možné zpoždění vývoje Zanikající firmy

Ve SWOT analýze jsem se zaměřil na silné a slabé stránky společnosti. Společnost má stálé zákazníky, kteří už nyní využívají služeb společnosti v oblasti poskytování a údržby informačního systému a business intelligence. Firma používá technologie od IBM, které mají zajištěnou téměř zpětnou kompatibilitu, a tím pádem je minimalizováno riziko výpadku systému při aktualizaci nějaké jeho části. Mezi slabé stránky patří zejména velikost firmy, která jí znemožňuje rychle a efektivně vytvářet softwarové řešení, aby se rychle přizpůsobila technologickým trendům doby. Dále mezi slabé stránky řadím marketing, protože se firma marketingovým aktivitám příliš nevěnuje i přesto, že by jí mohly přinést nové zákazníky.

Mezi příležitosti vstupu firmy na trh s aplikacemi patří neustále se zvětšující poptávka po technologických řešeních umožňujících efektivní řízení podniku a rozvoj technologií v době pandemie. Jako hrozbu můžeme vyhodnotit velkou konkurenci, která již poskytovala podobná řešení dříve, možné zpoždění vývoje z důvodu snahy o minimální dobu strávenou společně v kanceláři kvůli ochraně zdraví, a momentálně špatně se vyvíjející ekonomická situace, která přinese zánik mnoha firem a tím pádem i zmenší počet potenciálních zákazníků.

2.2.3 Analýza trhu operačních systémů v tabletech v Evropě

Vzhledem k účelu, za jakým bude aplikace používána, a plánovanému trhu je vhodné se zaměřit na analýzu trhu operačních systémů v tabletech se zaměřením na Evropský trh. Dle aktuálních dat (k březnu 2021) není rozdíl mezi dvěma hlavními hráči na trhu mobilních zařízení co do počtu zařízení, která jejich operační systémy využívají, a dokonce má lehce navrch operační systém iOS.



Graf 2: Podíl operačních systémů v tabletech v Evropě k březnu 2021
(Zdroj: Vlastní vypracování podle [42])

2.3 Závěr analýzy

Ze získaných informací je patrné, že je vhodné aplikaci vyvíjet jako multiplatformní aplikaci pro iOS i Android. Dále je vhodné vyvinout aplikaci tak, aby bylo snadné ji rozšiřovat a přidávat funkce, které budou zákaznické firmy požadovat, kvůli velmi dobré možnosti integrace s již vytvořenými řešeními, která firma poskytuje svým zákazníkům.

3 VLASTNÍ NÁVRH ŘEŠENÍ

Náplní této kapitoly bude vlastní návrh řešení pro společnost B2C Support. Aplikace má sloužit jako tzv. point-of-sale, musí tedy poskytovat svým uživatelům intuitivní a přehledné uživatelské prostředí, pomocí kterého budou moci evidovat prodeje na pokladně a platby v hotovosti či kartou a interagovat s připojeným příslušenstvím, jako je čtečka čárových kódů, pokladna, nebo bankovní terminál. V samotném návrhu aplikace se zaměřím nejdříve na uživatelské rozhraní, poté databázové schéma aplikace a rozvržení části kódové základny, které bude splňovat standardy čisté architektury. Další částí této kapitoly bude ekonomické zhodnocení, kde bude zohledněn čas potřebný na vývoj a s tím spojené finanční náklady. V závěru se zaměřím na přínosy, které toto řešení přináší.

3.1 Návrh aplikace

Aplikace má být dílčí součástí většího systému zpracovávajícího data a poskytujícího služby v oblasti business intelligence. Má fungovat jako point-of-sale, což zahrnuje veškeré služby běžné pokladny, jimiž jsou autentizace uživatele, služby samotné pokladny – ruční výběr produktů, načtení čárového kódu zboží, správa otevřených účtů, načtení zákaznických karet, případně přiřazení účtu konkrétnímu zákazníkovi, synchronizace s daty o firmě, synchronizace kusovníku, otevření pokladny, platební příkaz na terminál, výpis dokladů, storno dokladu, uzávěrka pokladny a uzávěrka měsíce s druhovým a finančním přehledem. Měla by obsahovat přehledné uživatelské prostředí, které je vhodné pro uživatele všech kategorií, tedy i těch, kteří nejsou zběhlí v používání technologií.

Primárním cílovým operačním systémem je Android, pro vývoj aplikace jsem zvolil Flutter, který umožňuje flexibilitu v oblasti kódové základny a umožňuje rychlé rozšíření i na trh se zařízeními s operačním systémem iOS.

Pro účely demonstrace návrhu a vývoje aplikace je primárním zařízením, pro které je aplikace určena tablet s rozlišením 1280x800px a operačním systémem Android, který je jedním z nejrozšířenějších typů zařízení tablet, otočeném horizontálně, kupříkladu uchyceném na stojanu pokladny nebo položený na stole (dále „terminál“).

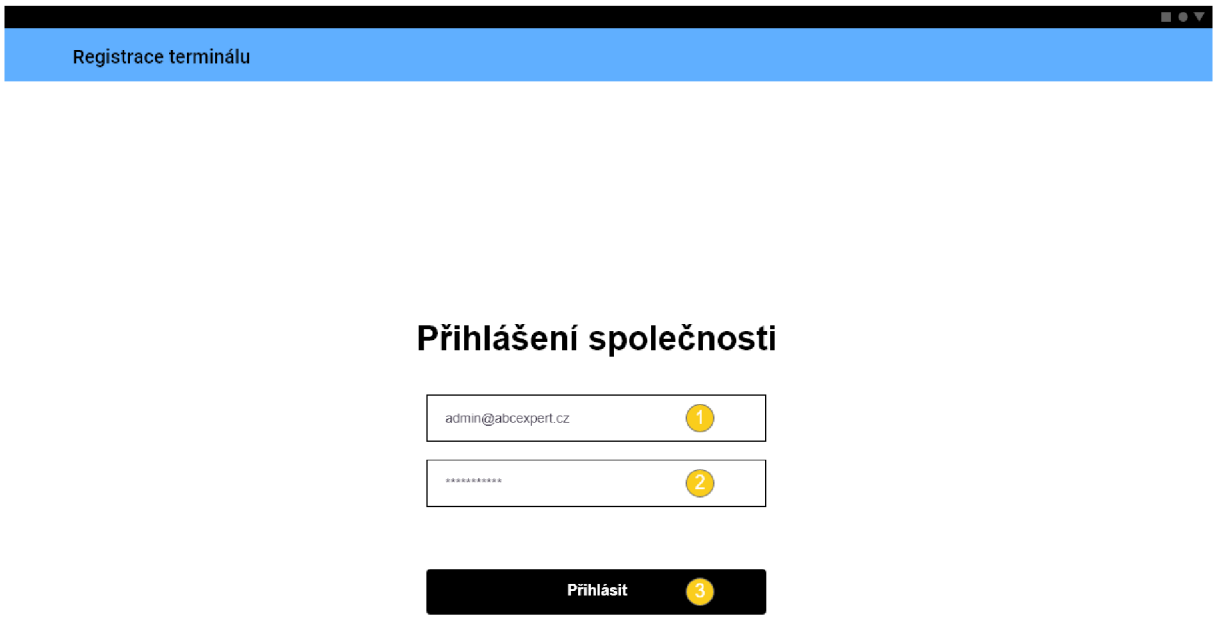
Celý návrh aplikace bude v češtině, jelikož je směřována na český trh a české uživatele.

3.1.1 Uživatelské rozhraní

V této podkapitole bude představen přehled obrazovek a funkcí, které aplikace obsahuje. Grafický návrh je důležitou součástí vývoje aplikace. V tomto případě trval 14 dní práce v programu Adobe XD.

3.1.1.1 Obrazovky přihlášení

Při každém prvotním spuštění aplikace na terminálu proběhne kontrola databáze, jestli se v ní nachází unikátní ID terminálu, které lze získat pouze komunikací s webovou službou. Pokud se v databázi pod tímto údajem nachází hodnota *null*, je spuštěn proces registrace terminálu skrze údaje firmy, která je zaregistrovaná v systému. Uživatel zadá přihlašovací údaje firmy, které budou zašifrovány a http požadavkem odeslány na server, který ověří jejich správnost. Po úspěšné validaci obdrží uživatel na telefon uvedený v databázi serveru náhodně generovaný licenční kód, který zadá do textového pole na následující obrazovce. Při neúspěšné validaci údajů nebude žádný validační kód odeslán a uživatel bude vyzván k novému zadání údajů.



Obrázek 18: Přihlašovací obrazovka společnosti
(Zdroj: Vlastní vypracování)

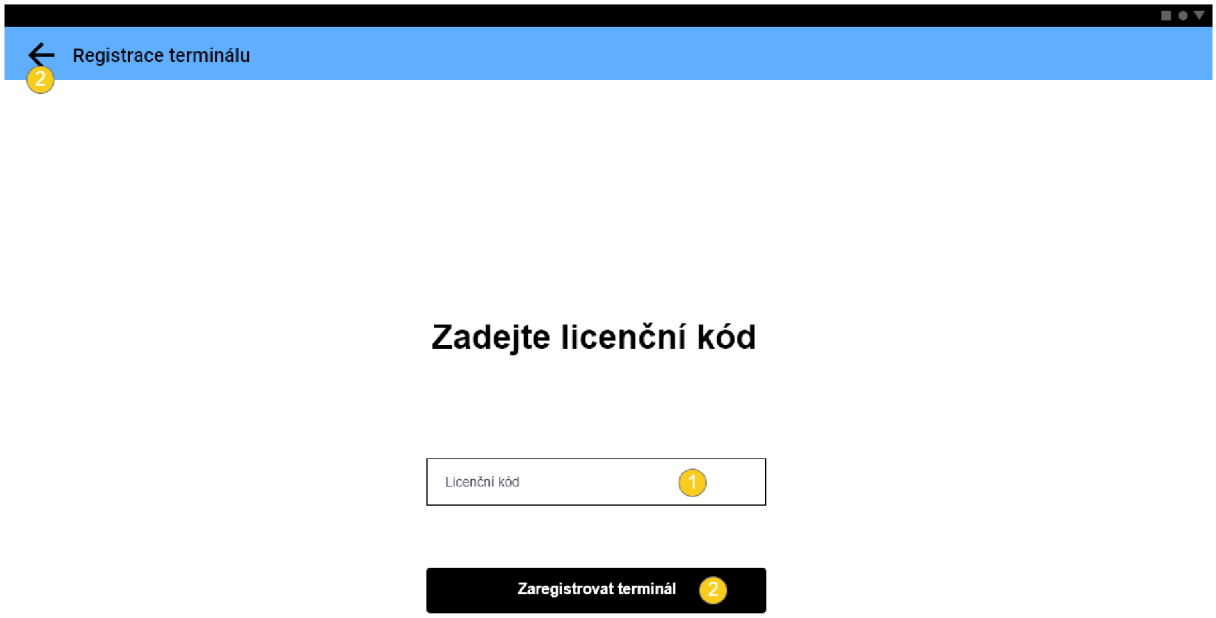
Ovládací prvky obrazovky:

1 – Textové pole s běžně zobrazeným textem pro zadání emailu společnosti

2 – Textové pole se skrytým textem pro zadání hesla

3 – Tlačítko přihlášení

Po přihlášení firmy přijde na telefonní číslo v databázi firmy náhodně generovaný licenční kód, který administrátor zadá do textového pole na této obrazovce. Poté potvrdí zadání tohoto kódu stiskem tlačítka na obrazovce. V případě, že validace neproběhne správně, vyzve aplikace k zadání kódu znovu. Pokud je kód zadán správně, jsou staženy veškeré potřebné informace o společnosti a terminálu z vnějšího zdroje, hlavně tedy identifikátor terminálu, který je posléze natrvalo uložen do databáze a aplikace při dalším spuštění již nebude vyžadovat registraci.



Obrázek 19: Obrazovka zadání licenčního kódu
(Zdroj: Vlastní vypracování)

Ovládací prvky obrazovky:

1 – Textové pole s běžně zobrazeným textem pro zadání licenčního kódu

2 – Tlačítko registrace

3 – Tlačítko zpět, kterým se uživatel dostane na předchozí obrazovku

Pokud proběhla registrace terminálu úspěšně, nebo již aplikace zná identifikátor zařízení a ke které firmě náleží, bude spuštěna obrazovka přihlášení zaměstnance, na které je jasně zobrazeno, k jaké firmě je terminál přihlášený. Na té se zaměstnanec přihlásí stejným způsobem, jako při přihlašování firmy. Pokud jsou zadané údaje správné, bude otevřena zaměstnanecká zóna aplikace.

Obrázek 20: Přihlašovací obrazovka zaměstnance
(Zdroj: Vlastní vypracování)

Ovládací prvky obrazovky:

1 – Textové pole s běžně zobrazeným textem pro zadání emailu zaměstnance

2 – Textové pole se skrytým textem pro zadání hesla

3 – Tlačítko pro přihlášení zaměstnance

3.1.1.2 Rozcestník

Obrazovka rozcestníku slouží jako hlavní navigační místo celé aplikace. Z této obrazovky může zaměstnanec spustit moduly a akce, ke kterým má oprávnění, které je možné v budoucnu rozšiřovat.

V tuto chvíli jde o spuštění pokladny, uzávěrku měsíce a procházení a případně storno již vystavených dokladů. Na této obrazovce je možné se i odhlásit ze zaměstnaneckého účtu, po kterém proběhne kontrola, zda není otevřená pokladna a uživatel na to bude případně upozorněn vyskakovacím oknem. Stejně tak, pokud se zaměstnanec pokusí provést uzávěrku měsíce, pokud bude stále otevřená pokladna.

Obrázek 21: Obrazovka rozcestníku
(Zdroj: Vlastní vypracování)

Ovládací prvky obrazovky:

1 – Tlačítko pro přechod do pokladny

2 – Tlačítko pro spuštění uzávěrky měsíce

3 – Tlačítko pro zobrazení vystavených dokladů

4 – Tlačítko pro odhlášení zaměstnance

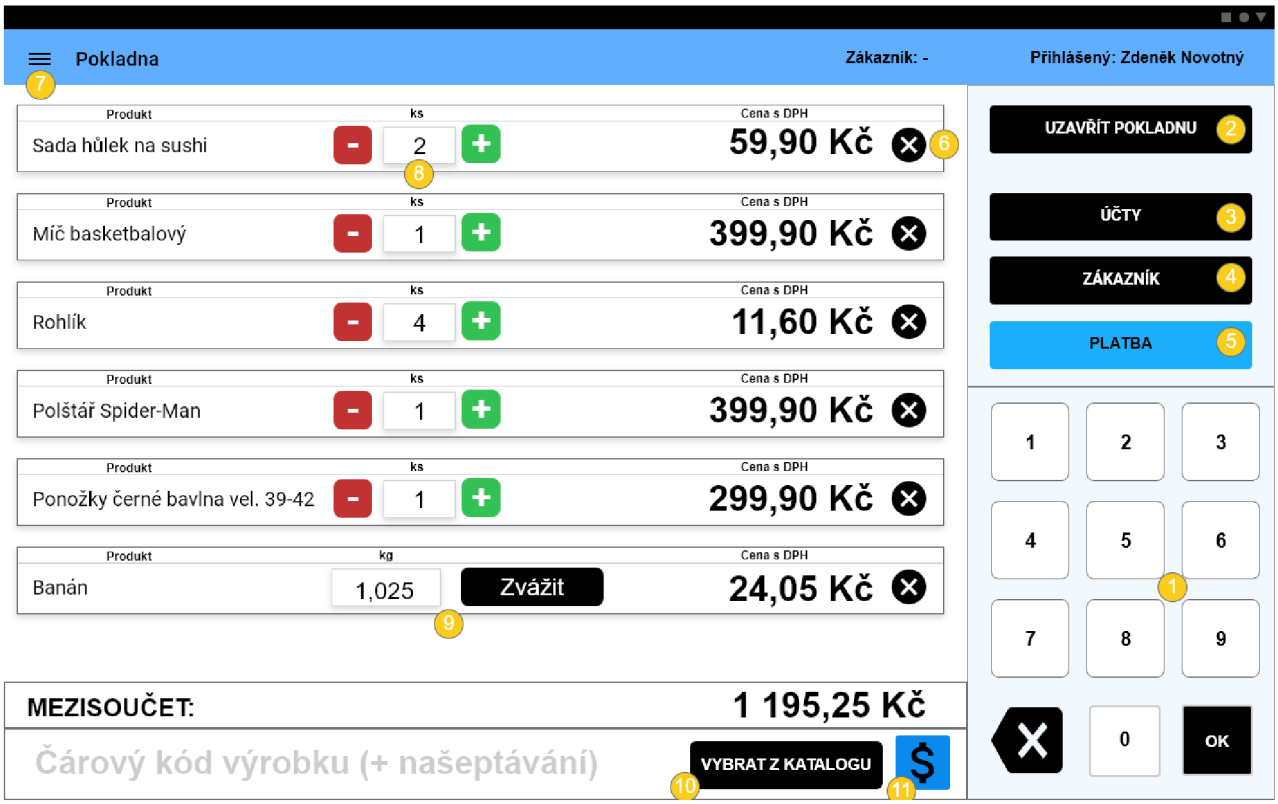
3.1.1.3 Pokladna

Hlavní a nejrozsáhlejší funkcí aplikace je pokladna, kde je obstaráván samotný prodej zboží.

Obrazovka pokladny

Tato obrazovka je rozdělena do tří hlavních oblastí – košík, kde jsou zobrazeny informace o načteném zboží, a který zabírá většinu obrazovky vlevo, horní panel, který poskytuje informace o aktuálně otevřeném účtu a možnost zobrazení rychlého navigačního menu a ovládací tlačítka pokladny, které slouží k manipulaci s celým účtem, případně pokladnou.

Během chodu této obrazovky bude možné zadat čárový kód zboží buďto pomocí USB čtečky čárových kódů připojené k zařízení, nebo pomocí číselníku po pravé straně (1). Při manuálním zadávání se bude kód vypisovat na spodní řádek a nabídne našeptávání výrobků se stejným čárovým kódem. Kromě toho je možné vybrat položku zboží z katalogu, který je uložen v databázi zařízení.
Pokladní má možnost přiřadit konkrétní otevřený účet zákazníkovi a přepínat mezi nimi. Na obrazovce se také nachází prvky pro manipulaci s počtem či hmotností konkrétního zboží.
V případě, že je manipulováno s hotovostí na pokladně bez uskutečnění prodeje či reklamace nějakého zboží, například při doplňování hotovosti na pokladnu, je použito tlačítko se znamením dolaru. Po jeho zmáčknutí zadá uživatel do dialogového okna částku, a jestli se jedná o výdaj, nebo příjem do pokladny.
Na konci dne pokladní uzavře pokladnu tlačítkem nahoře vpravo (2). Systém zkontroluje, jestli není otevřený nějaký účet a případně na to pokladní upozorní.



Obrázek 22: Obrazovka pokladny
(Zdroj: Vlastní vypracování)

Ovládací prvky obrazovky:

1 – Číselník

2 – Tlačítko pro uzavření pokladny

3 - Tlačítko pro výběr otevřených účtů

4 - Tlačítko pro výběr zákazníka

5 – Tlačítko pro spuštění platby

6 – Tlačítko pro odebrání položky z košíku

7 – Tlačítko pro otevření rychlého menu

8 – Číselné pole s počtem kusů zboží. Hodnota může být manipulována také tlačítky vedle

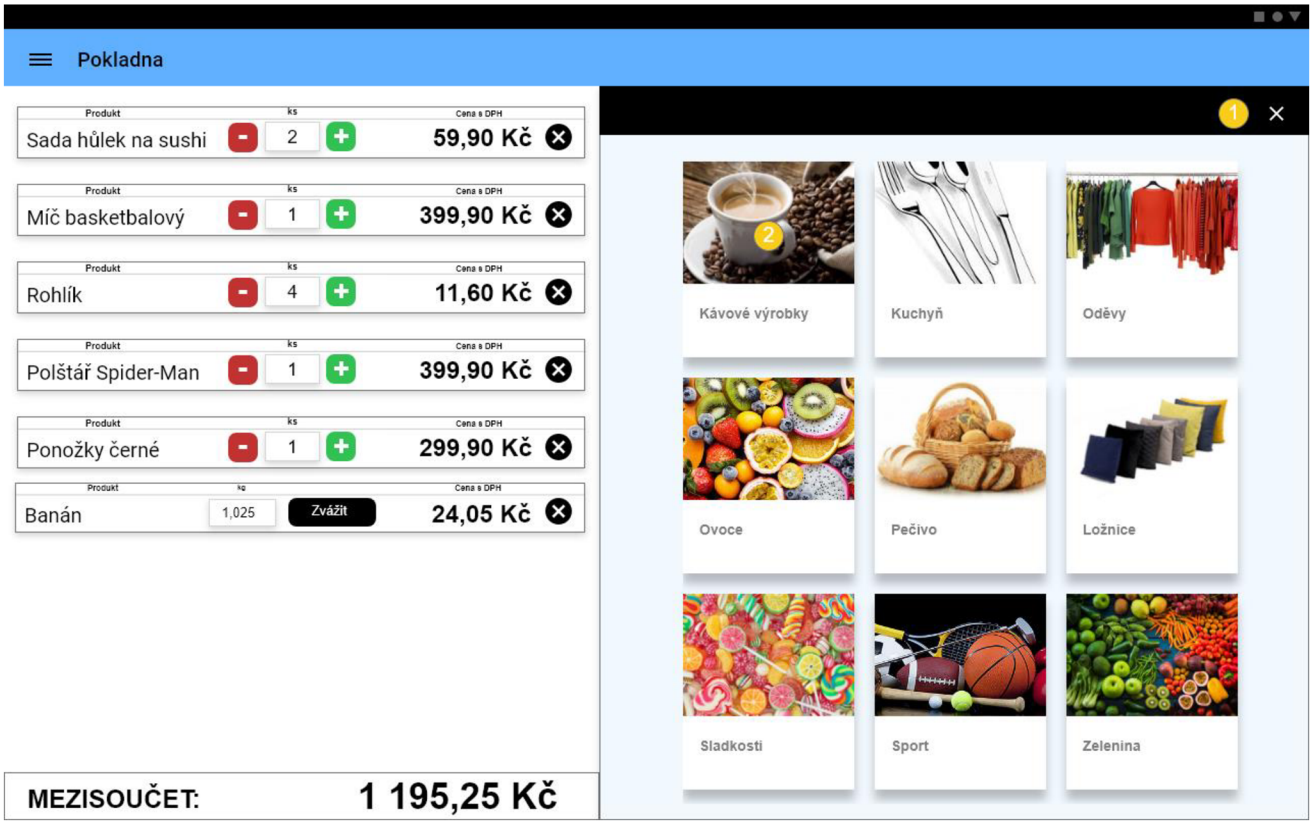
9 – Číselné pole s hmotností zboží. Hodnota může být aktualizována opětovným převážením pomocí tlačítka vpravo.

10 – Tlačítko pro otevření katalogu zboží.

11 – Tlačítko pro spuštění dialogu pro zadání výdajů nebo příjmů do pokladny.

Výběr z katalogu

Při výběru z katalogu větší část obrazovky zabere dlaždicové menu s kategoriemi výrobků, které jsou uloženy v databázi zařízení. I v tomto režimu zobrazení může pokladní manipulovat se zbožím v košíku, ale zmizí tlačítka pro navigaci a číselník. Toto zobrazení na obrazovce zůstane, dokud ho pokladní manuálně nezavře.



Obrázek 23: Obrazovka výběru z katalogu
(Zdroj: Vlastní vypracování)

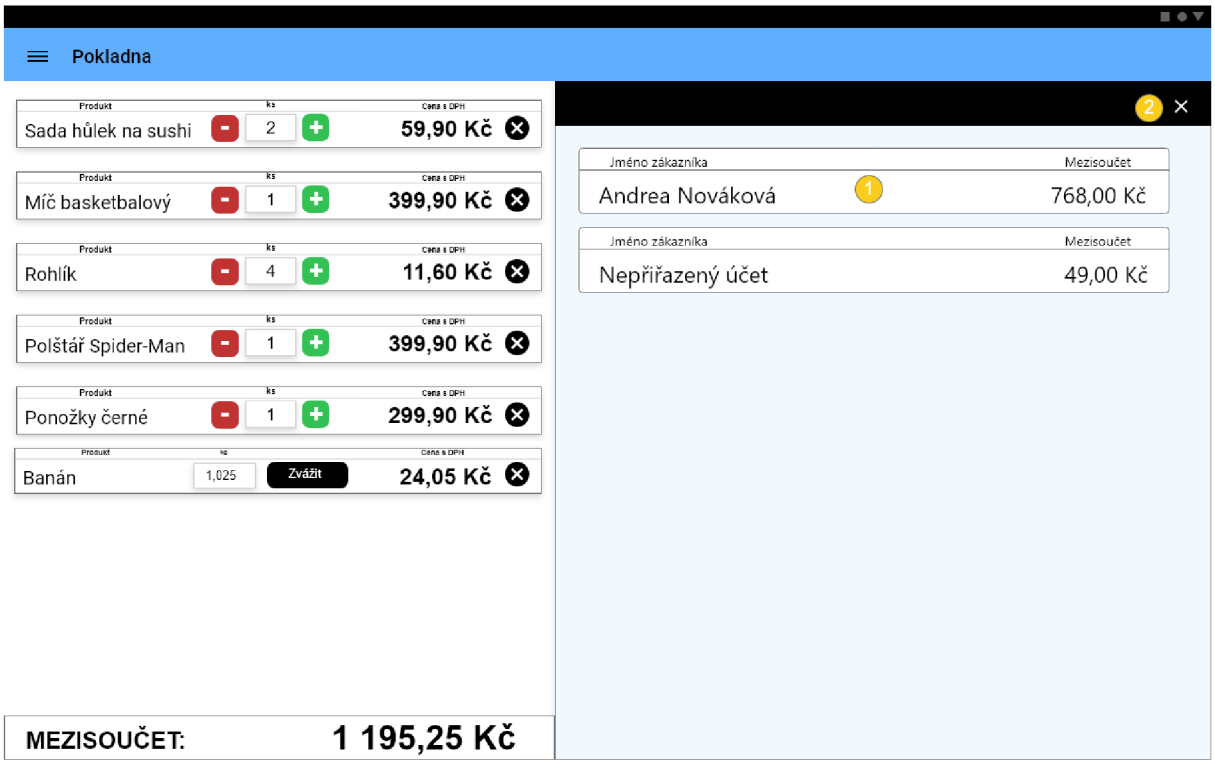
Ovládací prvky obrazovky:

1 – Tlačítko pro uzavření výběru kategorie

2 – Tlačítko pro výběr kategorie

Otevřené účty

Režim otevřených účtů je určen pro situace, když pokladní obsluhuje více zákazníků najednou. Výběrem ze seznamu se do košíku načtou položky z vybraného účtu, se kterými je možné dále manipulovat.



Obrázek 24: Obrazovka otevřených účtů
(Zdroj: Vlastní vypracování)

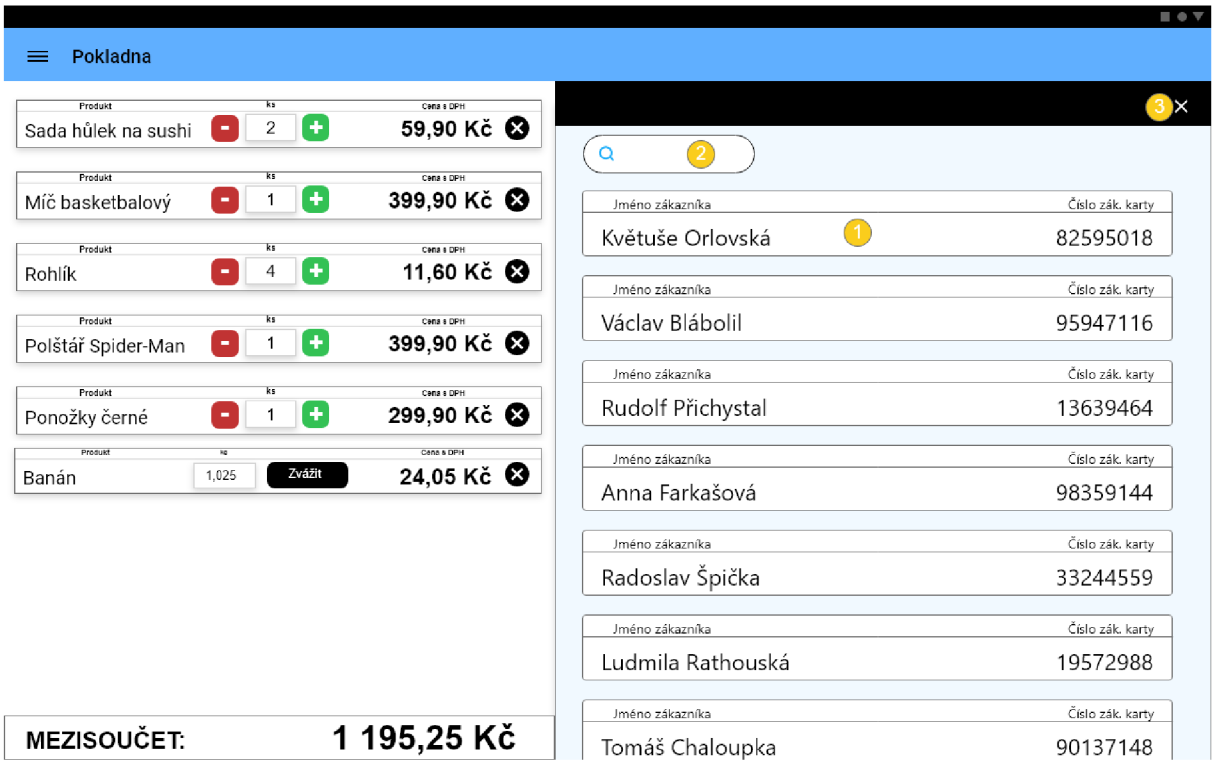
Ovládací prvky obrazovky:

1 – Pole, jehož výběrem bude otevřen účet, ke kterému patří

2 – Tlačítko pro uzavření výběru účtu

Výběr zákazníka

V případě, že zákazník vlastní zákaznickou kartu a není možné ji načíst běžným způsobem přes čtečku čárových kódů, je možné mu přiřadit účet z databáze zákazníků výběrem ze seznamu, kde je možné vyhledávat zákazníka podle jména.



Obrázek 25: Obrazovka výběru zákazníka
(Zdroj: Vlastní vypracování)

Ovládací prvky obrazovky:

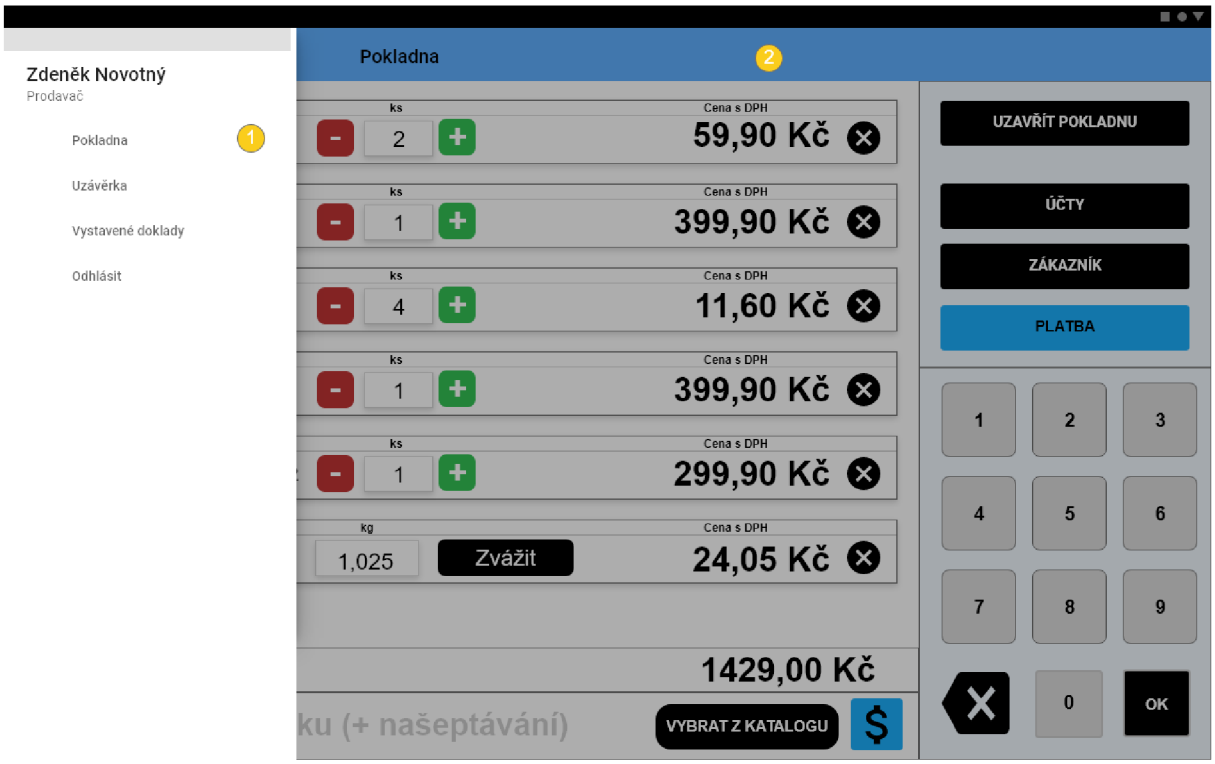
1 – Pole, jehož výběrem bude přiřazen zákazník k právě otevřenému účtu

2 – Textové pole pro vyhledávání zákazníka

3 – Tlačítko pro uzavření výběru zákazníka

Rychlé menu

K rychlé navigaci mezi režimy aplikace slouží rychlé menu, které je otevřeno pomocí ikony tří pruhů nad sebou v levém horním rohu obrazovky. Po otevření je možné kliknout na položku, jejíž okno se otevře a není nutné přecházet přes rozcestník.



Obrázek 26: Rychlé navigační menu
(Zdroj: Vlastní vypracování)

Ovládací prvky obrazovky:

1 – Tlačítka pro rychlou navigaci

2 – Oblast, jejímž výběrem se uzavře rychlé menu

Platba

Po zadání pokynu platby se uzamkne možnost manipulovat s položkami, na pravé straně obrazovky se objeví dvě velká tlačítka s ikonami pro platbu kartou a v hotovosti. Pod nimi se ukáží informace o nákupu – počet položek, zákazník, ke kterému je nákup přiřazený, sleva, která na nákup je, cena bez DPH a cena s DPH. Z této obrazovky se dá odejít pouze šipkou v pravém horním rohu.

Obrázek 27: Obrazovka platby
(Zdroj: Vlastní vypracování)

Ovládací prvky obrazovky

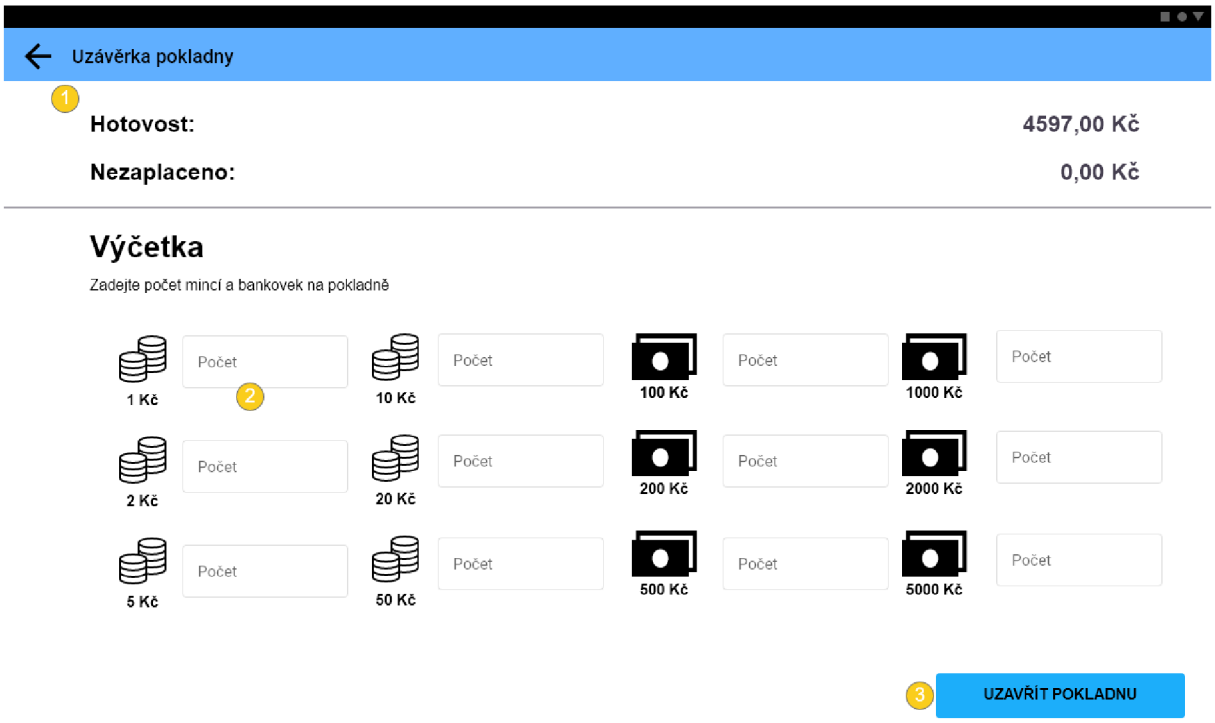
1 – Tlačítko pro platbu v hotovosti

2 – Tlačítko pro platbu kartou

3 – Tlačítko pro návrat

Uzávěrka pokladny

Na konci směny pokladní uzavře pokladnu. Do výčetky zadá počet mincí na pokladně, který ještě potom zkontroluje s hodnotou hotovosti, kterou si pokladna udržuje v mezipaměti. Tlačítkem uzavřít pokladnu uloží údaje do databáze a vrátí se zpět na rozcestník.



Obrázek 28: Uzávěrka pokladny
(Zdroj: Vlastní vypracování)

Ovládací prvky obrazovky:

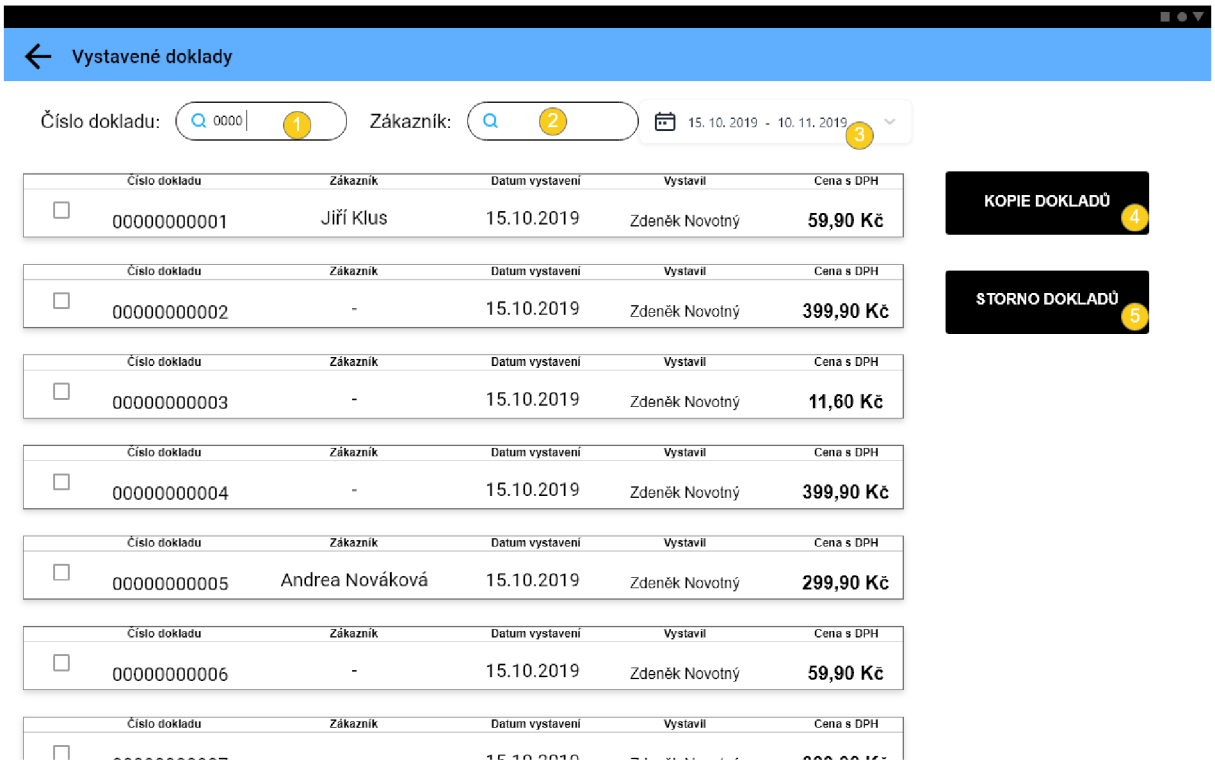
1 – Tlačítko pro návrat zpět

2 – Textové pole pro zadání počtu mincí a bankovek

3 – Tlačítko pro uzavření pokladny

3.1.1.4 Vystavené doklady

V obrazovce vystavených dokladů probíhá správa již uzavřených účtů. V horní části obrazovky jsou vyhledávací pole, pomocí kterých může zaměstnanec filtrovat doklady, které chce zobrazit. Buď podle čísla dokladu, zákazníka, nebo data vystavení. Poté může doklady označit poklepáním na ně a v pravé části obrazovky vybrat, jestli pro ně chce vystavit účtenku, nebo stornovat dané doklady.



Obrázek 29: Vystavené doklady
(Zdroj: Vlastní vypracování)

Ovládací prvky obrazovky:

1 – Textové pole pro zadání čísla dokladu

2 – Textové pole pro zadání jména zákazníka

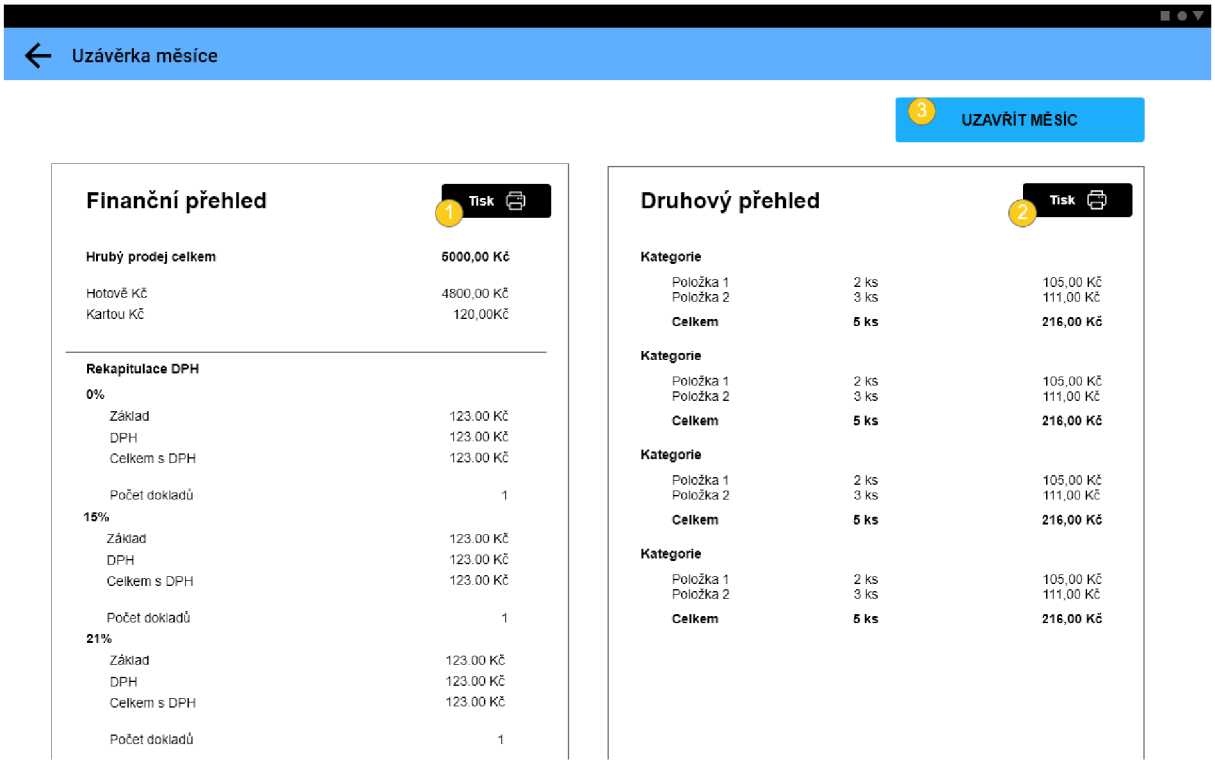
3 – Prvek pro výběr data vystavení

4 – Tlačítko pro vytisknutí kopie dokladů

5 – Tlačítko pro reklamaci účtů

3.1.1.5 Uzávěrka měsíce

Uzávěrka měsíce se na pokladně provádí každý poslední den v měsíci vedoucím pracovníkem prodejny. Slouží k potvrzení všech odeslaných dat pro účetní účely. Na levé straně obrazovky je finanční přehled, kolik peněz bylo přijato v hotovosti a rozdělení tržeb podle daňových sazeb prodávaného zboží. Na pravé straně obrazovky se nachází druhový přehled, kde je vidět zboží rozdělené podle kategorií, a za kolik se které prodalo. Každou z těchto výčetek je možné vytisknout na připojené tiskárně zvolením možnosti tisku.



Obrázek 30: Uzávěrka měsíce
(Zdroj: Vlastní vypracování)

Ovládací prvky obrazovky:

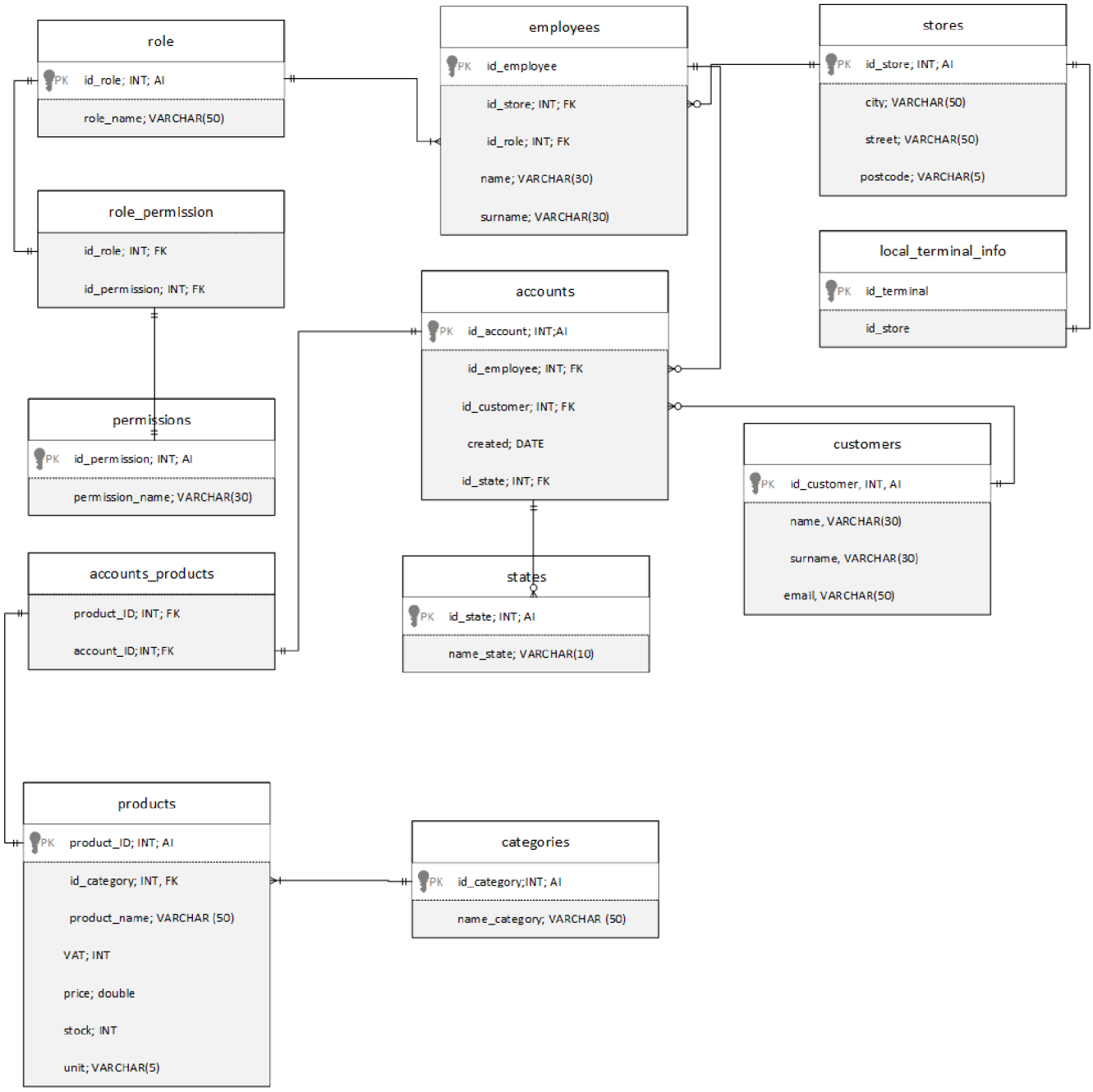
1 – Tlačítko pro tisk finančního přehledu měsíce

2 – Tlačítko pro tisk druhového přehledu měsíce

3 – Tlačítko pro potvrzení uzávěrky měsíce

3.1.2 Databáze aplikace

Aplikace bude mít vlastní SQL databázi, která bude zajištěna pomocí Androidové služby sqflite. Obsahuje tabulky (entity), které budou plněny pouze informacemi ze vzdáleného zdroje (employees, stores, local_terminal_info, products, categories, permissions, role_permission, role, states, customers) a tabulky, kde budou vznikat záznamy přímo při samotné činnosti aplikace, které budou odesílány dále na server (accounts, accounts_products). Komunikaci se serverem zajišťuje REST API, které zajišťuje všechny potřebné funkce, jako jsou *GET, POST, PUT,* nebo *DELETE.* Bližší pohled na databázi a vztahy mezi entitami poskytuje následující diagram, který vyl vytvořen v programu Microsoft Visio.



Obrázek 31: ER Diagram databáze aplikace
(Zdroj: Vlastní vypracování)

Entita role

Slouží k ukládání informací o pozicích ve společnosti. Popis atributů:

id_role – primární klíč, automaticky rostoucí číslo

role_name – název pozice

Entita permissions

Slouží k ukládání akcí, pro které je potřeba povolení

id_permission – primární klíč, automaticky rostoucí číslo

permission_name – název akce, ke které je potřeba povolení

Entita accounts

Tabulka, ve které se uchovávají informace o všech účtech, ať už otevřených, nebo uzavřených

id_account – primární klíč, automaticky rostoucí číslo

id_employee – cizí klíč, identifikátor zaměstnance, který účet vytvořil

id_customer – cizí klíč, identifikátor zákazníka, ke kterému účet patří

created – datum vytvoření záznamu

id_state – cizí klíč, identifikátor stavu, ve kterém se účet nachází

Entita employees

Slouží k uchovávání informací o zaměstnancích společnosti

id_employee – primární klíč, automaticky rostoucí číslo

id_store – cizí klíč, identifikátor pobočky, ke které zaměstnanec patří

id_role – cizí klíč, identifikátor pozice, na které zaměstnanec pracuje

name – jméno zaměstnance

surname – příjmení zaměstnance

Entita stores

Uchovává informace o pobočkách

id_store – primární klíč, automaticky rostoucí číslo

city – město

street – ulice

postcode – poštovní směrovací číslo

Entita local_terminal_info

V této entitě se bude nacházet pouze jeden záznam, slouží k uložení informací, kterými se bude terminál prokazovat při autorizaci.

id_terminal – primární klíč, ID terminálu, které obdrží systému

id_store – cizí klíč, vyjadřuje, k jaké pobočce terminál náleží

Entita customers

Uchovává informace o zákaznících

id_customer – primární klíč, číslo zákaznické karty

name – jméno

surname – příjmení

email – kontaktní email

Entita products

V této entitě jsou uloženy informace o produktech

id_product – primární klíč, automaticky rostoucí číslo

id_category – cizí klíč, identifikátor kategorie, ke které produkt náleží

product_name – název produktu

VAT – sazba DPH v procentech

price – cena na 1 ks nebo kg výrobku

stock – momentální počet jednotek výrobku na skladě

unit – název jednotky, ve které se produkt prodává

Entita categories

id_category – primární klíč, automaticky rostoucí číslo

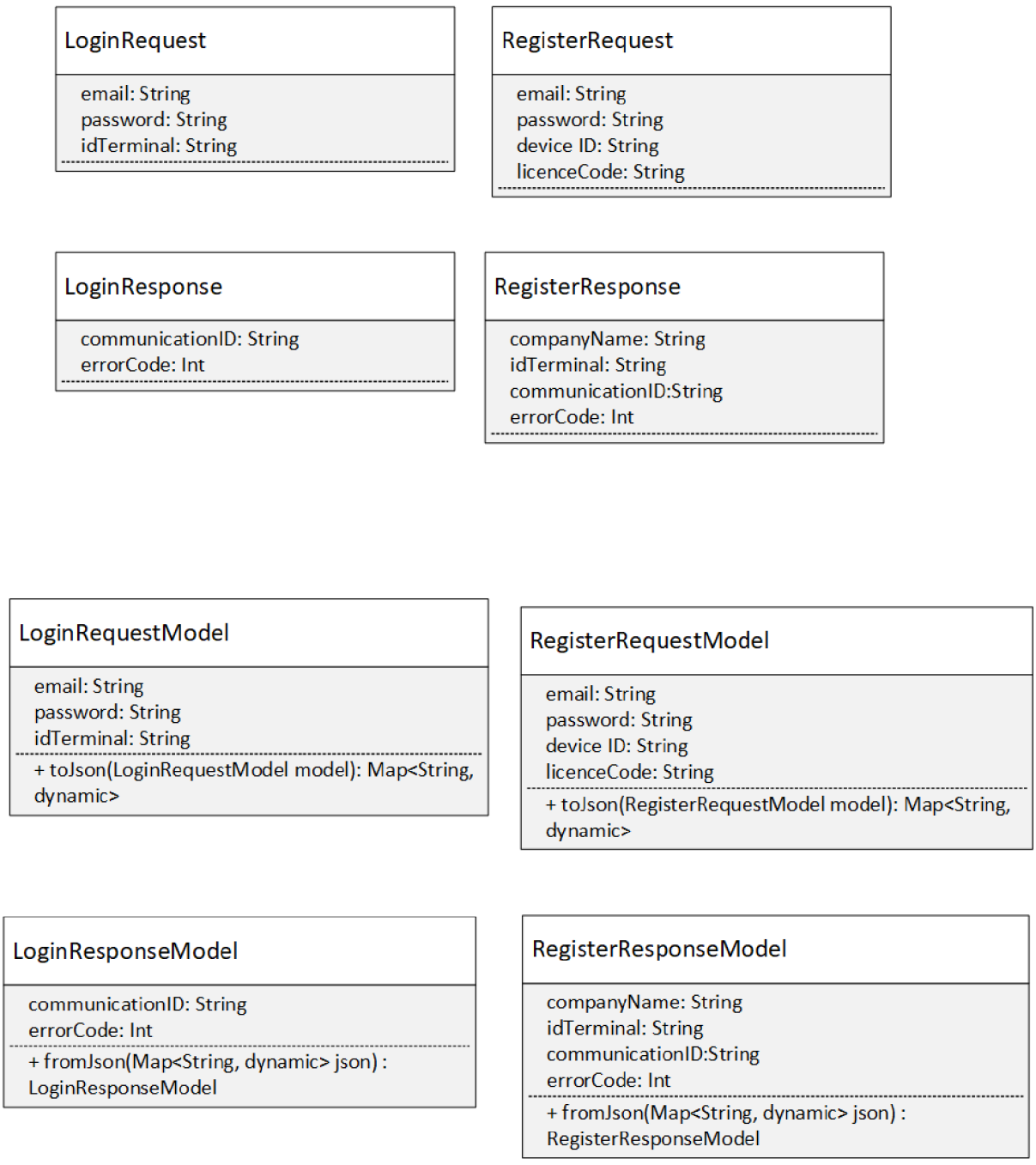
name_category – název kategorie

3.1.3 Rozvržení kódové základny vybrané činnosti

V této části bude na jednoduchém příkladu demonstrováno, jakým způsobem bude rozvržena kódová základna aplikace. Je postavena podle principů čisté architektury popsaných v teoretické části práce.

3.1.3.1 Entity

Entity jsou objekty, se kterými pracuje kód, a které stojí mimo SQL databázi, jejichž instance svou činností vytváří uživatel, a které nemusí být nikam dlouhodobě ukládány. Jedná se o třídy, které mají svůj konstruktor, atributy, a mohou, ale nemusí obsahovat metody. Vznikají a zanikají při činnosti uživatele v aplikaci a mají jasně dané použití. Pro každou entitu ještě existuje třída datového modelu, se kterou pracuje datová vrstva aplikace, a která dědí z entity pomocí klíčového slova *extends*.



Obrázek 32: Datové modely aplikace
(Zdroj: Vlastní vypracování)

Entita LoginRequest

Slouží ke shromáždění dat pro vytvoření http požadavku na server

Email – emailová adresa, pod kterou se uživatel přihlašuje

Password – heslo zadané uživatelem

idTerminal – při prvotním přihlášení neexistuje a je nutné ho získat ze serveru a uložit do databáze

Entita RegisterRequest

email – emailová adresa, pod kterou se uživatel přihlašuje

password – heslo zadané uživatelem

deviceID – jednoznačný identifikátor zařízení, který je získaný pomocí *platform channel* z operačního systému a které má každé mobilní zařízení na světě unikátní

licCode – licenční kód zadaný uživatelem

Entita LoginResponse

Slouží ke zpracování odpovědi ze serveru při činnosti přihlášení

communicationID – ID komunikace, kterým se terminál při pozdější komunikaci prokazuje serveru

errorCode – chybový kód komunikace

Entita RegisterResponse

Slouží ke zpracování odpovědi ze serveru při činnosti registrace

companyName – Jméno firmy, ke které terminál patří

idTerminal – ID terminálu přidělené serverem, které je uložené do paměti

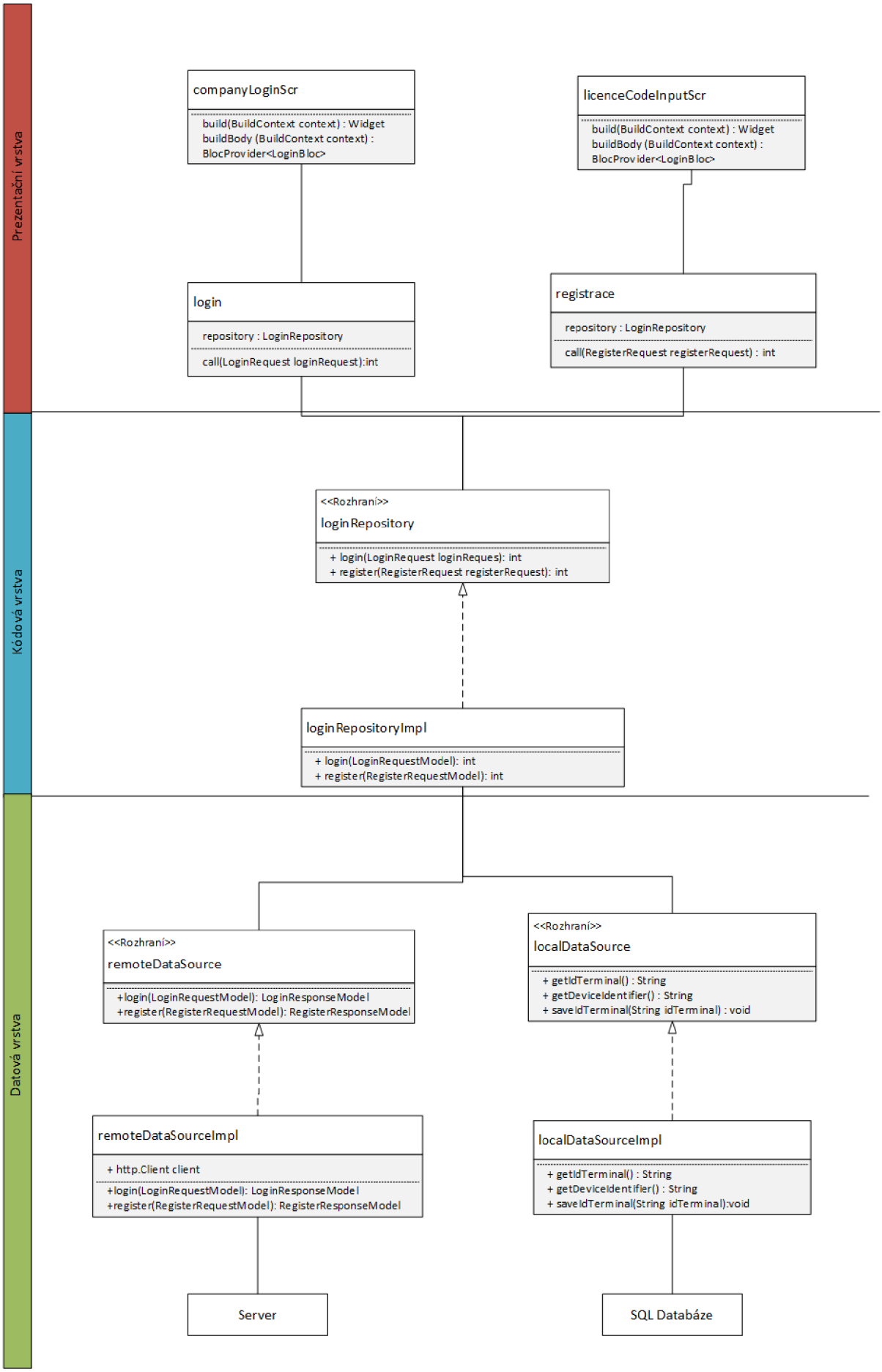
communicationID – ID komunikace, kterým se terminál při pozdější komunikaci prokazuje serveru

errorCode – chybový kód komunikace

Každá třída má ještě vlastní model, který je rozšiřuje původní třídu o metodu převodu na a z formátu Json, který je vložen do těla http požadavku nebo zpracován z odpovědi serveru.

3.1.3.2 Třídový model funkce přihlašování

Pro co největší přehlednost a flexibilitu kódu je kódová základna rozvržena do tří vrstev – prezentační, kódové a datové. V každé vrstvě existují třídy, které mají jasně vymezený účel a oprávnění. V prezentační vrstvě jsou třídy určené k zobrazování uživatelského rozhraní a příklady použití, což jsou činnosti, které může uživatel spustit svojí interakcí se zařízením. V kódové vrstvě je repozitář funkcí, který obsahuje logiku aplikace a propojuje prezentační vrstvu s datovou vrstvou. V datové vrstvě jsou třídy, které interagují se zdroji dat, například se serverem nebo s databází. Každá třída v datové a kódové vrstvě má i svoje rozhraní, skrze které v ní spouští kód třídy v diagramu nad ní, podle Interface Segregation Principle.


Prezentační vrstva
Kódová vrstva
Datová vrstva
companyLoginScr
build(BuildContext context) : Widget
buildBody (BuildContext context) :
BlocProvider<LoginBloc>
licenceCodeInputScr
build(BuildContext context) : Widget
buildBody (BuildContext context) :
BlocProvider<LoginBloc>
login
repository : LoginRepository
call(LoginRequest loginRequest):int
registrace
repository : LoginRepository
call(RegisterRequest registerRequest) : int
<<Rozhraní>>
loginRepository
+ login(LoginRequest loginReques): int
+ register(RegisterRequest registerRequest): int
loginRepositoryImpl
+ login(LoginRequestModel): int
+ register(RegisterRequestModel): int
<<Rozhraní>>
remoteDataSource
+login(LoginRequestModel): LoginResponseModel
+register(RegisterRequestModel): RegisterResponseModel
<<Rozhraní>>
localDataSource
+ getIdTerminal() : String
+ getDeviceIdentifier() : String
+ saveIdTerminal(String idTerminal) : void
remoteDataSourceImpl
+ http.Client client
+login(LoginRequestModel): LoginResponseModel
+register(RegisterRequestModel): RegisterResponseModel
localDataSourceImpl
+ getIdTerminal() : String
+ getDeviceIdentifier() : String
+ saveIdTerminal(String idTerminal):void
Server
SQL Databáze

Obrázek 33: Třídní model přihlašovací a registrační části
(Zdroj: Vlastní vypracování)

Třída companyLoginScr

Slouží k vykreslení uživatelského rozhraní přihlašovací obrazovky a k přijetí vstupu od uživatele.

build – metoda, kterou Flutter potřebuje, aby mohl vykreslit uživatelské rozhraní na obrazovku
buildBody – metoda, která vykresluje samotné tělo přihlašovací obrazovky. Přiřazuje akce k prvkům uživatelského rozhraní.

Třída licenceCodeInputScr

Slouží k vykreslení uživatelského rozhraní obrazovky zadávání licenčního kódu a přijetí vstupu od uživatele.

build – metoda, kterou Flutter potřebuje, aby mohl vykreslit uživatelské rozhraní na obrazovku
buildBody – metoda, která vykresluje samotné tělo obrazovky zadávání licenčního kódu

Třída loginRepositoryImpl

Třída, která slouží jako propojující prvek prezentační a datové vrstvy. Obsahuje logiku aplikace, využívá služby nižších vrstev a vyšším vrstvám navrací výsledky prováděných operací. Prezentační vrstva s touto třídou může interagovat přes rozhraní.

login– metoda, která slouží k přihlášení uživatele. Při svém běhu zavolá metodu getIdTerminal a login. Jejím návratovým typem je chybový kód.
register – metoda, která slouží k registraci terminálu. Při svém běhu zavolá metodu getDeviceIdentifier a register. Její návratovou hodnotou je chybový kód (integer).

Třída localDataSource

Obsahuje metody, které přímo interagují s lokálními datovými zdroji aplikace, v tomto případě SQL Databází.

getTerminalID – slouží k získání identifikátoru terminálu z databáze, který mu byl přiřazen webovou službou a je využíván k přihlášení. Navrací textový řetězec.
getDeviceIdentifier – slouží k získání identifikátoru zařízení, který je unikátní pro každé zařízení s Android nebo iOS na světě, a jenž je získáván pomocí *platform channel*. Navrací

textový řetězec.

saveIdTerminal – metoda, která uloží vstupní parametr do databáze jako ID Terminálu.

Třída remoteDataSource

Obsahuje metody, které slouží k interakci se vzdáleným zdrojem dat pomocí instance třídy http client, která zajišťuje vytváření a posílání http požadavků.

login – vstupní parametr převede na Json a vloží ho do těla http požadavku, který odešle na adresu uloženou ve funkci. Tělo odpovědi opět převede z formátu Json na datový model, se kterým pracuje aplikace.

register - vstupní parametr převede na Json a vloží ho do těla http požadavku, který odešle na adresu uloženou ve funkci. Tělo odpovědi opět převede z formátu Json na datový model, se kterým pracuje aplikace.

3.2 Ekonomické zhodnocení

Mzda při realizaci byla stanovena na 300 korun za hodinu. V jednom pracovním dni je 8 hodin. Vzhledem k tomu, že aplikace je vyvíjena pomocí multiplatformního frameworku, je potřeba dovyvinout kód, který je unikátní pro každou platformu. Jako první je potřeba vytvořit grafický návrh a prototyp aplikace, se kterou je klient spokojený. V tomto případě vývoj trval 14 pracovních dní. Po odsouhlasení grafického návrhu následuje vývoj multiplatformního kódu, který podle plánu zabere 60 pracovních dní. Dále následuje vývoj kódu specifického pro každou platformu zvlášť, který zabere 15 pracovních dní. Vývoj probíhá zároveň s testováním funkcí, a hotová aplikace je pak nasazena do testovacího provozu, který bude trvat 20 pracovních dní, a bude za něj odměna 200 korun na hodinu. Vzhledem k tomu, že aby aplikace fungovala správně, je zapotřebí i funkčních webových služeb na serveru, jejichž vývoj bude mít vlastní rozpočet, nemá smysl uvažovat výnosy plynoucí z prodeje služby do rozpočtu aplikace.

Položka	Dní	Hodin	Kč/h	Celkem kč
Prototyp	14	112	300	33 600
Vývoj společného kódu	60	480	300	144 000
Vývoj kódu pro Android	15	120	300	36 000

Vývoj kódu pro iOS	15	120	300	36 000
Testovací provoz	20	160	200	32 000
Celkem	**-**	**-**	**-**	**281 600**

Tabulka 3: Ekonomické zhodnocení nákladů vývoje aplikace
(Zdroj: Vlastní vypracování)

3.3 Přínosy práce

Aplikace vytvořená podle této bakalářské práce bude mít přínos pro firmu B2C Support jako důležitá součást jejích služeb. Dále bude mít přínos pro jejich zákazníky, kteří v rámci nové služby dostanou do rukou nástroj, který je posune o krok blíže k implementaci robustního informačního systému do své firmy, ve kterém bude tato aplikace sloužit jako prostředek k automatickému sběru dat v oblasti prodeje.

Přínosy má toto řešení i vůči samotným zaměstnancům (potažmo prodavačům). Po implementaci tohoto řešení budou snáze a rychleji vyřizovat běžné úkony na pokladně, a noví zaměstnanci se budou snáze zaučovat do používání pokladního systému.

V rámci inovace produktu bude tato aplikace obohacována a aktualizována podle potřeb zákazníků, mohou tedy přibýt i kompletně nové části jako je například management skladových zásob nebo správa zaměstnaneckých profilů, a to, díky uspořádání kódové základny do jasně oddělených a člověkem rozeznatelných částí, relativně snadno. Díky volbě vývojových nástrojů je možné tuto aplikaci nabízet firmám či jednotlivcům kteří disponují zařízením s Android i iOS.

ZÁVĚR

Cílem této bakalářské práce bylo vytvořit návrh vývoje aplikace pro firmu ABC Expert, kterou by mohla firma dále nabízet svým zákazníkům jako součást svých služeb a která by usnadnila zákazníkům evidenci tržeb, zefektivnila práci na pokladnách a zajistila uživatelské rozhraní pro sběr dat, která budou využita v rámci poskytování business intelligence služeb. Vzhledem k tomu, že aplikace využívá multiplatformního frameworku, je možné ji instalovat i na zařízení s operačním systémem iOS, pokud to bude odběratel vyžadovat.

Před vytvořením aplikace bylo nutné položit teoretické základy pro samotný vývoj aplikací na mobilní platformy. V teoretické části byl vysvětlený princip technologie Flutter a programovacího jazyku Dart, ve kterém je psaná většina kódu aplikace. Následně byly popsány různé možnosti vývojových prostředí, která tento způsob vývoje umožňují, tedy Android Studio, IntelliJ IDEA. Poté byla vysvětlena analýza SWOT, která se používá k identifikaci silných a slabých stránek analyzovaného subjektu, a také příležitostí a hrozeb, které mohou ovlivnit jeho další vývoj. Byl představen jazyk SQL, který je používaný pro tvorbu a správu databází v drtivé většině systémů dnešní doby. Byly popsány principy čisté architektury, které jsou využívány pro vývoj softwaru, aby pokračoval stabilní rychlostí a zamezilo se zdržením vývoje plynoucím z nedodržení těchto principů. Dále byl představen jazyk UML, který je v části návrhu aplikace použit jako médium pro navrhnutí softwarové základny, a byla představena jeho další využití. Pro datové modelování bylo dále nutné představit entitně-relační diagram, který sloužil v třetí kapitole k návrhu lokální databáze aplikace. V neposlední řadě se tato kapitola věnovala popsání uživatelského prostředí a návrhovému programu Adobe XD. Na závěr byly představeny operační systémy, které se na mobilních platformách nachází, a podíl jejich zastoupení.

Analytická část se v úvodu věnovala představením firmy, službám, které nabízí a organizaci toku informací ve firmě a jejím silným a slabým stránkám při vstupu na tento trh. Vzhledem k tomu, že novým produktem, který bude firma nabízet, bude pokladní aplikace, analyzoval jsem aplikace, které se již na trhu nachází. Závěr této kapitoly byl věnovaný analýze podílu zastoupení operačních systémů na zařízení typu tablet, aby bylo jasné, jaké bude pravděpodobné koncové zařízení aplikace. Z analýzy bylo patrné, že 99 % tabletů v Evropě má jako operační systém buď Android, nebo iOS, a že tyto dva systémy jsou využívány v podobné míře.

Poslední kapitola už se věnovala samotnému návrhu aplikace. V první fázi bylo nutné vytvořit prototyp aplikace kvůli rozvržení jejích funkcí a operativní logice, kterou se bude aplikace řídit v provozu. Po vytvoření tohoto návrhu byla navržena databáze aplikace pomocí ERD diagramu, který sloužil k představení entit, které v aplikaci budou zastoupené. Taktéž napomohl k zobrazení vazeb a jejich kardinality. Následoval výčet entit, a popis jednotlivých atributů. Poté bylo za pomocí třídního modelu na příkladu přihlašovací funkce demonstrováno rozložení kódové základny. Byly představeny entity, se kterými pracuje aplikace za chodu a nikam se neukládají, a poté byly třídním modelem znázorněny jednotlivé třídy a funkce, které obsahují, jejichž účel byl poté dále rozveden. Na závěr této kapitoly bylo ukázáno ekonomické zhodnocení celého vývoje a přínosům, které tato aplikace přináší klientovi i uživatelům.

Kromě přínosů klientovi a uživatelům měla tato práce velký osobní přínos. Díky ní jsem nabyl mnoha vědomostí z oblasti vývoje softwaru pro mobilní aplikace, a dozvěděl se o osvědčených postupech používaných ve vývoji, které zlepšují mé programátorské schopnosti. Dozvěděl jsem se o nejnovějších technologiích z oblasti vývoje mobilních aplikací. Dostal jsem možnost práce s grafickým programem, který se v průmyslu využívá pro prototypování aplikací. Taktéž jsem si vyzkoušel navržení databáze a její zevrubný popis, a navržení kódové základny, která by při vývoji mohla sloužit jako návod pro ostatní programátory, kteří by se na projektu podíleli. V oblasti analýzy jsem si vyzkoušel použít dovednosti nabyté při studiu ve vztahu k analýze firemního prostředí.

SEZNAM POUŽITÝCH ZDROJŮ

1) Flutter SDK releases. *Flutter* [online]. 2021 [cit. 2021-5-10]. Dostupné z: https://flutter.dev/docs/development/tools/sdk/releases
2) Kotlin 1.0 Released: Pragmatic Language for JVM and Android. *The Kotlin Blog* [online]. 2021 [cit. 2021-5-10]. Dostupné z: https://blog.jetbrains.com/kotlin/2016/02/kotlin-1-0-released-pragmatic-language-for-jvm-and-android/
3) GitHub - Flutter. *GitHub* [online]. 2021 [cit. 2021-5-10]. Dostupné z: https://github.com/flutter/flutter
4) GitHub – Kotlin. *GitHub* [online]. 2021 [cit. 2021-5-10]. Dostupné z: https://github.com/JetBrains/kotlin
5) Flutter widgets intro. *Flutter* [online]. 2021 [cit. 2021-5-10]. Dostupné z: https://flutter.dev/docs/development/ui/widgets-intro
6) Flutter architectural overview. *Flutter* [online]. 2021 [cit. 2021-5-10]. Dostupné z: https://flutter.dev/docs/resources/architectural-overview
7) Hot Reload. *Flutter* [online]. 2021 [cit. 2021-5-10]. Dostupné z: https://flutter.dev/docs/development/tools/hot-reload
8) Download Android Studio and SDK tools | Android Developers. *The official site for Android app developers. Provides the Android SDK tools and API documentation.* [online]. 2021 [cit. 2021-5-10]. Dostupné z: https://developer.android.com/studio/index.html
9) Google releases Android Studio 1.0, the first stable version of its IDE. *VentureBeat* [online]. 2021 [cit. 2021-5-10]. Dostupné z: https://venturebeat.com/2014/12/08/google-releases-android-studio-1-0-the-first-stable-version-of-its-ide/
10) What was the very first IntelliJ IDEA version? *IDEs Support (IntelliJ Platform)* | *JetBrains* [online]. 2021 [cit. 2021-5-10]. Dostupné z: https://intellij-support.jetbrains.com/hc/en-us/community/posts/360008145040-What-was-the-very-first-IntelliJ-IDEA-version
11) JetBrains Supports the Apache Software Foundation. *JetBrains News* [online]. 2021 [cit. 2021-5-10]. Dostupné z: https://blog.jetbrains.com/blog/2019/05/30/jetbrains-supports-the-apache-software-foundation/

12) Free Educational Licenses. *JetBrains: Essential tools for software developers and teams* [online]. 2021 [cit. 2021-5-10]. Dostupné z: https://www.jetbrains.com/community/education/#students
13) IntelliJ IDEA: The Capable & Ergonomic Java IDE by JetBrains. *JetBrains: Essential tools for software developers and teams* [online]. 2021 [cit. 2021-5-10]. Dostupné z: https://www.jetbrains.com/idea/
14) A tour of the Dart language. *Dart programming language | Dart* [online]. 2021 [cit. 2021-5-10]. Dostupné z: https://dart.dev/guides/language/language-tour
15) Dart: a language for structured web programming. *The official Google Code blog* [online]. 2021 [cit. 2021-5-10]. Dostupné z: http://googlecode.blogspot.com/2011/10/dart-language-for-structured-web.html
16) Dart programming language | Dart. *Dart programming language | Dart* [online]. [cit. 2021-5-10]. Dostupné z: https://dart.dev/
17) Language samples. *Dart programming language | Dart* [online]. [cit. 2021-5-10]. Dostupné z: https://dart.dev/samples

18) SWOT Analysis: How to Develop a Strategy For Success. *Management Training and Leadership Training - online* [online]. [cit. 2021-5-10]. Dostupné z: https://www.mindtools.com/pages/article/newTMC_05.htm
19) ŘEPA, Václav. VÝVOJOVÉ TRENDY METODIK VÝVOJE INFORMAČNÍCH SYSTÉMŮ – VÝZVA BPR. Dostupné z: http://nb.vse.cz/~repa/veda/EurOpen99%20Paper.pdf
20) UML - Overview. *Tutorialspoint* [online]. [cit. 2021-5-10]. Dostupné z: https://www.tutorialspoint.com/uml/uml_overview.htm
21) What is Unified Modeling Language (UML)? *Visual Paradigm* [online]. [cit. 2021-5-10]. Dostupné z: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/
22) UML - Building Blocks. *Tutorialspoint* [online]. [cit. 2021-5-10]. Dostupné z: https://www.tutorialspoint.com/uml/uml_building_blocks.htm
23) What is Entity Relationship Diagram (ERD)? *Visual Paradigm* [online]. [cit. 2021-5-10]. Dostupné z: https://www.visual-paradigm.com/guide/data-modeling/what-is-entity-relationship-diagram/
24) *SQL: Practical Guide for Developers* [online]. San Francisco: Morgan Kaufmann Publishers, 2005 [cit. 2021-5-10]. ISBN 978-0-1222-0531-6. Dostupné z:

https://books.google.cz/books?hl=en&lr=&id=19mS1g45fUEC&oi=fnd&pg=PP1&dq=sql+introduction&ots=xcPAPBZXBA&sig=aJCrklnVA4bUuohwmmaRo8uUb-8&redir_esc=y#v=onepage&q=sql%20introduction&f=false

25) SQL Data Types. *Wikimedia Commons* [online]. San Francisco, 2017 [cit. 2021-5-10]. Dostupné z: https://commons.wikimedia.org/wiki/File:SQL_data_types.png

26) *Clean Architecture: A Craftsman's Guide to Software Structure and Design* [online]. London: Pearson Education, 2018 [cit. 2021-5-10]. ISBN 978-0-13-449416-6. Dostupné z: https://raw.githubusercontent.com/sdcuike/Clean-Code-Collection-Books/master/Clean%20Architecture%20A%20Craftsman's%20Guide%20to%20Software%20Structure%20and%20Design.pdf

27) What is User Interface Design? *UX Design Courses & Global UX Community | Interaction Design Foundation (IxDF)* [online]. [cit. 2021-5-10]. Dostupné z: https://www.interaction-design.org/literature/topics/ui-design

28) Adobe XD | Fast & Powerful UI/UX Design & Collaboration Tool. *Adobe: Řešení pro kreativitu, marketing a správu dokumentů* [online]. [cit. 2021-5-10]. Dostupné z: https://www.adobe.com/cz/products/xd.html

29) Mobile & Tablet Operating System Market Share Worldwide | StatCounter Global Stats. *StatCounter Global Stats - Browser, OS, Search Engine including Mobile Usage Share* [online]. [cit. 2021-5-10]. Dostupné z: https://gs.statcounter.com/os-market-share/mobile-tablet/worldwide/#monthly-202103-202103-bar

30) Platform Architecture. *Android Developers* [online]. [cit. 2021-5-10]. Dostupné z: https://developer.android.com/guide/platform

31) An Overview of Android Operating System and Its Security Features. *Journal of Engineering Research and Applications* [online]. 2014, **4**(2), 3 [cit. 2021-5-10]. Dostupné z: https://www.gadgetgyani.com/wp-content/uploads/2016/03/android-features-pdf.pdf

32) IOS Architecture | iOS Tutorial | Intellipaat.com. *Online Professional Training Courses and Certification - Intellipaat* [online]. [cit. 2021-5-10]. Dostupné z: https://intellipaat.com/blog/tutorial/ios-tutorial/ios-architecture/

33) Core OS Layer. *Documentation Archive* [online]. [cit. 2021-5-10]. Dostupné z: https://developer.apple.com/library/archive/documentation/MacOSX/Conceptual/OSX_Technology_Overview/CoreOSLayer/CoreOSLayer.html

34) Core Services Layer. *Documentation Archive* [online]. [cit. 2021-5-10]. Dostupné z: https://developer.apple.com/library/archive/documentation/MacOSX/Conceptual/OSX_Technology_Overview/CoreOSLayer/CoreOSLayer.html
35) Media Layer. *Documentation Archive* [online]. [cit. 2021-5-10]. Dostupné z: https://developer.apple.com/library/archive/documentation/MacOSX/Conceptual/OSX_Technology_Overview/CoreOSLayer/CoreOSLayer.html
36) Cocoa Application Layer. *Documentation Archive* [online]. [cit. 2021-5-10]. Dostupné z: https://developer.apple.com/library/archive/documentation/MacOSX/Conceptual/OSX_Technology_Overview/CoreOSLayer/CoreOSLayer.html
37) B2C. *B2C* [online]. [cit. 2021-5-10]. Dostupné z: https://www.b2csupport.cz/
38) Sybila. *B2C* [online]. [cit. 2021-5-12]. Dostupné z: https://www.b2csupport.cz/index.php/informacni-systemy/16-sybila
39) SiD. *B2C* [online]. [cit. 2021-5-12]. Dostupné z: https://www.b2csupport.cz/index.php/informacni-systemy/15-sid
40) Dotykačka - pokladní systém pro restaurace, obchody a služby. *Dotykačka - pokladní systém pro restaurace, obchody a služby* [online]. [cit. 2021-5-10]. Dostupné z: https://www.dotykacka.cz/
41) Funkce aplikace – Profi Účtenka pro EET | Solitea Česká republika. *Profi Účtenka – Jedna EET Aplikace, mnoho možností* [online]. [cit. 2021-5-10]. Dostupné z: https://profiuctenka.cz/funkce-aplikace/
42) EET – Google Play. *Google Play* [online]. [cit. 2021-5-10]. Dostupné z: https://play.google.com/store/search?q=EET&hl=cs

SEZNAM POUŽITÝCH OBRÁZKŮ

SEZNAM POUŽITÝCH TABULEK

SEZNAM POUŽITÝCH GRAFŮ