UNIVERZITA PARDUBICE

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

DIPLOMOVÁ PRÁCE

2023 Bc. Karel Andres

Univerzita Pardubice

Fakulta elektrotechniky a informatiky

Integrace pokladní webové aplikace s dalšími vstupně-výstupními zařízeními

Diplomová práce

2023 Bc. Karel Andres

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2022/2023

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Karel Andres**
Osobní číslo:	**I21271**
Studijní program:	**N0613A140007 Informační technologie**
Téma práce:	**Integrace pokladní webové aplikace s dalšími vstupně-výstupními zařízeními**
Zadávající katedra:	**Katedra softwarových technologií**

Zásady pro vypracování

Cílem diplomové práce je integrace propojení webové aplikace s dalšími vstupně-výstupními zařízení (např. tiskárny, pokladní zásuvky, platební terminály apod.) prostřednictvím REST API. Tímto způsobem bude zabezpečen přístup k dostupným službám a bude umožněna komunikace s potřebnými zařízeními.

V teoretické části práce bude provedena analytická část zaměřená na zpracování požadavků, případů užití, analytických tříd atd.

Pro realizaci aplikace bude použita technologie Java Spring Boot a důležité části budou testovány pomocí jednotkových a integračních testů.

Rozsah pracovní zprávy: **40-50 normostran**
Rozsah grafických prací:
Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam doporučené literatury:

TURNQUIST L. G.Learning Spring Boot 2.0 – Second Edition: Simplify the development of lightning fast applications based on microservices and reactive programming. Packt Publishing, 2017, 370 pp. ISBN978-1786463784.

ARLOW J., NEUSTADT I. UML 2 a unifikovaný proces vývoje aplikací: objektově orientovaná analýza a návrh prakticky. 2., vyd. Brno: Computer Press, 2007. ISBN 978-80-251-1503-9.

Vedoucí diplomové práce: **doc. Ing. Michael Bažant, Ph.D.**
Katedra softwarových technologií

Datum zadání diplomové práce: **8. listopadu 2022**
Termín odevzdání diplomové práce: **19. května 2023**

L.S.

Ing. Zdeněk Němec, Ph.D. v.r.
děkan

prof. Ing. Antonín Kavička, Ph.D. v.r.
vedoucí katedry

V Pardubicích dne 30. listopadu 2022

Prohlašuji:

Práci s názvem “Integrace pokladní webové aplikace s dalšími vstupně-výstupními zařízeními” jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 7/2019 Pravidla pro odevzdávání, zveřejňování a formální úpravu závěrečných prací, ve znění pozdějších dodatků, bude práce zveřejněna prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 25. 5. 2023

Karel Andres v.r.

PODĚKOVÁNÍ

Mé poděkování patří doc. Ing. Michaelu Bažantovi, PhD., vedoucímu práce, za poskytnuté konzultace a pomoc při psaní této diplomové práce. Děkuji kolegům z firmy ENIGOO, kteří mě přivedli na myšlenku této diplomové práce. Dále bych chtěl moc poděkovat své manželce a rodině za nejen psychickou podporu při tvorbě této práce, ale i po celou dobu mých studií.

ANOTACE

Cílem diplomové práce je integrace propojení webové aplikace se vstupně-výstupními zařízeními (tiskárny, platební terminály, pokladní zásuvky apod.) prostřednictvím REST API.

V teoretické části bude provedena analýza zaměřená na zpracování požadavků, případů užití či analytických tříd.

Pro realizaci aplikace se předpokládá využití technologie Java Spring Boot a předpokládá se otestování důležitých částí pomocí jednotkových a integračních testů.

KLÍČOVÁ SLOVA

Webová aplikace, Vstupně-výstupní zařízení, Java Spring Boot, analýza, požadavky, případy užití

TITLE

Integration of the cash register web application with other input-output devices

ANNOTATION

Aim of the thesis is integration connection between web application and input-output devices (printers, payment terminal, cash drawer etc.) via REST API.

In the theoretical part, an analysis focused on the processing of requirements, use cases or analytical classes will be carried out.

For the implementation of the application, the use of Java Spring Boot technology is assumed, and the testing of important parts using unit and integration tests is assumed.

KEYWORDS

Web application, Input-output device, Java Spring Boot, analysis, requirements, use cases

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM UKÁZEK ZDROJOVÝCH KÓDŮ

SEZNAM TABULEK

SEZNAM ZKRATEK

API	Application
CSS	Cascading Style Sheets
DTO	Data Transfer Object
GNU	GNU’s Not Unix
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
IoT	Internet of Things
JSON	Java Script Object Notation
MVC	Model-view-controller
REST	Representational State Transfer
RTM	Release To Manufaturing
SIIO	Systém pro Integraci vstupně výstupních zařízení
URL	Uniform Resource Locator

ÚVOD

Uveďme si zde problém, který může nastat. Máme webovou aplikaci, která potřebuje komunikovat se vstupně-výstupním zařízením připojeným k hostitelskému počítači. Technologie, ve kterých jsou psány webové aplikace, neumožňují přímý přístup k těmto zařízením, je nutno tedy navrhnout řešení, jež tuto komunikaci umožní.

Předmětem této diplomové práce je propojení webové aplikace se vstupně-výstupními zařízeními. Existuje několik řešení, jak propojení realizovat, ale je nutné vybrat to nejvhodnější a řádně řešení otestovat.

Tato práce je zaměřena hlavně na propojení webové aplikace s tiskárnou a platebním terminálem. Implementace propojení s platebním terminálem podléhá interním předpisům bankovní společnosti, proto zde budou zmíněny operace pouze principiálně a nebudou zde žádné konkrétní příkazy, které komunikaci s platebním terminálem umožňují.

Jako konkrétní příklad nám zde slouží pokladní aplikace, která je realizována jako webová aplikace. Co potřebuje pokladní aplikace k svému chodu? Představme si, že jsme na nákupu a probíhá platba za nákup u pokladny. Aplikace musí samozřejmě umožňovat nějakým způsobem zaúčtovat zboží do systému (zadat cenu a množství zboží), musí umožňovat sečíst všechny položky nákupu, to není pro náš případ tak důležité. Po zaplacení nákupu obvykle obdržíte od pokladníka účtenku, že? Co je tedy dál třeba? Vytvořit účtenku a vytisknout ji. To je problém, na který jsme narazili právě tady. Webová aplikace sestaví účtenku bez problému, ale už nemá možnost ji jednoduše poslat k tisku. Platit můžeme hotově či platební kartou. Pro platbu kartou nám slouží platební terminál, který nám poskytla bankovní společnost. Platební terminál pak čte platební karty a realizuje samotnou platbu. Terminál má dvě možnosti funkčnosti. Platby se mohou zadávat ručně přímo na terminálu, případně může být terminál skrze síť přímo propojen s pokladní aplikací. V našem příkladu se využívá první možnost, při které je proces platby zdržován ručním zadáváním a při níž se mohou objevit chyby lidského faktoru. Pokladník může zadat částku na terminál špatně, zákazníkovi je poté ve výsledku naúčtována špatná částka a součty pokladny a terminálu pak nesouhlasí.

Tento konkrétní příklad je právě to, co mě vedlo k napsání diplomové práce na téma integrace webové aplikace se vstupně-výstupními zařízeními.

1 WEBOVÉ APLIKACE

Webová aplikace je počítačový program, který je umístěn na vzdáleném serveru. Koncovými uživateli se pak spouští prostřednictvím webového prohlížeče. Je kompatibilní s většinou standardních počítačů a operačních systémů a je dostupná z jakéhokoliv zařízení, které může daný uživatel používat (počítač, mobilní telefon, tablet). Je taktéž umožněno používat aplikaci více uživatelům současně. Webové aplikace jsou závislé na síťovém připojení, což bylo považováno za nedostatek, ovšem v současnosti, kdy je internet téměř všudypřítomný, tento fakt ztrácí na důležitosti. V podstatě kterákoliv služba nabízená skrze internet je podle definice formou webové aplikace. Jako příklad webové aplikace zde lze uvést online formuláře, internetové obchody (e-shopy), streamování videa, sociální média či email.

Obecně jsou webové aplikace rozděleny na frontend (klientská strana – to, co vidí uživatel) a backend (serverová strana – to, co běží na pozadí a není vidět). K vývoji webových aplikací lze využít různé programovací jazyky. [13]

1.1 Frontend

Jedná se o vše, co uživatel vidí a s čím může manipulovat v případě, kdy zadá webovou adresu či klikne na odkaz. Je to klientská část aplikace, která zahrnuje veškerý obsah a styly, tlačítka, různé efekty, formuláře, vyhledávací pole, rozvržení obsahu, text a barvy. Při vývoji této části se nejčastěji využívají technologie HTML, CSS a JavaScript, ke kterému lze připojit spoustu knihoven či frameworků. [4]

1.2 Backend

Zabývá se technologiemi, které jsou odpovědné za ukládání a bezpečnou manipulaci s uživatelskými daty. Je to část aplikace na straně serveru. Obsahuje veškerou skrytou logiku, kterou je poháněna aplikace, a v neposlední řadě zajišťuje, že vše funguje optimálně. Backend přijímá požadavky od klienta, zpracuje příchozí požadavek a zajistí příslušná data, která odesílá v odpovědi zpět klientovi. Součástí backendu může být databáze, jež slouží jako „mozek“ aplikace, jsou zde uložena potřebná data, ke kterým lze snadno a rychle přistupovat.

Při vývoji backendu lze využít různé programovací jazyky, nejčastěji to jsou PHP, Java nebo JavaScript, ke kterým lze přidat různé knihovny či frameworky. [7]

1.2.1 RestApi

API je sada pravidel, která definují, jak se mohou zařízení nebo aplikace připojit a komunikovat. REST API je pak rozhraní API, které je v souladu s principy návrhu REST. Lze je vyvíjet pomocí libovolného programovacího jazyka a podporuje různé formáty dat. Jediným požadavkem je, aby odpovídalo těmto principům návrhu REST:

1. **Jednotné rozhraní:** Všechny požadavky na API stejného zdroje by měly vypadat stejně, bez ohledu na to, odkud požadavek pochází. REST API by mělo zajistit, že stejná část dat, například jméno či mailová adresa uživatele, bude patřit pouze k jednomu identifikátoru zdroje.

2. **Rozdělení klienta a serveru:** Při návrhu REST API mají být klientské a serverové aplikace zcela nezávislé. Jedinou informací, kterou by měla klientská aplikace znát, je URL adresa požadovaného zdroje, nekomunikuje se serverem jiným způsobem. Obdobně by serverová aplikace neměla upravovat klientskou aplikaci jinak, než jí předávat požadovaná data přes HTTP.

3. **Bezstavovost:** REST API jsou bezstavová, každý požadavek musí obsahovat všechny informace pro jeho zpracování. Nevyžadují se žádné relace na straně serveru. Serverové aplikace by neměly ukládat žádná data související s požadavkem klienta.

4. **Možnost ukládání do mezipaměti:** Pokud je to možné, měly by být prostředky kešované na straně serveru nebo klienta. Odpovědi serveru by měly obsahovat informaci o tom, zda je pro dodaný zdroj povoleno ukládání do mezipaměti. Cílem je zlepšit výkon na straně klienta a zvýšit škálovatelnost na straně serveru.

5. **Architektura vrstveného systému:** Volání a odpovědi v REST API procházejí různými vrstvami. Obecně se nepředpokládá, že se klient a server připojují přímo k sobě. V komunikační smyčce může být řada různých prostředníků. Rozhraní REST API musí být navrženo tak, aby klient ani server nemohly zjistit, zda komunikují s koncovou aplikací, či prostředníkem.

6. **Kód na vyžádání:** Rozhraní obvykle odesílá statické prostředky, ale v určitých případech může odesílat také spustitelný kód. V těchto případech by měl kód běžet pouze na vyžádání.

[14]

1.2.1.1 Definice operací z hlediska metod HTTP

Protokol HTTP má definovaných několik metod, které přiřazují sémantický význam. Běžné metody, které využívá REST API jsou tyto:

- **GET** – slouží k načtení prostředku dle zadaného identifikátoru, tělo zprávy s odpovědí pak obsahuje podrobnosti o požadovaném prostředku.
- **POST** – vytváří nový prostředek se zadaným identifikátorem. Tělo zprávy v požadavku obsahuje podrobnosti k novému prostředku.
- **PUT** – vytvoří nebo nahradí prostředek se zadaným identifikátorem, kde tělo v požadavku pak určuje prostředek, který se má vytvořit či aktualizovat.
- **PATCH** – provádí částečnou aktualizaci prostředku, tělo takového požadavku pak určuje sadu změn, které se pro daný prostředek mají použít.
- **DELETE** – odebírá prostředek se zadaným identifikátorem.

[7]

1.2.1.2 HTTP stavové kódy

HTTP definuje standardní stavové kódy, které se používají k přenosu výsledku požadavku klienta a jsou rozděleny do několika kategorií:

- **1XX** – informativní, sdělují informace na úrovni přenosového protokolu.
- **2XX** – úspěšné, označují, že požadavek klienta byl úspěšně přijat.
- **3XX** – přesměrování, sdělují, že klient musí provést nějakou další akci, aby mohl dokončit požadavek.
- **4XX** – chyba klienta, chyby v této kategorii poukazují na to, že chyba nastala na straně klienta.
- **5XX** – chyba serveru, nastala chyba na straně serveru, odpovědnost za tyto chyby přebírá server.

Přehled nejpoužívanějších stavových kódů:

- **200** – Požadavek byl úspěšný (OK)
- **400 –** Špatný požadavek (Bad Request)
- **401** – Neoprávněný požadavek (Unauthorized)
- **404** – Server nemůže najít požadovaný zdroj (Not found)
- **500 –** Interní chyba serveru
- **501** – Metoda není serverem podporována (Neimplementováno)

[5]

2 VSTUPNĚ-VÝSTUPNÍ ZAŘÍZENÍ UVAŽOVANÉ V DIPLOMOVÉ PRÁCI

Kapitola se věnuje popisu vstupních a výstupních zařízení, způsobu a možnostem připojení k počítači a detailnějšímu popisu zařízení, která jsou součástí této práce.

2.1 Vstupní zařízení

Zařízení připojené k počítači za účelem zprostředkování komunikace mezi uživatelem a počítačem. Pomocí vstupního zařízení může uživatel zadávat jisté pokyny. Můžeme je rozdělit na ukazatelová (myš, touchpad), herní (joystick, volant, atp.), audiovizuální (scanner, webkamera, mikrofon) či textová (klávesnice). [6]

2.2 Výstupní zařízení

Slouží pro přenos dat z počítače směrem k uživateli. Nějakým způsobem zobrazuje data zpracovaná počítačem, případně jiným zařízením. Je to jeden ze způsobů, jak počítač sděluje výsledky své práce. Výsledek může zobrazovat pomocí obrazu, zvuku či fyzicky.

Zařízení, která se řadí do této kategorie, jsou tiskárny, monitory, reproduktory či sluchátka. [6]

2.3 Způsob připojení k počítači

Každé vstupně-výstupní zařízení má jiný způsob připojení k počítači. Zde je pár příkladů těch nejpoužívanějších způsobů.

2.3.1 USB

Univerzální sériová sběrnice nahrazující způsoby připojení, které se používaly v dřívějších časech (sériový port, paralelní port). USB konektor je každý z nás zvyklý používat téměř denně. Pomocí USB je dnes možné připojit tiskárny, klávesnice, myši, externí disky a spoustu jiných periferií. Je to rozhraní, které má budoucnost. Konektory prošly rozsáhlými změnami počínaje USB typem A, či typem B až po dnes asi nejpoužívanější USB typ C, který slouží pro přenos dat, či jako napájecí konektor pro mobilní telefony nebo notebooky. [11]

2.3.2 HDMI

Rozhraní sloužící především pro přenos zvukového a obrazového signálu v digitální podobě. Nejčastěji se touto zkratkou označuje slot, konektor nebo kabel. HDMI kabelem je možné

propojit televizi se set-top-boxem, počítač s monitorem či projektorem. HDMI v současné době patří mezi nejlepší řešení. Je univerzální, má srovnatelnou kvalitu s méně rozšířeným Display Portem a v neposlední řadě je kompatibilní s USB-C standardem. Oproti analogovým možnostem přenosu (VGA, SCART atp.) má výhody (kvalita, více možnosti rozlišení). [2]

2.3.3 Bluetooth

Bluetooth je standard bezdrátové komunikace, který umožňuje propojit dvě či více elektronických zařízení. Mezi taková zařízení může patřit mobilní telefon, tablet, sluchátka, klávesnice, myš a další. Nejnovější verze Bluetooth má dosah zhruba 40 metrů v interiéru a až 240 metrů venku. Data jsou přenášena na základě rádiových vln. Jediná podmínka pro funkčnost je vzdálenost, po celou dobu přenosu dat pak musí být všechna zařízení v dosahu. [1]

2.3.4 Počítačová síť

Počítačová síť představuje vzájemně propojená zařízení, která si mohou vyměňovat data a vzájemně sdílet zdroje. Tato síťová zařízení používají systém pravidel, nazývaných komunikační protokoly, k přenosu informací prostřednictvím fyzických či bezdrátových technologií. Základem počítačové sítě jsou uzly a linky (spojení). Uzel je zde představován síťovým zařízením (rozbočovač, přepínač atp.) nebo koncovým zařízením (počítač, mobilní telefon, tablet), spojení je přenosové médium, spojující dva uzly. Spojení může být fyzické (kabel), nebo „nefyzické“ v podobě bezdrátové technologie (WiFi). V této počítačové síti můžeme mít připojený klientský počítač a zároveň námi požadované vstupně-výstupní zařízení, prostřednictvím sítě mohou mezi sebou komunikovat. [16]

2.4 Vybraná zařízení

Předmětem diplomové práce je integrace pokladní webové aplikace se vstupně-výstupními zařízeními, proto je zde zahrnut detailnější popis konkrétních zařízení, se kterými pokladní aplikace bude komunikovat.

2.4.1 Tiskárna

Tiskárna je výstupní zařízení umožňující přenos digitálních dokumentů, obrázků, či jejich kombinaci do fyzické podoby (nejčastěji na papír). Mezi dva nejpoužívanější druhy tiskáren patří inkoustové tiskárny a laserové tiskárny. [9]

Dalším druhem, který je zajímavý s ohledem na tuto práci, je termotiskárna. Termotiskárny fungují na principu působení tepla, nepotřebují k fungování inkoustové náplně ani tonery, jsou velmi kompaktní a mají nízké provozní náklady. Takovýto typ tiskárny se používá k tisku štítků, účtenek či vstupenek. Své využití najde jak v obchodu, tak i v průmyslu. Termotiskárny mají jisté dva poddruhy, termotransferové tiskárny a termotiskárny pro přímý tisk. Termotransferová tiskárna je vybavena speciální teplocitlivou páskou, která se při působení tepla obarví a následně pak přenáší barvu na požadovaný materiál. U termotiskáren pro přímý tisk dochází k tisku na speciální papír, který obsahuje teplocitlivou vrstu. [21]

2.4.2 Platební terminál

Platební terminál je elektronické zařízení, díky kterému je možné realizovat bezhotovostní transakce platební kartou. Platba prostřednictvím platebního terminálu probíhá v několika krocích:

1. Obchodník zadá požadovanou částku k platbě.
2. Zákazník platební kartu přiloží k terminálu nebo vloží do terminálu.
3. Terminál skrze internet odesílá data o kartě a placené částce na server.
4. Server vyhodnotí platbu (zamítne ji, nebo přijme) a odešle výsledek zpět terminálu.
5. Terminál tiskne stvrzenku, zda platba proběhla korektně.

Existují zde také možnosti, že platební terminál je nastaven tak, že je ovládán přímo pokladní aplikací, která mu zadává příkazy prostřednictvím smluveného protokolu.

Platební terminál je poskytován formou pronájmu v hodnotě předem smluvené částky, takový pronájem pak zprostředkovává nejčastěji bankovní společnost a veškeré potřebné kroky k získání platebního terminálu řeší případný obchodník s bankovní společností. [12]

3 ANALÝZA SOUČASNÉHO ŘEŠENÍ

Tato část diplomové práce bude věnována analýze současného řešení integrace se vstupně-výstupními zařízeními. V kapitole je zjednodušený slovní popis současného řešení.

3.1 Zjednodušený popis současného řešení

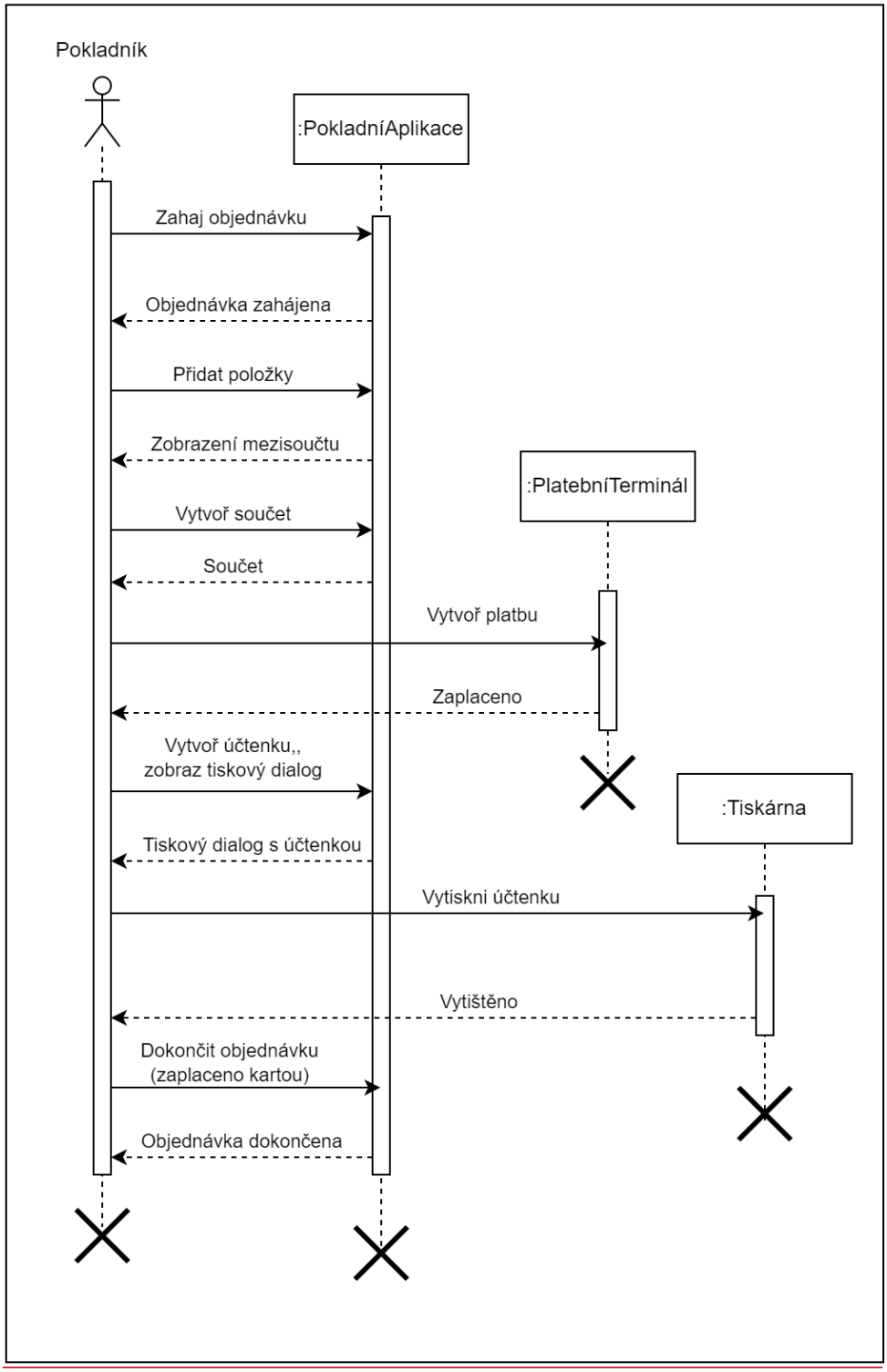
V současnosti máme pokladní webovou aplikaci, která funguje běžně, jen vznikají problémy při tisku účtenek nebo faktur a při platbách kartou.

Data pro tisk účtenek nebo faktur jsou ve formátu HTML, samotný tisk se pak realizuje zobrazením nového okna s daty jako HTML stránkou a následným vyvoláním okna pro tisk v prohlížeči (v podstatě nasimulování stisku tlačítek Ctrl + P), kde pak uživatel musí vybrat správnou tiskárnu a musí zde zvolit správné rozměry papíru. Často se stane, že požadovaná tiskárna není na prvním místě a taktéž že není uložena velikost požadovaného papíru, což způsobuje špatně vytištěný dokument případně i na špatné tiskárně. Tyto chyby se dají odstranit tím, že bude aplikace sama připravovat data k tisku a odesílat do příslušné tiskárny.

Platby kartou se musejí zadávat ručně na platebním terminálu a jednoduše se stane, že obsluha zadá požadovanou částku špatně. Vznikají pak chyby v součtech a nesedí uzávěrky, proto jsme se tyto chyby rozhodli limitovat tím, že bude pokladní aplikace přímo komunikovat s platebním terminálem.

3.2 Zjednodušený sekvenční diagram současného řešení

Zjednodušený diagram současného řešení v podstatě graficky kopíruje slovní popis předchozí podkapitoly.



Obrázek 1 - Sekvenční diagram současného řešení

3.3 Nevýhody současného řešení a jejich důsledky

a) Volba špatné tiskárny

V případě zvolení špatné tiskárny může nastat nevytištění účtenky (v případě, kdy zvolená tiskárna není aktivní), nebo může dojít k vytištění účtenky na jiné tiskárně (například na tiskárně v jiné místnosti). Vznikají pak komplikace při ukončení nákupu a zvyšuje se čekací doba zákazníka.

b) Volba špatné velikosti papíru

V dialogovém okně je nastavena velikost papíru pro tisk jinak, než by měla být, požadovaná účtenka je vytištěna špatně (může se stát, že není kompletní). Obsluha musí řešit opravu tisku, zákazník opět musí čekat a muže být nespokojený.

c) Špatně zadaná částka na terminálu

Obsluha pokladní aplikace zadá špatnou částku na terminál, obsluha ani zákazník si této chyby nevšimnou, platba proběhne. Nebude poté souhlasit porovnání součtů z pokladny a platebního terminálu (vzniknou komplikace a bude nutné dohledávat, kde nastala chyba). V případě, kdy chyba bude odhalena hned po platbě, bude nutné udělat návrat částky a provést platbu znovu. To představuje nadbytečné činnosti pro obsluhu i pro zákazníka a budoucí nespokojenost zákazníka.

d) Komplikovaná obsluha

Jak při ručním zadávání částky na terminál, tak při výběru tiskárny a příslušného papíru vzniká celkové zdržení v odbavení zákazníka. Práce pro obsluhu je zdlouhavá a není uživatelsky přívětivá. Zákazník čeká delší dobu, než by mohl.

Celkově se dá shrnout, že všechny tyto kroky jsou nepřívětivé jak z pohledu obsluhy pokladní aplikace, tak z pohledu zákazníka. Ve všech případech může nastat časové zdržení, které nemusí být příjemné ani pro jednu stranu. Důležité tady je, aby byla obsluha spokojená při práci s pokladní aplikací a aby odcházel spokojený zákazník.

4 ANALÝZA NOVÉHO ŘEŠENÍ

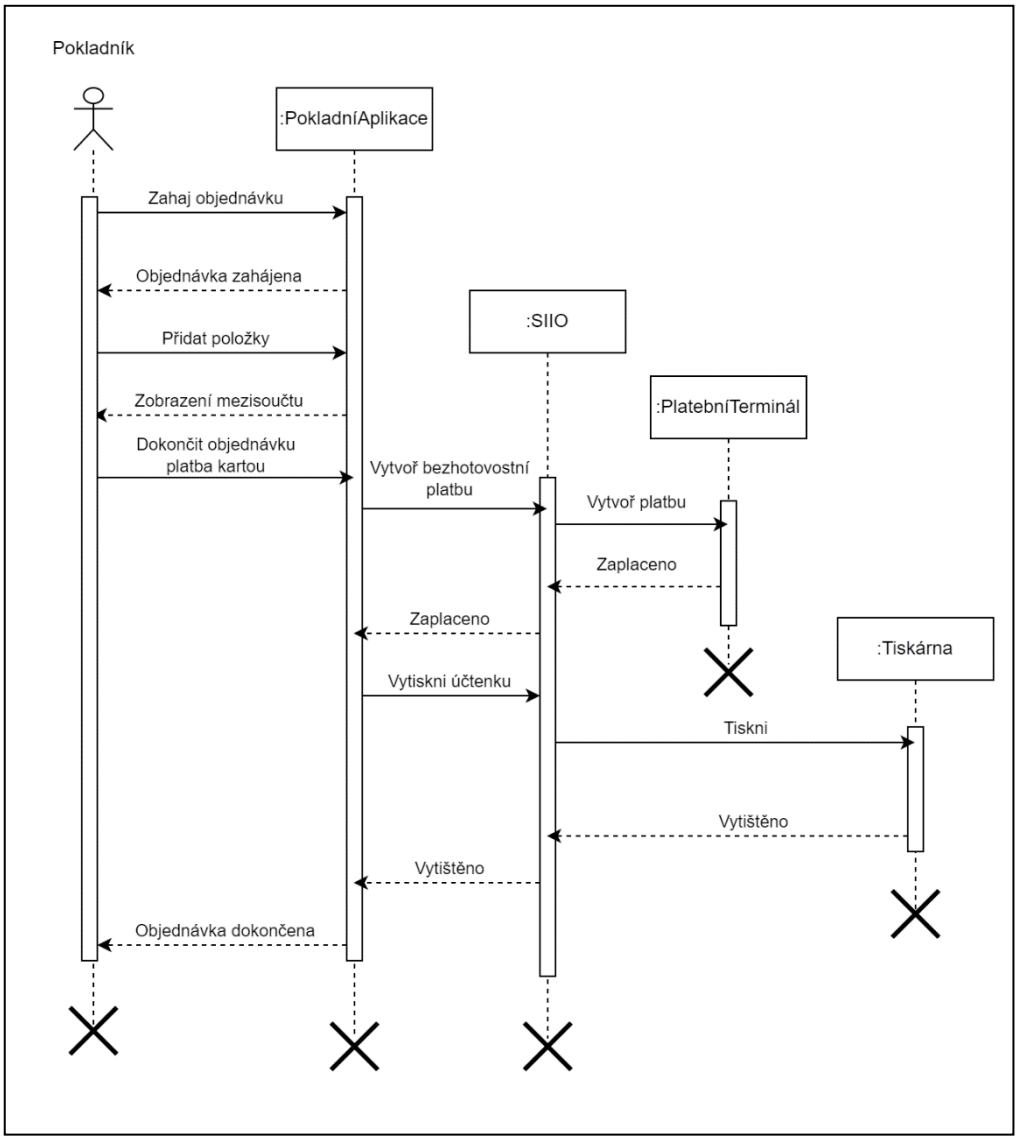
Tato část diplomové práce bude věnována analýze nového řešení integrace se vstupně-výstupními zařízeními. V kapitole je zjednodušený slovní popis současného řešení.

4.1 Zjednodušený popis nového řešení

Nové řešení má představit zjednodušení a zpřesnění práce s webovou aplikací. Návrhem řešení je aplikace spuštěná na hostitelském počítači, se kterou bude moci webová aplikace komunikovat skrze adresu 127.0.0.1 (localhost). Součástí aplikace by mělo být rozhraní, na které bude moci webová aplikace odesílat své požadavky, grafické uživatelské rozhraní, ve kterém bude umožněno nastavit parametry tiskáren a platebního terminálu. Celé toto řešení bude napsáno v jazyce Java 19 s využitím frameworku Spring Boot 2.7.5, pro grafické uživatelské rozhraní bude využita JavaFX 19. Aplikace bude sloužit pro nastavení parametrů vstupně-výstupních zařízení a pro komunikaci webové aplikace se vstupně-výstupními zařízeními. Bude dostupný instalační soubor, díky kterému bude možné tuto aplikaci nainstalovat na libovolný hostitelský počítač, na němž bude spuštěna pokladní webová aplikace.

4.2 Zjednodušený sekvenční diagram nového řešení

Následující diagram je grafickým vyjádřením slovního popisu nového řešení. Oproti současnému řešení zde přibyl objet SIIO (aplikace, která je předmětem této diplomové práce). Ubyla zde spousta operací, které musí provádět sám pokladník a které budou nyní prováděny za pomoci aplikace SIIO. Mělo by to zjednodušit práci pokladníka a omezit vznikající chyby.



Obrázek 2 - Sekvenční diagram nového řešení

4.3 Definice požadavků

Požadavek je základem pro každý systém. Jedná se vlastně o to, co by systém měl umět. Požadavky jsou formulovány bez specifikace způsobů, jakými budou dosaženy. Definice požadavků je velmi důležitou součástí, přesto bývá velmi podceňována a spousta problémů při vývoji systému je způsobena špatnou definicí požadavků, nebo nedefinováním požadavků. [19]

4.3.1 Funkční požadavky

Funkční požadavky určují, jaké bude chování systému a co by měl systém nabízet. Jsou označeny jedinečným identifikátorem, v tomto případě FR (Functional Requirement) a pořadovým číslem požadavku. [19]

- **FR01** – Systém bude poskytovat REST API pro komunikaci s webovou aplikací.

Webová aplikace bude odesílat požadavky na příslušný lokální server, který bude poskytovat rozhraní REST API.

- **FR02** – Systém bude umožňovat nastavení připojených zařízení prostřednictvím uživatelského rozhraní.

V systému bude k dispozici aplikace s grafickým uživatelským rozhraním, ve kterém bude umožněno uživateli nastavit potřebná zařízení, případně otestovat jejich funkčnost.

- **FR03** – Systém bude schopen komunikovat s tiskárnami a bude umožňovat tisk účtenek, faktur či jiných potřebných dokumentů.

Systém bude schopen vyhledat tiskárny připojené k počítači, ověřit, zda jsou schopny tisknout, připravit jim data k tisku a odeslat na ně data k tisku.

- **FR04** – Systém bude přijímat data pro tisk ve formátu HTML.

Data pro tisk jsou ve formátu HTML, backend pokladní webové aplikace generují data pro tisk na základě šablon Nette Latte, proto je nutné akceptovat data v takovém formátu.

- **FR05** – Systém bude schopen odesílat požadavky na platební terminál.

Platební terminál disponuje specifickým komunikačním protokolem, který podléhá přísným interním pravidlům bankovní společnosti. Zjednodušeně je nutné data obalit příslušnými hlavičkami a převést na pole bytů.

- **FR06** – Systém bude schopen zpracovávat odpovědi od platebního terminálu.

Platební terminál odesílá odpovědi za pomoci stejného komunikačního protokolu, proto systém musí umět příslušná data převést na data, která jsou čitelná uživatelem.

4.3.2 Nefunkční požadavky

Nefunkčními požadavky se rozumí vlastnosti či omezující podmínky systému. [19]

- **NFR01** – Systém bude napsán v jazyce Java ve verzi 19.
- **NFR02** – Systém bude využívat framework Spring Boot ve verzi 2.7.5.
- **NFR03** – Systém bude dostatečně rychlý a efektivní.
- **NFR04** – Systém bude spolehlivý a dostupný.
- **NFR05** – Systém bude umožňovat využívat integraci s více zařízeními.
- **NFR06** – Systém bude kompatibilní s operačním systémem Windows 10 a vyšším.
- **NFR07** – Součástí systému bude desktopová aplikace sloužící pro nastavení připojených zařízení.

4.4 Modelování případů užití (Use Case diagram)

Pod pojmem modelování případů užití je skryt jiný způsob definice požadavků. Skládá se z několika kroků:

1. **Nalezení hranic systému**[1]

Systém zde máme pouze jeden a tím je systém, který realizuje integraci se vstupně-výstupními zařízeními. Jedná se o systém, který je předmětem této diplomové práce.

2. **Vyhledání aktérů**[2]

Jako aktéry tohoto systému jsem zvolil: Pokladní webovou aplikaci a aplikaci pro konfiguraci zařízení. Pokladní webová aplikace představuje již existující pokladní aplikaci, pro kterou je nutno zprostředkovat komunikaci se vstupně-výstupními zařízeními. Aplikace pro konfiguraci zařízení představuje aplikaci běžící na lokálním počítači, která umožňuje nastavit příslušná zařízení pro jejich budoucí použití.

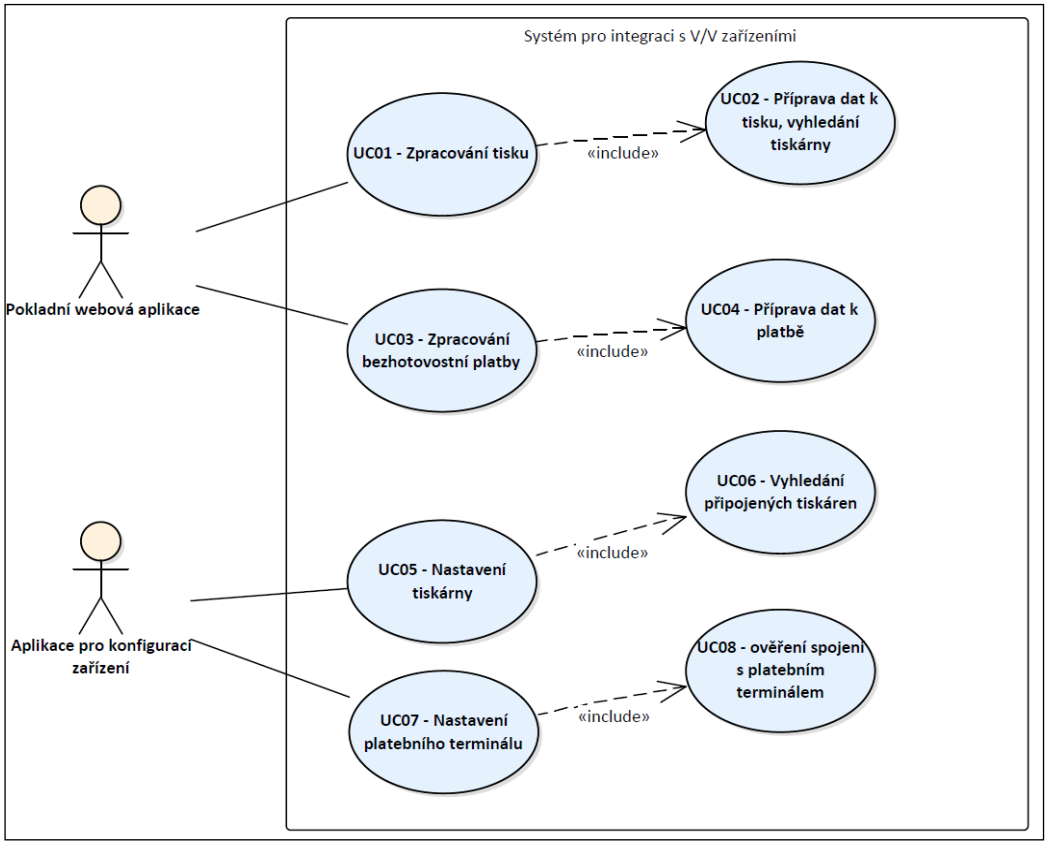
3. **Nalezení případů užití**[3]

Případy užití vycházejí z vytvořených funkčních požadavků a z jednotlivých kroků, které budou třeba pro realizaci jednotlivých operací. V modelu jsou reprezentovány modrým oválem, vždy mají v názvu jedinečný identifikátor (UC + pořadové číslo).

[1] Hranice modelovaného systému. V modelu ji představuje obdélník. [19]
[2] Role přidělené osobám či předmětům užívajícím systém. V modelu aktéra představuje obrázek panáčka. [19]
[3] Činnosti v systému vykonávané aktéry. [19]

4. **Vytvoření smysluplných relací**[4]

Relace představuje spojení jednotlivých případů užití mezi sebou, či mezi aktéry. Plná čára znázorňuje vztah mezi aktérem a případem užití. Přerušovaná čára <<*include*>> je spojení mezi případy užití, které znamená, že první případ užití zahrnuje druhý případ užití neboli že před samotným provedením prvního případu užití musí být vykonán druhý případ užití.



Obrázek 3 - UseCase diagram navrhovaného řešení

4.4.1 Popis modelu.

Model případů užití se skládá z osmi případů užití a ze dvou aktérů. Aktéry zde představují dvě aplikace, které budou komunikovat se samotným systémem SIIO. Pokladní webová aplikace zde představuje aplikaci, které je spuštěna ve webovém prohlížeči a potřebuje komunikovat se vstupně-výstupními zařízeními připojenými k počítači. Aplikace pro konfiguraci zařízení je desktopová aplikace, které slouží ke konfiguraci připojených zařízení prostřednictvím systému SIIO. Jednotlivé případy užití pak představují operace, které budou probíhat v systému SIIO, čáry uvnitř systému definují, jak spolu budou jednotlivé součásti komunikovat.

[4] Vztahy mezi aktéry a případy užití. [19]

4.5 Scénáře případů užití

Jedná se o jednotlivé kroky daného případu užití, scénář lze také nazvat tokem událostí. Scénáře jsou hlavní a alternativní. Hlavní scénář obsahuje kroky, které zachycují ideální řešení, v němž nedochází k žádným chybám. Alternativní scénář pak popisuje kroky, kdy je zaznamenána chyba, dojde k rozvětvení nebo přerušení hlavního scénáře. Každý případ užití pak obsahuje jeden hlavní scénář a může obsahovat několik alternativních scénářů. Alternativní scénáře se obvykle zpět k hlavnímu nevracejí. [19]

V následujících tabulkách je vždy pro každý UseCase uveden jeden hlavní scénář. Co se týče alternativních scénářů, je jich z pravidla více, ale z důvodu přehlednosti jsem přiložil vždy pouze jeden.

4.5.1 UC01 – Zpracování tisku

Případ užití:	Zpracování tisku
ID:	UC01
Stručný popis:	Systém SIIO provede tisk požadovaných dat na požadované tiskárně
Aktéři:	Pokladní webová aplikace
Vstupní podmínky:	Žádné
Hlavní scénář:	1. Pokladní webová aplikace odešle požadavek na tisk 2. Systém SIIO přijme požadavek 3. Systém SIIO provede přípravu dat k tisku a vyhledání příslušné tiskárny 4. Systém SIIO připravená data odešle na příslušnou tiskárnu 5. Systém SIIO zpracuje výsledek tisku a odešle odpověď pokladní webové aplikaci 6. Pokladní webová aplikace přijme výsledek operace a provede dál příslušné kroky
Výstupní podmínky:	1. Systém SIIO provedl validaci a přípravu dat 2. Systém SIIO odeslal data k tisku
Alternativní scénáře:	Nevalidní data Nenastavená tiskárna

Tabulka 1 - Hlavní scénář UC01

Případ užití:	Zpracování tisku: neplatná data
ID:	UC01.1
Stručný popis:	Systém SIIO sděluje webové aplikaci, že nelze provést tisk.
Aktéři:	Pokladní webová aplikace
Vstupní podmínky:	Pokladní webová aplikace odeslala nevalidní data k tisku
Alternativní scénář:	1. Pokladní webová aplikace odešle požadavek na tisk 2. Systém SIIO přijme požadavek 3. Systém SIIO neprovede přípravu dat k tisku a vyhledání příslušné tiskárny 4. Systém SIIO připravená data odešle na příslušnou tiskárnu 5. Systém SIIO odešle odpověď pokladní webové aplikaci (neplatná data) 6. Pokladní webová aplikace přijme výsledek operace a provede další kroky
Výstupní podmínky:	1. Systém SIIO provedl validaci a přípravu dat 2. Systém SIIO neodeslal data k tisku

Tabulka 2 - Vedlejší scénář (neplatná data) UC01

4.5.2 UC02 – Příprava dat k tisku, vyhledání tiskárny

Případ užití:	Příprava dat k tisku, vyhledání tiskárny
ID:	UC02
Stručný popis:	Systém SIIO provede přípravu dat a vyhledání příslušné tiskárny
Aktéři:	Pokladní webová aplikace
Vstupní podmínky:	Zadaná data jsou validní, tiskárna je aktivní
Hlavní scénář:	1. Systém SIIO přijme požadavek na přípravu dat a vyhledání tiskárny 2. Systém SIIO provede převod přijatých dat ve formátu HTML na obrázek 3. Systém SIIO vyhledá příslušnou tiskárnu a ověří její stav 4. Systém SIIO pokračuje v dalších krocích
Výstupní podmínky:	Systém SIIO provedl validaci a přípravu dat Systém SIIO vyhledat tiskárnu a ověřil její stav
Alternativní scénáře:	Neaktivní tiskárna

Tabulka 3 - Hlavní scénář UC02

Případ užití:	Příprava dat k tisku, vyhledání tiskárny: neaktivní tiskárna
ID:	UC02.1
Stručný popis:	Systém SIIO sděluje webové aplikaci, že nelze provést tisk, protože tiskárna není aktivní
Aktéři:	Pokladní webová aplikace
Vstupní podmínky:	Pokladní webová aplikace požaduje tisk na neaktivní tiskárně
Alternativní scénář:	1. Systém SIIO přijme požadavek na přípravu dat a vyhledání tiskárny 2. Systém SIIO provede validaci a převod dat 3. Systém SIIO vyhledá tiskárnu a ověří její stav (tiskárna není aktivní) 4. Systém SIIO nepokračuje v dalších krocích a vrací chybovou hlášku (tiskárna není aktivní)
Výstupní podmínky:	Systém SIIO provedl validaci a přípravu dat Systém SIIO vyhledal tiskárnu a ověřil její stav (neaktivní)

Tabulka 4 - Vedlejší scénář (tiskárna není aktivní) UC02

4.5.3 UC03 – Zpracování bezhotovostní platby

Případ užití:	Zpracování bezhotovostní platby
ID:	UC03
Stručný popis:	Systém SIIO odešle požadavek platby na platební terminál
Aktéři:	Pokladní webová aplikace
Vstupní podmínky:	Zadaná data jsou validní, platební terminál je aktivní
Hlavní scénář:	1. Pokladní webová aplikace odešle požadavek na vytvoření bezhotovostní platby 2. Systém SIIO přijme požadavek 3. Systém SIIO připraví data pro platbu (konverze do formátu, který je akceptován příslušným protokolem) 4. Systém SIIO ověří funkčnost platebního terminálu 5. Systém SIIO odešle připravená data platebnímu terminálu 6. Platební terminál zpracuje bezhotovostní platbu a odešle systému SIIO její výsledek 7. Systém SIIO obdrží odpověď s výsledkem platby 8. Systém SIIO zpracuje výsledek platby 9. Systém SIIO odešle výsledek platby pokladní webové aplikaci 10. Pokladní webová aplikace přijme výsledek platby
Výstupní podmínky:	Systém SIIO provedl validaci a přípravu dat Systém SIIO ověřil stav platebního terminálu Systém zprostředkoval bezhotovostní platbu
Alternativní scénáře:	Nevalidní data pro platbu Neaktivní platební terminál

Tabulka 5 - Hlavní scénář UC03

Případ užití:	Zpracování bezhotovostní platby: nevalidní data pro platbu
ID:	UC03.1
Stručný popis:	Systém SIIO sděluje webové aplikaci, že nelze provést platbu a odesílá jí stav platebního terminálu
Aktéři:	Pokladní webová aplikace
Vstupní podmínky:	Pokladní webová aplikace odešle nevalidní data pro platbu
Alternativní scénář:	1. Pokladní webová aplikace odešle požadavek na vytvoření bezhotovostní platby 2. Systém SIIO přijme požadavek 3. Systém SIIO připraví data pro platbu (konvertování do formátu akceptovaného příslušným protokolem) 4. Systém SIIO ověří funkčnost platebního terminálu, při kterém zjistí, že terminál není připojen nebo není ve stavu, ve kterém je schopen platbu zpracovat 5. Systém SIIO odešle stav terminálu pokladní webové aplikaci 6. Pokladní webová aplikace přijme výsledek
Výstupní podmínky:	Systém SIIO provedl validaci a přípravu dat pro platbu Systém SIIO ověřil funkčnost platebního terminálu (neaktivní)

Tabulka 6 - Alternativní scénář UC03 (terminál není aktivní)

4.5.4 UC04 – Příprava dat k platbě, ověření stavu platebního terminálu

Případ užití:	Příprava dat k platbě, ověření stavu platebního terminálu
ID:	UC04
Stručný popis:	Systém SIIO provede přípravu dat k platbě a ověření stavu platebního terminálu
Aktéři:	Pokladní webová aplikace
Vstupní podmínky:	Zadaná data jsou validní, platební terminál je aktivní
Hlavní scénář:	1. Systém SIIO přijme požadavek na přípravu dat k platbě a ověření stavu terminálu 2. Systém SIIO provede ověření stavu platebního terminálu – terminál je připraven 3. Systém SIIO provede konverzi dat k platbě (převod do formátu čitelného v daném protokolu) 4. Systém SIIO pokračuje v dalších krocích
Výstupní podmínky:	Systém SIIO provedl validaci a přípravu dat Systém SIIO ověřil stav platebního terminálu
Alternativní scénáře:	Nevalidní data pro platbu Neaktivní platební terminál

Tabulka 7 - Hlavní scénář UC04

Případ užití:	Příprava dat k platbě, ověření stavu platebního terminálu: Nevalidní data pro platbu
ID:	UC04.1
Stručný popis:	Systém SIIO sděluje webové aplikaci, že nelze provést platbu a odesílá jí chybovou hlášku (nevalidní data pro platbu)
Aktéři:	Pokladní webová aplikace
Vstupní podmínky:	Pokladní webová aplikace odešle nevalidní data pro platbu
Alternativní scénář:	1. Systém SIIO přijme požadavek na přípravu dat k platbě a ověření stavu terminálu 2. Systém SIIO provede validaci dat k platbě (data nejsou validní) 3. Systém SIIO nepokračuje v dalších krocích
Výstupní podmínky:	Systém SIIO provedl validaci a přípravu dat pro platbu (nevalidní data)

Tabulka 8 - Vedlejší scénář UC04 (nevalidní data)

4.5.5 UC05 – Nastavení tiskárny

Případ užití:	Nastavení tiskárny
ID:	UC05
Stručný popis:	Systém SIIO provede změnu nastavení tiskárny
Aktéři:	Aplikace pro konfiguraci zařízení
Vstupní podmínky:	Zadaná data jsou validní, zadaná platební tiskárna existuje
Hlavní scénář:	1. Aplikace pro konfiguraci zařízení odešle požadavek na změnu nastavení tiskárny 2. Systém SIIO přijme požadavek na změnu nastavení tiskárny 3. Systém SIIO provede vyhledání existujících tiskáren připojených k počítači (požadovaná tiskárna je k počítači připojena) 4. Systém SIIO provede validaci nového nastavení tiskárny 5. Systém SIIO provede otevření potřebného konfiguračního souboru 6. Systém SIIO uloží data do konfiguračního souboru 7. Systém SIIO odešle výsledek změny nastavení 8. Aplikace pro konfiguraci zařízení přijme výsledek o změně nastavení
Výstupní podmínky:	Systém SIIO provedl validaci a přípravu dat Systém SIIO ověřil stav platebního terminálu
Alternativní scénáře:	Neexistující tiskárna

Tabulka 9 - Hlavní scénář UC05

Případ užití:	Nastavení tiskárny: neexistující tiskárna
ID:	UC05.1
Stručný popis:	Systém SIIO sděluje aplikaci pro konfiguraci, že nelze provést tisk a odesílá jí chybovou hlášku (neexistující tiskárna)
Aktéři:	Aplikace pro konfiguraci zařízení
Vstupní podmínky:	Aplikace pro konfiguraci zařízení požaduje nastavit neexistující tiskárnu
Alternativní scénář:	1. Aplikace pro konfiguraci zařízení odešle požadavek na změnu nastavení tiskárny 2. Systém SIIO přijme požadavek na změnu nastavení tiskárny 3. Systém SIIO se pokusí vyhledat požadovanou tiskárnu v existujících tiskárnách (tiskárna není připojena k počítači) 4. Systém SIIO odešle výsledek změny nastavení (neprovedeno, neznámá tiskárna) 5. Aplikace pro konfiguraci přijme výsledek o neprovedení změny nastavení
Výstupní podmínky:	Systém SIIO nenalezl požadovanou tiskárnu

Tabulka 10 - Vedlejší scénář UC05 (neexistující tiskárna)

4.5.6 UC06 – Vyhledání připojených tiskáren

Případ užití:	Vyhledání připojených tiskáren
ID:	UC06
Stručný popis:	Systém SIIO provede vyhledání tiskárny v seznamu tiskáren připojených k počítači
Aktéři:	Aplikace pro konfiguraci zařízení
Vstupní podmínky:	Žádné
Hlavní scénář:	1. Systém SIIO přijme požadavek na vyhledání požadované tiskárny 2. Systém SIIO vyhledá v seznamu tiskáren připojených k počítači požadovanou tiskárnu 3. Systém SIIO pokračuje v dalších krocích
Výstupní podmínky:	Systém ověřil, zda je požadovaná tiskárna připojena k počítači
Alternativní scénáře:	Požadovaná tiskárna neexistuje

Tabulka 11 - Hlavní scénář UC06

Případ užití:	Vyhledání připojených tiskáren: požadovaná tiskárna neexistuje
ID:	UC06.1
Stručný popis:	Systém SIIO sděluje webové aplikaci, že nelze nalézt tiskárnu a odesílá jí chybovou hlášku (tiskárna neexistuje)
Aktéři:	Aplikace pro konfiguraci zařízení
Vstupní podmínky:	Aplikace pro konfiguraci zařízení požaduje nastavit neexistující tiskárnu
Alternativní scénář:	1. Systém SIIO přijme požadavek na vyhledání požadované tiskárny 2. Systému SIIO se nepodaří nalézt požadovanou tiskárnu v seznamu existujících tiskáren 3. Systém SIIO dál nepokračuje v dalších krocích
Výstupní podmínky:	Systém nenalezl požadovanou tiskárnu v seznamu připojených tiskáren

Tabulka 12 - Vedlejší scénář UC06 (požadovaná tiskárna neexistuje)

4.5.7 UC07 – Nastavení platebního terminálu

Případ užití:	Vyhledání připojených tiskáren
ID:	UC07
Stručný popis:	Systém SIIO provede změnu nastavení platebního terminálu
Aktéři:	Aplikace pro konfiguraci
Vstupní podmínky:	Zadané parametry jsou validní, konfigurační soubor existuje
Hlavní scénář:	1. Aplikace pro konfiguraci zařízení odešle požadavek na změnu nastavení platebního terminálu 2. Systém SIIO přijme požadavek na změnu nastavení platebního terminálu 3. Systém SIIO provede validaci zadaných parametrů 4. Systém SIIO vyhledá konfigurační soubor pro platební terminály 5. Systém SIIO provede uložení nové konfigurace do nalezeného souboru 6. Systém SIIO ověří spojení s platebním terminálem 7. Systém SIIO vrací výsledek operace uložení nové konfigurace s výsledkem ověření spojení s platebním terminálem (uloženo, terminál připojen) 8. Aplikace přijímá výsledek požadované operace
Výstupní podmínky:	Systém SIIO provedl změnu nastavení platebního terminálu
Alternativní scénáře:	Zadané parametry nejsou validní, konfigurační soubor neexistuje

Tabulka 13 - Hlavní scénář UC07

Případ užití:	Vyhledání připojených tiskáren: konfigurační soubor neexistuje
ID:	UC07.1
Stručný popis:	Systém SIIO sděluje aplikaci pro konfiguraci, že nelze provést změnu nastavení a odesílá jí chybovou hlášku (neexistuje konfigurační soubor)
Aktéři:	Aplikace pro konfiguraci zařízení
Vstupní podmínky:	Aplikace pro konfiguraci odešle požadavek na změnu nastavení ve chvíli, kdy neexistuje konfigurační soubor
Alternativní scénář:	1. Aplikace pro konfiguraci zařízení odešle požadavek na změnu nastavení platebního terminálu 2. Systém SIIO přijme požadavek na změnu nastavení platebního terminálu 3. Systém SIIO provede validaci zadaných parametrů 4. Systému SIIO se nepodaří nalézt konfigurační soubor pro platební terminály 5. Systém SIIO vrací výsledek ukládání a ověření stavu platebního terminálu (neuloženo, neověřeno)
Výstupní podmínky:	Systém SIIO nenalezl konfigurační soubor

Tabulka 14 - Alternativní scénář UC07 (neexistuje konfigurační soubor)

4.5.8 UC08 – Ověření spojení s platebním terminálem

Případ užití:	Ověření spojení s platebním terminálem
ID:	UC08
Stručný popis:	Systém SIIO ověří spojení s platebním terminálem
Aktéři:	Aplikace pro konfiguraci
Vstupní podmínky:	Zadané parametry jsou validní, konfigurační soubor existuje
Hlavní scénář:	1. Systém SIIO přijme požadavek na ověření spojení s platebním terminálem 2. Systém SIIO vyhledá aktuální konfiguraci pro spojení s platebním terminálem 3. Systém SIIO se pokusí navázat spojení s platebním terminálem (spojení navázáno) 4. Systému SIIO ukončí spojení s platebním terminálem 5. Systém SIIO pokračuje v dalších krocích
Výstupní podmínky:	Systém SIIO provedl změnu nastavení platebního terminálu
Alternativní scénáře:	Spojení se nepodařilo navázat Konfigurační soubor neexistuje

Tabulka 15 - Hlavní scénář UC08

Případ užití:	Ověření spojení s platebním terminálem: spojení se nepodařilo navázat
ID:	UC08.1
Stručný popis:	Systém SIIO sděluje aplikaci pro konfiguraci, že nelze navázat spojení s platebním terminálem
Aktéři:	Aplikace pro konfiguraci zařízení
Vstupní podmínky:	Aplikace pro konfiguraci odešle požadavek na ověření spojení ve chvíli, kdy platební terminál není schopen komunikovat
Alternativní scénář:	1. Systém SIIO přijme požadavek na ověření spojení s platebním terminálem 2. Systém SIIO vyhledá aktuální konfiguraci pro spojení s platebním terminálem 3. Systém SIIO se pokusí navázat spojení s platebním terminálem (spojení nenavázáno) 4. Systém SIIO nepokračuje v dalších krocích
Výstupní podmínky:	Systému SIIO se nepodařilo navázat spojení s platebním terminálem

Tabulka 16 - Vedlejší scénář UC08 (spojení se nepodařilo navázat)

4.6 Matice sledovatelnosti

Jedná se o užitečný nástroj pro ověření konzistence. Používá se pro ni zkratka RTM. Je to matice, která má na jedné ose identifikátory jednotlivých požadavků a na ose druhé názvy případů užití (případně jejich identifikátory). V jednotlivých buňkách matice je označeno, zda daný případ užití splňuje odpovídající požadavek. Pomocí matice sledovatelnosti můžeme sledovat, zda v našem modelu nemáme nějaký požadavek, který není mapován na žádný případ užití, z čehož pak můžeme usoudit, že případy užití nejsou kompletní. Lze tak usoudit i v opačném případě, kdy případ užití není mapován na žádný požadavek, pak může nějaký požadavek v systému chybět. [19]

	Requirements::FR01	Requirements::FR02	Requirements::FR03	Requirements::FR04	Requirements::FR05	Requirements::FR06
useCase::UC01 - Zpracování tisku	↑		↑			
useCase::UC02 - Příprava dat k tisku, vyhle...				↑		
useCase::UC03 - Zpracování bezhotovostn...	↑				↑	↑
useCase::UC04 - Příprava dat k platbě					↑	
useCase::UC05 - Nastavení tiskárny		↑				
useCase::UC06 - Vyhledání připojených tisk...			↑			
useCase::UC07 - Nastavení platebního ter...		↑				
useCase::UC08 - ověření spojení s platební...					↑	↑

Obrázek 4 - Matice sledovatelnosti navrhovaného systému

Všechny případy užití jsou mapovány na jeden či více požadavků, v opačném případě všechny požadavky jsou mapovány na jeden či více případů užití. Z čehož vyplývá, že případy užití i požadavky by měly být kompletní.

4.7 Analytický model

Analytický model se zaměřuje na to, co systém udělat musí, ale nezabývá se detaily, které se týkají způsobu, jak to udělat. Tato otázka je realizována v aktivitě návrh. Celkově spočívá v tvorbě modelů, které zachycují podstatné požadavky a charakteristické rysy požadovaného systému. Analytický model je vytvořen v jazyce daného odvětví, zachycuje problém z dané perspektivy, v podstatě vypráví příběh o požadovaném systému. V analytickém modelu nepřijímáme žádná rozhodnutí, která se týkají samotné implementace. Samotný model by měl být co nejjednodušší. Analytický model se skládá z tříd a relací.

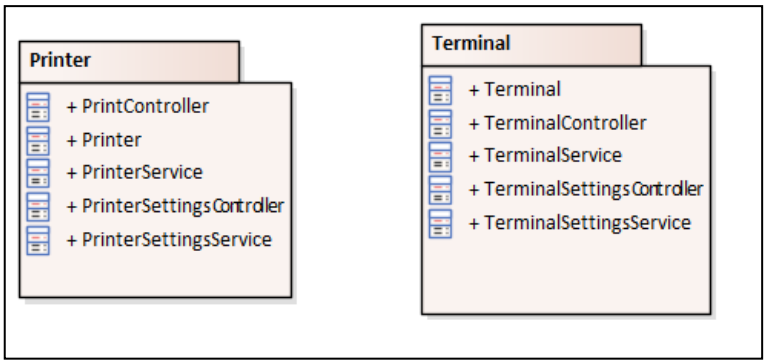
Třída představuje předpis pro skupinu objektů, které mají stejné rysy, sdílejí atributy a chování. Každý objekt pak je instancí přesně jedné třídy. Jiným způsobem by se dala třída označit jako šablona objektů. Třída je reprezentována obdélníkem, v horní části je název třídy, zbylá část je rozdělena na dvě části. Horní část je oddílem atributů, kterými je instance dané třídy identifikována, dolní část je oddílem operací, které objekt, jenž je instancí dané třídy, může vykonávat. Atributy i operace mají různé typy viditelnosti (+ je veřejný, - soukromý a # chráněný).

Relace je spojením mezi třídami, umožňuje komunikaci mezi zúčastněnými třídami a určuje, jakým způsobem mezi sebou mohou komunikovat a jak jsou na sobě závislé. Relací je několik typů. Mezi základní patří:

- **Asociace** – relace mezi třídami, sdružuje třídy. Aby mohlo existovat spojení mezi instancemi, musí existovat asociace mezi jejich třídami. Jednoduše značí, že mezi objekty, které jsou instancí daných tříd, lze vytvářet spojení. Asociace je znázorněna plnou čárou
- **Závislost –** relace mezi dvěma prvky, kdy změna jednoho prvku se promítá do druhého. Je znázorněna tečkovanou čárou

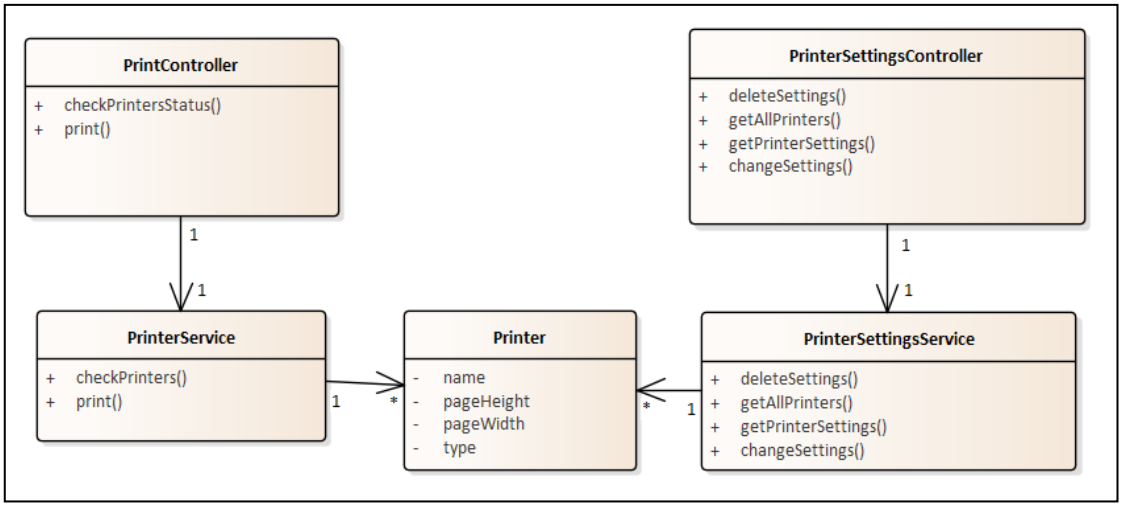
Tyto analytické modely mohou být rozděleny do balíčků. Balíčkem se rozumí sdružení modelovaných prvků. Umožňuje uspořádat prvky a diagramy do skupin.

[19]



Obrázek 5 - Diagram balíčků analytického modelu

V modelu, který je součástí této diplomové práce byly vytvořeny dva analytické balíčky (viz Obrázek 5), kde každý z nich obsahuje analytický model pro danou část: Část týkající se tiskáren a část týkající se platebních terminálů. Dané balíčky pak obsahují konkrétní analytický model, který přibližuje funkčnost modulu.



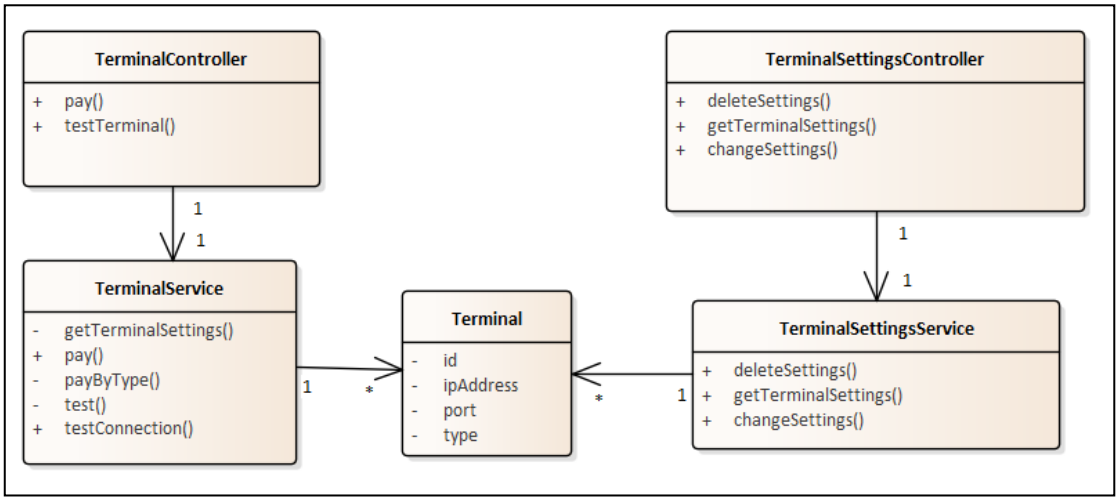
Obrázek 6 - Analytický model balíčku tiskárny

Analytický model balíčku tiskárny se skládá z pěti tříd a jejich vzájemných relací. Tyto oddělené prvky celkově obstarávají pouze operace týkající se tiskáren, které jsou součástí systému.

Třídy se sufixem **Controller** představují kontroléry, které zprostředkovávají komunikaci mezi webovou aplikací a samotnou logikou provádění potřebných operací.

- **Třída PrintController** – slouží pro volání operací spojených s tiskárnami v systému.
- **Třída PrinterSettingsController** – slouží pro poskytování operací týkajících se konfigurace tiskáren v systému.
- **Třída PrintService** – obstarává operace potřebné k provedení tisku.

- **Třída PrinterSettingsController** – vykonává operace potřebné ke konfiguraci připojených tiskáren.
- **Třída Printer** – představuje předpis pro entity typu tiskárna a uchovává potřebné parametry pro identifikaci tiskárny v systému.



Obrázek 7 - Analytický model balíčku terminálu

Analytický balíček terminálu obsahuje pět tříd a relace mezi nimi. Jednotlivé prvky obstarávají pouze operace, které se týkají platebního terminálu.

Sufixem Controller u tříd se rozumí kontrolér, který poskytuje operace skrze REST API a zprostředkovává tak komunikaci mezi webovou aplikací a logickou částí aplikace.

- Třída **TerminalController** – poskytuje metody, které zpřístupňují operace s platebním terminálem.
- Třída **TerminalService** – realizuje jednotlivé operace, které jsou potřeba pro komunikaci s platebním terminálem.
- Třída **TerminalSettingsController** – poskytuje metody, které zpřístupňují operace nutné pro konfiguraci platebního terminálu.
- Třída **TerminalSettingsService** – realizuje operace pro konfiguraci platebního terminálu.
- Třída **Terminal** – jedná se o entitu reprezentující předpis pro konfiguraci platebního terminálu, který disponuje atributy, jež jsou zároveň parametry potřebnými pro připojení k platebnímu terminálu.

5 POUŽITÉ NÁSTROJE A TECHNOLOGIE

Kapitola je věnována všem použitým nástrojům a technologiím, které mi byly nápomocny při psaní této diplomové práce.

5.1 Enterprise Architect

Jedná se o nástroj sloužící pro modelování celého životního cyklu pro:

- Obchodní a IT systémy
- Software a systémové inženýrství
- Vývoj v reálném čase a vestavěný vývoj

Nástroj pomáhá sledovat specifikace na vysoké úrovni až po modely analýzy, návrhu, implementace a údržby pomocí UML, SysML, BPMN a dalších otevřených standardů. Enterprise Architect je navržen tak, aby pomohl vybudovat robustní a udržovatelné systémy.

Nástroj jsem využil k analýze nového řešení. Navrhl jsem v něm požadavky, model případů užití včetně návrhu scénářů. Za pomoci tohoto nástroje vznikla matice sledovatelnosti a taktéž analytický model. Nástroj mi byl velkým pomocníkem při této části práce. [3]

5.2 Java

Java je programovací jazyk, je multiplatformní a objektově orientovaný. Funguje na miliardách zařízení na světě. Pojem multiplatformní zde představuje schopnost přenést program napsaný v jazyce Java na jakoukoliv platformu (není tedy vázán například na použití pouze na Windows). Jedná se o objektově orientovaný jazyk, tudíž má uspořádaný kód do tříd a objektů. Většina moderních programovacích jazyků (C++, C# či Python) jsou objektově orientované. Co se týče využití tohoto jazyka, lze ho využít téměř kdekoliv (mobilní aplikace, webové aplikace, hry, podnikový software či aplikace IoT).

Pokud programátor napíše kód v jazyku Java, je tento kód následně sadou (Java Development Kit) přeložen do počítačového kódu, který lze přečíst na libovolném zařízení. Součástí zařízení musí být softwarová součást zvaná kompilátor. Kompilátor přebírá počítačový kód a překládá ho do bajtového kódu, kterému rozumí každý operační systém. Tento bajtový kód je následně zpracován překladačem (prostředí Java Virtual Machine – JVM). JVM je zpravidla k dispozici

pro většinu softwarových i hardwarových platforem, díky nim lze zdrojový kód v jazyce Java snadno přenášet na libovolné zařízení.

Přestože jazyk Java existuje již přes 20 let, zůstává pořád nejdůležitějším jazykem, protože je výjimečně univerzální a lze se poměrně snadno naučit. [20]

5.3 Spring Boot

Nejprve je nutné si upřesnit, co je to Spring, protože Spring a Spring Boot nejsou totožné. Spring je otevřené aplikační prostředí, které je založena na jazyku Java. Spring Boot je modul, který vychází z prostřední Spring. Spring a Spring Boot jdou tedy vlastně ruku v ruce a je to spojení výhod obou řešení.

Mezi **klíčové funkce** modulu Spring Boot patří:

- Samostatné aplikace – pomáhá vytvářet aplikace, které nejsou závislé na dané platformě a je možné je spustit i na zařízení, které nemá přístup k internetu.
- Integrované servery – umožňuje přímo vkládat servery jako je Tomcat, Jetty nebo Undertow.
- Automatická konfigurace – automaticky konfiguruje Spring, případně další knihovny třetích stran, kdykoliv to je třeba.
- Konkrétní přístup – usnadnění konfigurace sestavení aplikace nabídnutím konkrétních počátečních závislostí.
- Funkce připravené pro provoz – nabízí funkce, které jsou ihned připraveny pro provoz (metriky, kontroly stavu či externí konfigurace).

Spring Boot je postaven na MVC architektuře, aplikace je díky tomu rozdělena na 3 části: uživatelské rozhraní, logika a komunikační vrstva, která tyto 2 vrstvy propojuje. [22]

5.4 JavaFX

JavaFX je sada mediálních a grafických balíčků, která umožňuje vývojářům navrhovat a vytvářet klientské aplikace na různých platformách. Knihovna JavaFX je psána jako Java API, proto může kód JavaFX odkazovat na API z libovolné knihovny jazyka Java. Vzhled a chování lze upravit pomocí CSS. Odděluje vývoj grafického uživatelského rozhraní od samotné implementace aplikace. Grafické části se může věnovat jeden programátor a funkční části pak

druhý programátor. Za pomoci JavaFX lze vytvářet mnoho typů aplikací. JavaFX je součástí Java SE Runtime Environment (JRE) a Java Development Kit (JDK). Jelikož JDK je možné spouštět na všech platformách, lze i JavaFX aplikaci spouštět na všech platformách.

Některé z vlastností:

- Integrované ovládací prvky – poskytuje všechny hlavní ovládací prvky uživatelského rozhraní, které jsou nutné k vývoji.
- Funkce 3D grafiky – díky těmto funkcím je umožněno vytvářet 3D tvary, kterými jsou například válec, kvádr nebo koule.
- FXML a nástroj pro tvorbu scén – jedná se o značkovací jazyk založený na XML, nabízí taktéž nástroj JavaFX Scene Builder, který slouží k návrhu grafického rozhraní bez nutnosti psát kód.

[23]

5.5 Git

Git je nejvíce používaný systém pro správu verzí. Sleduje změny, které jsou v souborech provedeny, takže díky Gitu získáváme záznam o tom, co bylo provedeno. Umožňuje taktéž se v případě potřeby vrátit ke konkrétní verzi. Usnadňuje spolupráci a umožňuje sloučit změny provedené více lidmi do jednoho zdroje. Ke Gitu lze přistupovat pomocí příkazové řádky, přes desktopovou aplikaci, která má grafické uživatelské rozhraní nebo přes IDE, které má Git integrovaný v sobě.

Všeobecně je Git software, který běží lokálně, kde soubory a jejich historie jsou uloženy na konkrétním počítači. Lze také využít online hostitele (GitHub či BitBucket) k ukládání kopií souborů a jejich historie. Díky tomuto centrálně umístěnému depozitáři můžeme nahrávat změny, stahovat změny, které provedl někdo jiný, a můžeme tak snadněji spolupracovat s ostatními vývojáři. [17]

5.6 Postman

Postman je platforma API pro vytváření, testování a používání API. Zjednodušuje každý krok životního cyklu API, zefektivňuje spolupráci, což umožňuje vytvářet lepší API rozhraní rychleji.

Klient Postman API je základní nástroj Postmana, který umožňuje snadno prozkoumávat, ladit a testovat API rozhraní a umožňuje definovat složité požadavky API pro HTTP, REST, SOAP či WebSockets. Automaticky detekuje jazyk odpovědi, odkazů a formátovaného textu uvnitř těla odpovědi. Jednoduše lze uspořádat požadavky do složek (sbírek), díky čemuž je lze využít opětovně, aniž bychom ztráceli čas vytvářením všeho od začátku. [24]

6 POPIS VÝSLEDNÉHO ŘEŠENÍ A IMPLEMENTACE

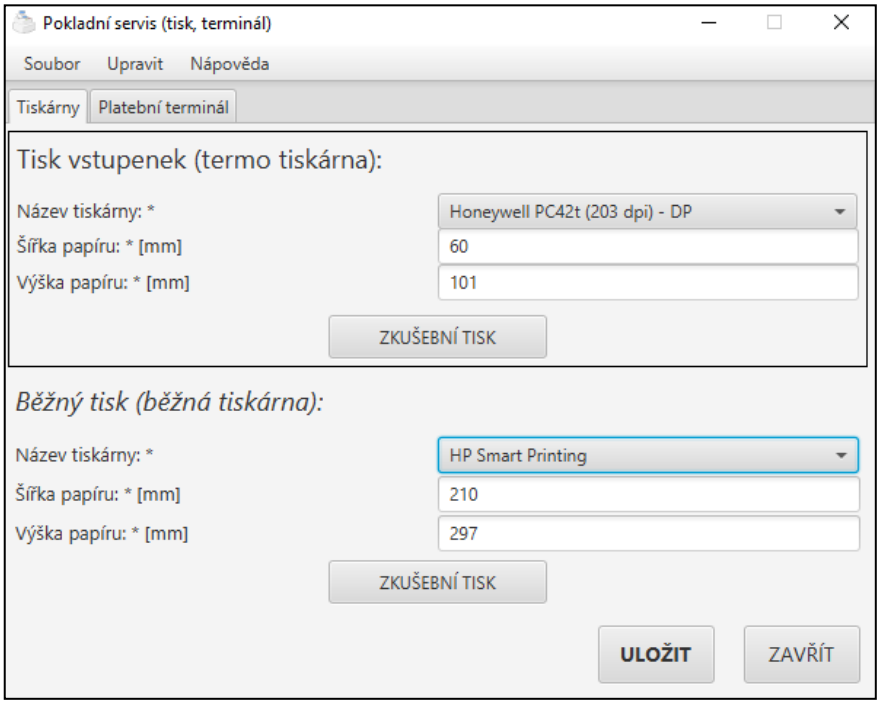
Kapitola je věnována výslednému řešení, popisu implementace, jehož součástí jsou ukázky grafického uživatelského rozhraní, zdrojových kódů a ukázkové požadavky odesílané na vytvořené REST API.

6.1 Popis

Aplikace, která je výsledkem této diplomové práce, slouží ke zprostředkování komunikace webové aplikace a vstupně-výstupních zařízení připojených k hostitelskému počítači a ke konfiguraci připojených zařízení. Aplikace je napsána v jazyce Java s využitím frameworku Spring Boot. Součástí aplikace je grafické uživatelské rozhraní, které je napsáno v jazyce JavaFX. Aplikace přijímá požadavky od webové aplikace prostřednictvím REST API, zpracovává je (zprostředkovává tisk požadovaných dat, či zrealizuje platbu na platebním terminálu), následně zpracuje výsledek tisku či platby a zpracovaný výsledek (uživatelsky čitelný) vrací zpět pokladní aplikaci. Součástí aplikace je uživatelské rozhraní, díky kterému je možné nastavit parametry pro tiskárny a pro platební terminál. Tyto parametry jsou uloženy v lokálním souboru typu JSON. Taktéž je v grafickém uživatelském rozhraní umožněno ověřit funkčnost připojených zařízení.

6.2 Uživatelské rozhraní pro konfiguraci připojených zařízení

Okno uživatelského rozhraní je rozděleno do dvou tabulek. Tabulka pro tiskárny a tabulka pro platební terminál. Dále je zde menu, které nabízí základní operace pro práci s rozhraním.



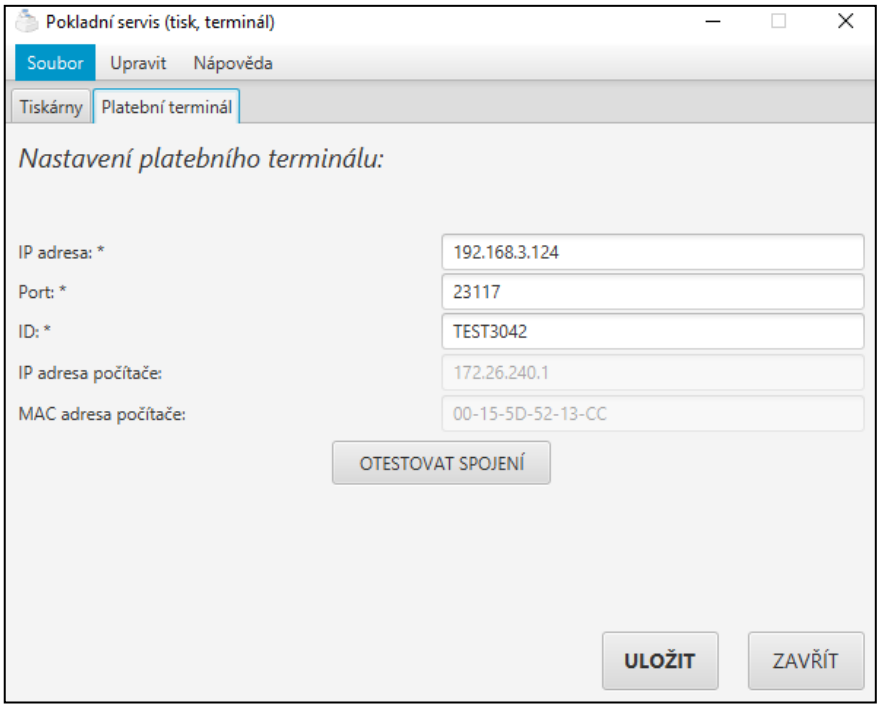
Obrázek 8 - Grafické uživatelské rozhraní část tiskárny

Tabulka pro tiskárny je rozdělena do dvou částí. První část je určena pro termotiskárnu, druhá pro běžnou tiskárnu. Každá část pak obsahuje tři pole a jedno tlačítko.

- **Pole pro výběr názvu tiskárny** – toto pole je plněno názvy tiskáren, které jsou připojeny k aktuálnímu počítači.
- **Pole šírka papíru** – pole pro vyplnění číselné hodnoty představující šířku papíru, na který chceme tisknout, v milimetrech.
- **Pole výška papíru** – pole pro vyplnění číselné hodnoty představující výšku papíru, na který cheme tisknout, v milimetrech.
- **Tlačítko zkušební tisk –** odešle zkušební data na příslušné existující REST API pro vykonání tisku na příslušné tiskárně, pokud tisk proběhne v pořádku, můžeme si díky tomu ověřit funkčnost tiskárny.

Ve spodní části okna jsou dvě tlačítka, uložit a zavřít. Tlačítka mají funkčnost, která vypovídá jejich popisku.

- **Tlačítko uložit** – po stisku tohoto tlačítka proběhne validace hodnot zadaných v polích výše a následně se odešle požadavek se změnou parametrů na existující REST API, po dokončení operace se zobrazí dialogové okno s hláškou o výsledku provedené operace.
- **Tlačítko zavřít** – jednoduše ukončí celou aplikaci, ukončí se grafické uživatelské rozhraní, tak i samotné REST API běžící na pozadí.



Obrázek 9 - Grafické uživatelské rozhraní část platební terminál

Tabulka „Platební terminál“ slouží pro nastavení parametrů platebního terminálu, který je připojen k hostitelskému počítači. Předpokládá se, že terminál bude připojen skrze síť, proto jsou zde definována čtyři pole a jedno tlačítko.

- **Pole IP adresa** – slouží pro zadání ip adresy, na které je připojen platební terminál.
- **Pole port** – slouží pro zadání portu, na kterém příslušný platební terminál naslouchá.
- **Pole ID** – slouží pro zadání jedinečného identifikátoru platebního terminálu, který je terminálu přidělen bankovní společností.
- **Pole IP adresa počítače** – pouze informativní pole, které obsahuje IP adresu hostitelského počítače, slouží pro snadné nalezení tohoto parametru, který zpravidla je nutné zadat do platebního terminálu, aby byl schopen přijímat požadavky.

- **Pole MAC adresa počítače** – taktéž pouze informativní pole, obsahující MAC adresu hostitelského počítače, některé platební terminály vyžadují zadat MAC adresu klienta, od kterého bude přijímat požadavky.

Ve spodní části okna jsou opět dvě tlačítka, uložit a zavřít. Tlačítka mají funkčnost, která vypovídá jejich popisku.

- **Tlačítko uložit** – po stisku tlačítka se validují hodnoty, které byly zadány do polí, odesílá se požadavek se změnou parametrů na existující REST API a po dokončení všech příslušných operací je zobrazeno dialogové okno s hláškou, která popisuje výsledek operace.
- **Tlačítko zavřít** – tlačítko, které ukončí celou aplikaci, ukončí jak REST API, které běží celou dobu na pozadí, tak i grafické uživatelské rozhraní.

6.3 Konfigurační soubory

Jedná se o soubory typu JSON, které jsou uloženy lokálně, ve stejné složce, v níž je umístěna aplikace. Soubory obsahují aktuální parametry potřebné pro komunikaci s vybranými zařízeními. První soubor obsahuje konfiguraci tiskáren a druhý soubor konfiguraci platebního terminálu.

```
{
  "printers": [
    {
      "name": "TestThermo",
      "width": 62,
      "type": "THERMO",
      "height": 102
    },
    {
      "name": "TestOther",
      "width": 210,
      "type": "OTHER",
      "height": 297
    }
  ]
}
```

Ukázka zdrojového kódu 1 - Konfigurační soubor pro tiskárny

Konfigurační soubor pro tiskárny obsahuje jedno pole s názvem „printers“ o dvou objektech. Každý z těchto objektů obsahuje name (název), widht (šířka), type (typ) a height (výška). Parametry název a typ jsou typu řetězec, parametry šířka a výška jsou typu číslo a jedná se o délku v milimetrech.

```
{
  "terminal": {
    "port": 23117,
    "ip": "192.168.3.124",
    "id": "TEST3042",
    "type": "default"
  }
}
```

Ukázka zdrojového kódu 2 - Konfigurační soubor pro platební terminál

Tento soubor reprezentuje konfiguraci platebního terminálu, obsahuje jeden objekt s několika atributy. Mezi tyto atributy patří port, ip, id a type. Port je reprezentován číslem, zbylé parametry jsou řetězcem. Atribut ip je IP adresa, která byla v síti přidělena platebnímu terminálu, port je port, na kterém platebním terminálu naslouchá, id je jedinečný identifikátor platebního terminálu a parametr type je zde pro případ, kdybychom chtěli aplikaci rozšířit pro komunikaci s více typy terminálů, prozatím je vždy vyplněn řetězcem „default“.

6.4 Struktura projektu

Projekt je rozdělen do několika balíčků, které obsahují další balíčky. Balíčky jsou pojmenovány na základě toho, k čemu slouží třídy uvnitř nich. Dva základní větší balíčky jsou balíček „**api**“ a balíček „**gui**“. Balíček „**api**“ obsahuje další balíčky, implementující třídy týkající se realizace REST API a dalších souvisejících operací. Balíček „**gui**“ pak obsahuje třídy obsluhující grafické uživatelské rozhraní.

6.4.1 Balíček api

Obsahuje dva balíčky, **printer_service** (týkající se tiskárny) a **terminal_service** (týkající se platebního terminálu). Dále obsahuje balíček **error**, ve kterém jsou nadefinované předpisy pro výjimky. Balíčky **printer_service** a **terminal_service** obsahují vždy další menší balíčky, jejích názvy a funkčnost jsou stejné, jsou to tyto:

- **controller** – balíček zapouzdřující kontroléry, které představují zprostředkování komunikace mezi webovou aplikací a samotnou logikou aplikace.
- **dto** – obsahuje třídy, které slouží jako předpis pro objekty přenášené prostřednictvím REST API.
- **entity** – obsahuje třídy, které jsou předpisem pro uložené konfigurace.
- **enums** – v tomto balíčku nalezneme výčtové typy, které jsou v dané části potřeba.

- **service** – součástí tohoto balíčku jsou třídy realizující samotné operace potřebné pro vykonání komunikace s danými zařízeními.
- **utils –** tento balíček zapouzdřuje třídy, které slouží jako pomocné pro realizaci potřebných operací.

6.4.2 Balíček gui

Balíček obsahuje třídy obsluhující grafické uživatelské rozhraní a další menší balíček s názvem „**api**“, jehož součástí jsou třídy zprostředkovávající komunikaci s REST API.

6.5 Důležité funkce a jejich implementace

Podkapitola obsahuje nejdůležitější a nejzajímavější metody, které jsou součástí implementace aplikace. Implementace propojení s platebním terminálem podléhá interním pravidlům bankovní společnosti, v odevzdaných zdrojových kódech jsou všechny třídy a příslušné metody, u metod obsluhujících platební terminál jsou zachovány pouze hlavičky metod naznačující princip fungování.

6.5.1 Využité knihovny a balíčky

Ke správnému fungování bylo nutné využít dvě knihovny, které nejsou standardně součástí jazyku Java, ale lze je snadno doinstalovat.

6.5.1.1 Balíček javax.print

Umožnuje klientským i serverovým aplikacím:

- Najít a vybrat tiskové služby na základě jejich vlastností
- Definovat formát tiskových dat
- Odesílat tiskové úlohy službám, které podporují daný typ dokumentu

Součástí balíčku je implementace třídy **PrintServiceLookup**, která slouží k nalezení tiskových platforem. Tato třída byla v této práci využita pro vyhledání všech dostupných tiskáren a vyhledání konkrétní tiskárny na základě názvu. Třída **PrintService** představuje službu konkrétní tiskárny, jejíž pomocí můžeme vytvořit tiskovou úlohu. **PrintRequestAttributeSet** reprezentuje seznam atributů, které můžeme tiskové nastavit, případně pomocí nich můžeme vyhledat tiskové služby. **DocPrintJob** je v podstatě vytvořená tisková úloha, která zajišťuje tisk, můžeme jí nastavit potřebné parametry a odchytit výsledek tisku. Další implementovanou

třídou v tomto balíčku je třída **Doc**, jež představuje konkrétní dokument, který se bude odesílat k tisku. [25]

6.5.1.2 openhtmltopdf

Je knihovnou určenou pro vykreslování podmnožiny správně formátovaného XML, XHTML či HTML (součástí může být stylování za pomocí CSS), výstupem pak je buď PDF, nebo obrázek. Tato knihovna byla využita v diplomové práci k převodu zaslaných dat k tisku ve formátu HTML na obrázek. Tato knihovna je distribuována pod licencí GNU Lesser General Public License (verze 2.1 nebo novější).

Konkrétně byly využity tyto třídy: **SVGDrawer** (slouží jako parametry třídy Java2DRenderer a vytváření obrázku), **Java2DRendererBuilder** (třída realizující samotný převod HTML na obrázek), **BufferedImagePageProcessor** (pomocná třída, do které se ukládá výsledek převodu HTML na obrázek). [8]

6.5.2 Zprostředkování tisku

Požadavek na tisk je přijat prostřednictvím REST API. Vyvolá se metoda print, která přijímá jako parametr objekt typu PrintDto obsahující data pro tisk a typ tiskárny, na níž se má tisknout. Návratovým typem této metody je boolean, představující logickou hodnotu true nebo false. Hodnota true je vrácena v případě, kdy vše proběhlo v pořádku. V metodě mohou být vyvolány výjimky v případě, že není tiskárna dostupná, případně když tisk neproběhl v pořádku nebo když nebyl nalezen konfigurační soubor.

Metoda začíná vyhledáním instance typu Printer na základě požadovaného typu, metoda getPrinter vyhledává v konfiguračním souboru pro tiskárny příslušné parametry pro žádanou tiskárnu. Pokračuje vyhledáním instance typu PrintService. PrintService je třída z balíčku javax.print poskytující komunikaci s tiskárnami připojenými k počítači. Následně po vytvoření instance třídy Html2ImageConvertor (třída pro převádění HTML na obrázek) se za pomoci cyklu projdou všechna přijatá data, která se postupně převedou na obrázek a odešlou k tisku. V této metodě může být vyvoláno několik výjimek NoPrinterException (není nastavena příslušná tiskárna), PrintFailedException (nepodařilo se vytisknout požadovaná data), FileNotFoundException (nebyl nalezen konfigurační soubor) a DataConvertException (chyba při konvertování dat). Tyto výjimky pak jsou odchyceny v třídě PrinterController.

```
public boolean print(PrintDto printDto) throws NoPrinterException,
PrintFailedException, FileNotFoundException, DataConvertException {
    //Get printer and them printservice
    printer =
getPrinter(PrinterType.getTypeFromString(printDto.getType()));
    service = getPrintService(printer);

    //Get convertor instance
    Html2ImgConvertor html2ImgConvertor = new Html2ImgConvertor();
    //foreach all data, convert them and print
    for (String data : printDto.getData()) {
        FileInputStream fis = null;
        fis = html2ImgConvertor.convert(data, printer.getPageHeight(),
printer.getPageWidth());
        printImage(fis);

    }
    return true;
}
```

Ukázka zdrojového kódu 3 - Metoda pro tisk dat přijatých prostřednictvím REST API

6.5.3 Převod HTML na obrázek

Jedná se o pomocnou metodu, jež slouží pro převod HTML dat na obrázek, který je následně možné použít jako data určená k tisku na požadované tiskárně.

Návratovým typem metody je FileInputStrem. V případě, že je vše v pořádku, vrací FileInputStrem, ve kterém je otevřený převedený obrázek, v případě chyby může být vyvoláno několik výjimek: IOException (problém s čtením nebo zápisem), SAXException (výjimka vyvolaná v případě, kdy HTML data nejsou validní), IllegalArgumentException (číselné parametry nejsou validní). Metoda využívá třídy z knihovny openhtmltopdf a princip celé metody vychází z dokumentace této knihovny. Třída Java2DRendererBuilder slouží ke generování obrázku z HTML dat. Postupně instanci nastavíme potřebné parametry, jako jsou samotná HTML data, cesta k souboru, ve kterém bude uložen obrázek, šířka a výška obrázku. Třída BufferedImagePageProcessor je třídou pomocnou, do jejíž instance jsou převedena data na obrázek, v konstruktoru této třídy nastavujeme typ a kvalitu výsledného obrázku. Závěrem je za pomoci builderu vybrána jedna stránka z HTML dat a převedena na obrázek, který je uložen na zvolenou cestu. Po ukončení všech těchto kroků se vrací otevřený proud dat souboru, do kterého byl uložený vytvořený obrázek.

```
public FileInputStream convert(String html, Long height, Long width)
throws DataConvertException {

    try (SVGDrawer svg = new BatikSVGDrawer();
         SVGDrawer mathMl = new MathMLDrawer()) {

        Java2DRendererBuilder builder = new Java2DRendererBuilder();
        builder.useSVGDrawer(svg);
        builder.useMathMLDrawer(mathMl);
        builder.addDOMMutator(LaTeXDOMMutator.INSTANCE);
        builder.useObjectDrawerFactory(buildObjectDrawerFactory());
        builder.withHtmlContent(html, IMG_PATH);

        BufferedImagePageProcessor bufferedImagePageProcessor = new
BufferedImagePageProcessor(
                BufferedImage.TYPE_INT_RGB, 3.0);

        builder.useDefaultPageSize(width, height,
Java2DRendererBuilder.PageSizeUnits.MM);
        builder.useEnvironmentFonts(true);

        //Render single page image

builder.toSinglePage(bufferedImagePageProcessor).runFirstPage();
        BufferedImage image =
bufferedImagePageProcessor.getPageImages().get(0);

        ImageIO.write(image, "PNG", new File(IMG_PATH));
        return new FileInputStream(IMG_PATH);

    }catch(IllegalArgumentException ex){
        throw new DataConvertException("INVALID_PAGE_SIZE");
    }catch (SAXException saex){
        throw new DataConvertException("INVALID_HTML_DATA");
    } catch (IOException ioex){
        throw new DataConvertException("DATA_CONVERT_ERROR");
    }
}
```

Ukázka zdrojového kódu 4 - Metoda pro převod html na obrázek

6.5.4 Odeslání obrázku k tisku

Metoda využívá tříd z balíčku javax.print (DocPrintJob, Doc, PrintRequestAttributeSet). Metoda přijme v parametru otevřený proud dat s vytvořeným obrázkem z HTML dat určených k tisku. Na základě nalezené služby pro tisk příslušné tiskárny se vytvoří úloha tisku. Poté se vytvoří dokument pro tisk (s otevřeným proudem dat vytvořeného obrázku a typem obrázku) a atributy pro tisk (počet kopií, velikost stránky). Následně se ve vytvořené úloze vyvolá metoda tisku, která přijímá jako parametry vytvořený dokument a potřebné atributy. V případě selhání této metody se vyvolá výjimka PrintException, která je odchycena a je vyvolána nová výjimka popisující konkrétní chybu, jež nastala.

```
private void printImage(InputStream fis) throws PrintFailedException {
    DocPrintJob job = service.createPrintJob();
    Doc doc = new SimpleDoc(fis, DocFlavor.INPUT_STREAM.PNG, null);
    PrintRequestAttributeSet attrib = new
HashPrintRequestAttributeSet();
    attrib.add(new Copies(1));
    attrib.add(new MediaPrintableArea(0, 0, printer.getPageWidth(),
printer.getPageHeight(), Size2DSyntax.MM));
    try {
        job.print(doc, attrib);
    } catch (PrintException e) {
        throw new PrintFailedException("PRINT_FAILED");
    }
}
```

Ukázka zdrojového kódu 5 - Metoda printImage sloužící pro tisk obrázku

6.6 Automatické testování

Automatické testování představuje techniku, která slouží k testování softwaru. Toto testování je prováděno pomocí speciální softwarových nástrojů. [15]

6.6.1 Jednotkové testy

Jednotkové testy je metoda testování jednotlivých částí kódu, jako jsou funkce či postupy, které jsou izolované od zbytku systému. Toto testování má zajistit, aby každá jednotka kódu fungovala tak, jak bylo zamýšleno, a identifikovat případné chyby v rané fázi vývojového procesu. [15]

6.6.1.1 Mockito

Mockito je mockovací framework založený na jazyku Java. Využívá se pro efektivní jednotkové testy. Napomáhá simulovat rozhraní, kterému lze přidat fiktivní funkce. Nabízí nám způsob, jak otestovat funkčnost třídy izolovaně, nevyžaduje připojení k databázi či přístup ke čtení souborů. Jedná se v podstatě o falšování objektů, které nahrazují skutečné služby, jimž můžeme předat fiktivní vstup. Zfalšovaný objekt pak vrací zfalšovaná data, která představují výsledek, jenž může nastat. V této práci byl tento framework využit pro testování controllerů. [26]

6.6.2 Integrační testy

Integrační testy slouží pro testování toho, jak různé jednotky spolu komunikují nebo spolupracují. Jejich cílem je zajistit, aby se integrovaný systém choval dle očekávání a nedocházelo ke konfliktům nebo chybám mezi různými jednotkami kódu. [15]

6.6.3 Implementované testy

Aplikace byla otestována několika testy, jak integračními, tak jednotkovými. Zde přikládám pár ukázek pro testování logiky týkající se tiskáren. Testy byly vytvořeny pro třídy Controllerů, servisní třídy a pro pomocnou třídu Html2ImageConvertor.

6.6.3.1 Jednotkové testy

Následující ukázka obsahuje testy, které ověřují funkčnost pomocné třídy sloužící k převodu dat ve formátu HTML na obrázek. Zkouší různé případy, které by mohly nastat.

```
@Test
public void testConvert_Success() {
    String htmlData = "<html><meta charset=\"UTF-8\"/><body><div>TEST
TISKU</div></body></html>";
    Html2ImgConvertor convertor = new Html2ImgConvertor();
    Assertions.assertDoesNotThrow(() -> {
        FileInputStream fis = convertor.convert(htmlData, 101L, 101L);
        Assertions.assertNotNull(fis);
    });
}

@Test
public void testConvert_InvalidLength() {
    String htmlData = "<html><meta charset=\"UTF-8\"/><body><div>TEST
TISKU</div></body></html>";
    Html2ImgConvertor convertor = new Html2ImgConvertor();
    Assertions.assertThrows(DataConvertException.class, () -> {
        convertor.convert(htmlData, 101L, -101L);
    },"INVALID_PAGE_SIZE");
}

@Test
public void testConvert_InvalidHeight() {
    String htmlData = "<html><meta charset=\"UTF-8\"/><body><div>TEST
TISKU</div></body></html>";
    Html2ImgConvertor convertor = new Html2ImgConvertor();
    Assertions.assertThrows(DataConvertException.class, () -> {
        convertor.convert(htmlData, 0L, 101L);
    }, "INVALID_PAGE_SIZE");
}

@Test
public void testConvert_InvalidData() {
    Html2ImgConvertor convertor = new Html2ImgConvertor();
    Assertions.assertThrows(DataConvertException.class, () -> {
        convertor.convert("", 101L, 101L);
    }, "INVALID_HTML_DATA");
}
```

Ukázka zdrojového kódu 6 - Testování třídy HTML2ImgConverter

Další ukázka, týkající se třídy PrinterService, provádí testování příslušných metod a správnosti jejich chování.

```
@Test
public void testPrint_PrintFaildException() {
    PrintDto printDto = new PrintDto();
    printDto.setType("THERMO");
    List<String> data = new ArrayList<>();
    data.add("<html><body><h1>Hello World!</h1></body></html>");
    printDto.setData(data);

    Assertions.assertThrows(PrintFailedException.class,()->{
        boolean result = printerService.print(printDto);
    },"PRINTER_SHUT_DOWN");
}

@Test
public void testPrint_NoPrinterException() {
    PrintDto printDto = new PrintDto();
    printDto.setType("INVALID_PRINTER_TYPE");
    List<String> data = new ArrayList<>();
    data.add("<html><body><h1>Hello World!</h1></body></html>");
    printDto.setData(data);

    assertThrows(NoPrinterException.class, () ->
printerService.print(printDto));
}

@Test
public void testPrint_PrintFailedException() {
    PrintDto printDto = new PrintDto();
    printDto.setType("THERMO");
    List<String> data = new ArrayList<>();
    data.add("<html><body><h1>Hello World!</h1></body></html>");
    printDto.setData(data);

    assertThrows(PrintFailedException.class, () ->
printerService.print(printDto));
}

@Test
public void testCheckPrinters() {
    List<StatusDto> results = printerService.checkPrinters();
    assertNotNull(results);
    assertFalse(results.isEmpty());
}
```

Ukázka zdrojového kódu 7 - Testování třídy PrinterService

6.6.3.2 Integrační testy

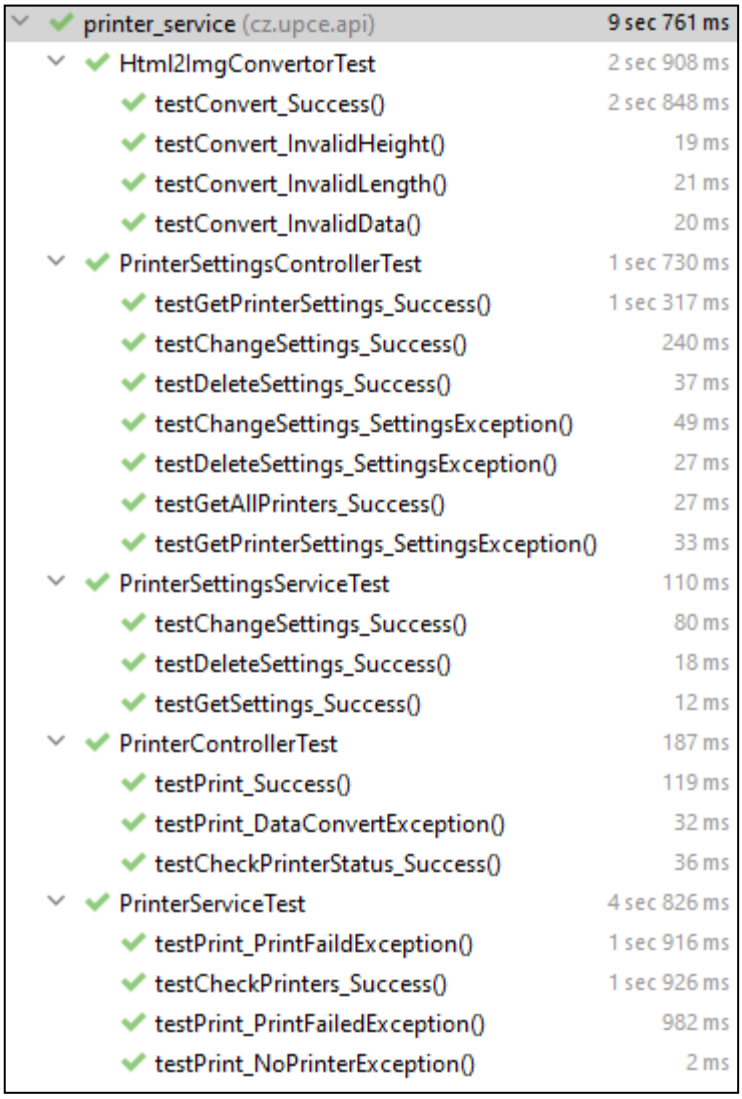
Poslední ukázka je věnována testování třídy PrinterController. V těle testu je za pomoci frameworku Mockito podvrženo chování dané metody a následně odeslán požadavek i vyhodnocen výsledek. Tímto způsobem je otestováno, zda se Controller chová správně a adekvátně reaguje na požadavky.

```
@Test
public void testPrint_Success() throws Exception {
    PrintDto printDto = new PrintDto();
    printDto.setType("THERMO");
    List<String> data = new ArrayList<>();
    data.add("<html><body><p>TEST</p></body></html>");
    printDto.setData(data);
    when(printerService.print(any(PrintDto.class))).thenReturn(true);
    mockMvc.perform(post("/api/v1/print")
                    .contentType(MediaType.APPLICATION_JSON)
                    .content(asJsonString(printDto)))
            .andExpect(status().isOk())
            .andExpect(content().string("true"));
}

@Test
public void testPrint_DataConvertException() throws Exception {
    PrintDto printDto = new PrintDto();
    printDto.setType("THERMO");
    List<String> data = new ArrayList<>();
    data.add("");
    printDto.setData(data);
    when(printerService.print(any(PrintDto.class))).thenThrow(new
DataConvertException("INVALID_HTML_DATA"));
    mockMvc.perform(post("/api/v1/print")
                    .contentType(MediaType.APPLICATION_JSON)
                    .content(asJsonString(printDto)))
            .andExpect(status().isBadRequest())
            .andExpect(content().string("INVALID_HTML_DATA"));
}

@Test
public void testCheckPrinterStatus_Success() throws Exception {
    StatusDto status1 = new StatusDto();
    status1.setType("THERMO");
    status1.setStatus("SHUT_DOWN");
    // Nastavit hodnoty status1
    StatusDto status2 = new StatusDto();
    status2.setType("OTHER");
    status2.setStatus("NOT_SET_UP");
    // Nastavit hodnoty status2
    List<StatusDto> statusList = new ArrayList<>();
    statusList.add(status1);
    statusList.add(status2);
    when(printerService.checkPrinters()).thenReturn(statusList);
    mockMvc.perform(get("/api/v1/print/check"))
            .andExpect(status().isOk())

.andExpect(content().json("[{\"type\":\"THERMO\",\"status\":
\"SHUT_DOWN\"}, {\"type\":\"OTHER\", \"status\": \"NOT_SET_UP\"}]"));
}
```

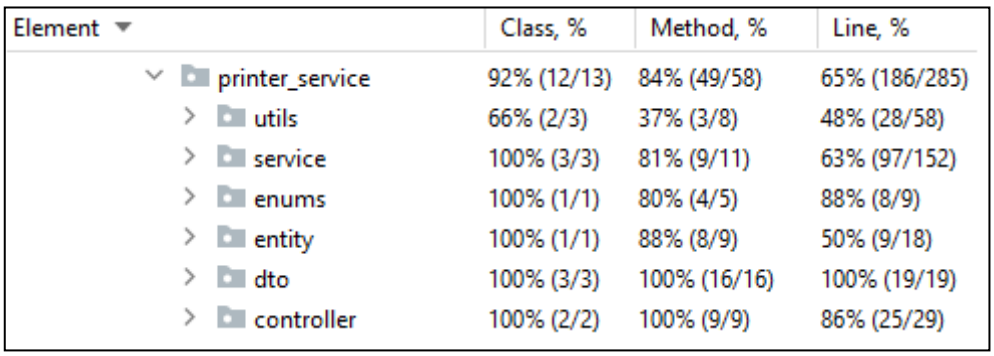
Ukázka zdrojového kódu 8 - Testování třídy PrinterController

6.6.3.3 Vyhodnocení testů

Využité vývojové prostředí nabízí grafické znázornění výsledku všech testů a celkové pokrytí daného balíčku testy. Všechny testy byly úspěšné a využité vývojové prostředí zobrazuje celkové pokrytí balíčku printer_service. Pro třídy je 92% pokrytí testy, metody jsou pokryty na 84 % a pro celkový počet řádků je pokrytí 65 %.



Obrázek 10 - Vyhodnocení implementovaných testů



Element	Class, %	Method, %	Line, %
printer_service	92% (12/13)	84% (49/58)	65% (186/285)
utils	66% (2/3)	37% (3/8)	48% (28/58)
service	100% (3/3)	81% (9/11)	63% (97/152)
enums	100% (1/1)	80% (4/5)	88% (8/9)
entity	100% (1/1)	88% (8/9)	50% (9/18)
dto	100% (3/3)	100% (16/16)	100% (19/19)
controller	100% (2/2)	100% (9/9)	86% (25/29)

Obrázek 11 - Celkové pokrytí balíčku printer_service testy

6.7 Testování REST API za pomoci nástroje Postman

Za pomoci nástroje Postman jsem manuálně otestoval funkčnost vytvořeného REST API. Níže je uvedeno pár ukázek. Ukázky se týkají pouze operací tiskárnami, protože data, která prochází při zprostředkování platby na platebním terminálu podléhají interním pravidlům bankovní společnosti a nemohou zde být uvedena.

6.7.1 Změna nastavení

Požadavek se odesílá na adresu: http://localhost:8082/api/v1/print/settings, jedná se o požadavek typu POST. Požadavky probíhají obdobně i při změně nastavení platebního terminálu, jen se odesílají na adresu http://localhost:8082/api/v1/terminal/settings. Proto zde přikládám ukázku pouze pro jeden z případů. Jsou zde uvedeny dva příklady, jeden, který proběhne korektně, druhý v případě, kdy zadáme jeden z parametrů špatně.

Příklad 1 (korektní nastavení):

```
[
  {
    "name": "TestThermo",
    "pageHeight": 101,
    "pageWidth": 60,
    "type": "THERMO"
  },
  {
    "name": "TestOther",
    "pageHeight": 297,
    "pageWidth": 210,
    "type": "OTHER"
  }
]
```

Ukázka zdrojového kódu 9 - Tělo požadavku na nastavení tiskáren – v pořádku

Odesláním tohoto požadavku aplikace provede změnu nastavení a v odpovědi bychom měli dostat nové nastavení (tudíž vlastně to stejné, co jsme odeslali).

```
[
  {
    "name": "TestOther",
    "pageHeight": 297,
    "pageWidth": 210,
    "type": "OTHER"
  },
  {
    "name": "TestThermo",
    "pageHeight": 101,
    "pageWidth": 60,
    "type": "THERMO"
  }
]
```

Ukázka zdrojového kódu 10 - Tělo odpovědi na požadavek změny nastavení tiskáren – v pořádku

Server odpověděl s kódem 200 (OK) a v těle odpovědi je nové nastavení, vše proběhlo v pořádku.

Příklad 2 (zadán špatný typ tiskárny):

```
[
  {
    "name": "TestThermo",
    "pageHeight": 101,
    "pageWidth": 60,
    "type": "IDONTKNOW"
  },
  {
    "name": "TestOther",
    "pageHeight": 297,
    "pageWidth": 210,
    "type": "OTHER"
  }
]
```

Ukázka zdrojového kódu 11 - Tělo požadavku na změnu nastavení tiskáren – špatný typ tiskárny

Jako odpověď na tento požadavek se očekává nějaká chyba, či jiné nastavení, než jsme vyžadovali.

```
[
  {
    "name": "TestThermo",
    "pageHeight": 101,
    "pageWidth": 60,
    "type": "UNKOWN"
  },
  {
    "name": "TestOther",
    "pageHeight": 297,
    "pageWidth": 210,
    "type": "OTHER"
  }
]
```

Ukázka zdrojového kódu 12 - Tělo odpovědi na požadavek změny nastavení tiskáren – špatný typ tiskárny

Server opět odpověděl s kódem 200 (OK) a v těle odpovědi je nové nastavení. Server provedl nastavení, ale námi požadovaný typ tiskárny není korektní, proto námi požadované tiskárně nastavil type na UNKOWN, což sice pro tento případ nijak nevadí, ale až budeme požadovat tisk, tak neproběhne z důvodu neznámého typu tiskárny.

6.7.2 Tisk

Požadavek se odesílá na adresu: http://localhost:8082/api/v1/print, jedná se o požadavek typu POST.

Příklad 1 (tisk v pořádku):

```
{
  "data": [
    "<html><meta charset=\"UTF-8\"/><body><div>TEST
TISKU</div></body></html>"
  ],
  "type": "OTHER"
}
```

Ukázka zdrojového kódu 13 - Tělo požadavku pro tisk – v pořádku

Po odeslání takového požadavku server odpovídá s kódem 200 (OK) a v těle odpovědi je true, což znamená, že data byla úspěšně odeslána k tisku.

Příklad 2 (neznámý typ tiskárny)**:**

```
{
  "data": [
    "<html><meta charset=\"UTF-8\"/><body><div>TEST
TISKU</div></body></html>"
  ],
  "type": "IDONTKNOW"
}
```

Ukázka zdrojového kódu 14 - Tělo požadavku pro tisk – neznámý typ tiskárny

V takovém případě odpovídá server s kódem 400 (Bad Request) a v těle je hláška „UNKOWN_TYPE_OF_PRINTER“. Tato odpověď vypovídá o tom, že jsme zadali špatný typ tiskárny a tisk tak nemohl být zprostředkován.

Příklad 3 (nenastavena žádná tiskárna):

```
{
  "data": [
    "<html><meta charset=\"UTF-8\"/><body><div>TEST
TISKU</div></body></html>"
  ],
  "type": "OTHER"
}
```

Ukázka zdrojového kódu 15 - Tělo požadavku na tisk – nenastavena žádná tiskárna

Na takový požadavek server odesílá odpověď s kódem 400 (Bad Request) a tělo obsahuje hlášku „NO_PRINTER_SET“. Odpověď značí to, že není nastavena žádná tiskárna, tisk tedy nebyl zprostředkován.

ZÁVĚR

Cílem diplomové práce byl vznik aplikace, která bude umožňovat webové aplikaci komunikovat se vstupně-výstupními zařízeními skrze nově vytvořené REST API a zároveň bude umožňovat konfiguraci potřebných zařízení prostřednictvím grafického uživatelského rozhraní. K vývoji této aplikace byl využit jazyk Java s pomocí frameworku Spring Boot a pro grafické uživatelské rozhraní JavaFX. Diplomová práce se taktéž věnuje analýze současného řešení a analýze nového řešení.

První kapitola obsahuje základní popis webové aplikace, toho, jak funguje, jak se dělí a k čemu vlastně slouží. Jsou zde popsány pojmy, které s webovou aplikací úzce souvisí.

Druhá kapitola se věnuje vstupně-výstupním zařízením a tomu, co vlastně tato zařízení jsou. Jsou zde zmíněny nejběžnější způsoby připojení takových zařízení k počítači. V závěru této kapitoly jsou popsána zařízení, pro které bylo napsáno řešení, jež je předmětem této diplomové práce.

Třetí kapitola popisuje současné řešení, je zde zjednodušený popis fungování v současném stavu. Jsou zde vyjmenovány nevýhody současného řešení a důsledky, které nastávají v současném řešení.

Čtvrtá kapitola je věnována analýze nového řešení. Je zde sepsán základní popis a představa toho, jak by nové řešení mělo fungovat. Dále jsou zde definovány funkční a nefunkční požadavky. Na základě požadavků byl zpracován model případů užití. Pro dané případy užití byly sepsány scénáře, tedy to, jak bude řešení fungovat v ideálním stavu, případně také v situaci, kdy nastane chyba. Na základě funkčních požadavků a případů užití byla sestavena matice sledovatelnosti, díky které bylo ověřeno, zda jsou definovány všechny požadavky a zda jsou k nim vytvořeny příslušné případy užití. Na základě všech předchozích kroků byl vytvořen analytický model neboli principiální model fungování systému.

Pátá kapitola obsahuje popis využitých nástrojů a technologií, které byly využity při psaní této diplomové práce. Nástroje pro analýzu, programování, verzování, tak i testování.

Poslední, šestá kapitola se věnuje výslednému řešení, je zde popis řešení, popis uživatelského rozhraní, konfiguračních souborů, je zde vysvětleno, jak funguje vytvořené REST API, přibližuje se zde základní struktura projektu, popis důležitých funkcionalit v programu a v neposlední řadě testování samotného REST API.

Téma diplomové práce jsem zvolil kvůli problémům, které nastávaly ve firmě, v níž pracuji. Mým cílem bylo zcela tyto problémy vyřešit, nebo alespoň omezit jejich výskyt. Při psaní této práce jsem si důsledně nastudoval komunikaci počítače se vstupně-výstupními zařízeními, prohloubil jsem svou znalost v oblasti analýzy a naučil jsem se ověřit funkčnost vytvořeného REST API bez využití jakékoliv frontendové aplikace. Věřím, že tato práce ulehčí život spoustě lidí, kteří se setkávají s podobnými problémy, a v neposlední řadě že bude přínosem pro mě samotného do budoucna.

POUŽITÁ LITERATURA

[1] Bluetooth (INFORMACE): verze, dosah, frekvence a protokoly. ALZA.CZ [online]. 5. října 2022 [cit. 2023-04-27]. Dostupné z: https://www.alza.cz/slovnik/bluetooth-art12370.htm

[2] Co je HDMI a jaké jsou jeho verze?. COMFOR [online]. 14. 03. 2019 [cit. 2023-04-27]. Dostupné z: https://www.comfor.cz/blog/co-je-hmdi-a-jake-jsou-jeho-verze

[3] ENTERPRISE ARCHITECT: create | verify | share. SPARX SYSTEMS [online]. 2-Mar-2023 [cit. 2023-04-27]. Dostupné z: https://sparxsystems.com/products/ea/

[4] LEMONAKI, Dionysia. Frontend VS Backend – What's the Difference?. FreeCodeCamp [online]. MARCH 18, 2022 [cit. 2023-04-27]. Dostupné z: https://www.freecodecamp.org/news/frontend-vs-backend-whats-the-difference/

[5] GUPTA, Lokesh. HTTP Status Codes. REST API Tutorial [online]. December 17, 2021 [cit. 2023-05-08]. Dostupné z: https://restfulapi.net/http-status-codes/

[6] IT-slovník. IT SLOVNÍK [online]. [cit. 2023-04-27]. Dostupné z: https://it-slovnik.cz/

[7] Návrh webového rozhraní RESTful API. Microsoft [online]. 29. 03. 2023 [cit. 2023-04-27]. Dostupné z: https://learn.microsoft.com/cs-cz/azure/architecture/best-practices/api-design

[8] OPEN HTML TO PDF. GitHub [online]. Apr 1, 2022 [cit. 2023-04-27]. Dostupné z: https://github.com/danfickle/openhtmltopdf

[9] Tiskárna. TechLib [online]. [cit. 2023-04-27]. Dostupné z: https://tech-lib.eu/definition/printer.html

[10] ATAMAN, Altay. Ultimate Guide to Integration Testing vs. Unit Testing in 2023 [online]. JANUARY 24, 2023 [cit. 2023-04-27]. Dostupné z: https://research.aimultiple.com/integration-testing-vs-unit-testing/

[11] USB - JAK SE V TOM VYZNAT?. AV HiFi [online]. [cit. 2023-04-27]. Dostupné z: https://www.avhifi.cz/blog/30-usb-jak-se-v-tom-vyznat

[12] Vše, co potřebujete vědět o POS terminálech. Průvodce podnikáním [online]. 1. 12. 2020 [cit. 2023-04-27]. Dostupné z: https://www.pruvodcepodnikanim.cz/clanek/vse-co-potrebujete-vedet-o-pos-terminalech/

[13] CONTRIBUTOR, TechTarget. Web application (web app). TechTarget [online]. 2023 [cit. 2023-04-27]. Dostupné z: https://www.techtarget.com/searchsoftwarequality/definition/Web-application-Web-app

[14] What is a REST API?. IBM [online]. [cit. 2023-04-27]. Dostupné z: https://www.ibm.com/topics/rest-apis

[15] HAMILTON, Thomas. What is Automation Testing? Test Tutorial. Guru99 [online]. May 1, 2023 [cit. 2023-04-27]. Dostupné z: https://www.guru99.com/automation-testing.html

[16] What Is Computer Networking?. AWS [online]. [cit. 2023-04-27]. Dostupné z: https://aws.amazon.com/what-is/computer-networking/

[17] What is Git and Why Should You Use it?. Noble desktop [online]. December 19, 2022 [cit. 2023-04-27]. Dostupné z: https://www.nobledesktop.com/learn/git/what-is-git

[18] TURNQUIST, Greg L. Learning Spring Boot 2.0 - Second Edition: Simplify the development of lightning fast applications based on microservices and reactive programming. Packt Publishing. 2017. ISBN 978-1786463784.

[19] ARLOW, Jim a Ila NEUSTADT. UML 2 a unifikovaný proces vývoje aplikací: objektově orientovaná analýza a návrh prakticky. 2., aktualiz. a dopl. vyd. Brno: Computer Press, 2007. ISBN 978-80-251-1503-9.

[20] Co je Java?: Příručka pro začátečníky v jazyce Java. *Microsoft Azure* [online]. [cit. 2023-04-27]. Dostupné z: https://azure.microsoft.com/cs-cz/resources/cloud-computing-dictionary/what-is-java-programming-language/

[21] Termotiskárny – jak fungují a jaké jsou jejich výhody. *Originálnítonery.cz* [online]. [cit. 2023-04-27]. Dostupné z: https://www.originalnitonery.cz/blog/termotiskarny-jak-funguji-a-jake-jsou-jejich-vyhody

[22] Co je Java Spring Boot?: Úvod k modulu Spring Boot. *Microsoft Azure* [online]. [cit. 2023-04-27]. Dostupné z: https://azure.microsoft.com/cs-cz/resources/cloud-computing-dictionary/what-is-java-spring-boot/

[23] *JavaFX: Getting Started with JavaFX* [online]. [cit. 2023-04-27]. Dostupné z: https://docs.oracle.com/javase/8/javafx/get-started-tutorial/jfx-overview.htm#JFXST784

[24] What is Postman?. *Postman* [online]. [cit. 2023-04-27]. Dostupné z: https://www.postman.com/product/what-is-postman/

[25] Package javax.print. *Oracle* [online]. [cit. 2023-04-27]. Dostupné z: https://docs.oracle.com/javase/8/docs/api/index.html?javax/print/package-summary.html

[26] Mockito - Overview [online]. [cit. 2023-05-14]. Dostupné z: https://www.tutorialspoint.com/mockito/mockito_overview.htm