

**SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE  
FAKULTA ELEKTROTECHNIKY A INFORMATIKY**

Evidenčné číslo: FEI-5382-97773

**VYUŽITIE ŠTANDARDU WEB COMPONENTS PRE  
NÁVRH WEBOVEJ APLIKÁCIE  
BAKALÁRSKA PRÁCA**

**Bratislava 2021**

**Dominik Fullajtár**

**SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE  
FAKULTA ELEKTROTECHNIKY A INFORMATIKY**

Evidenčné číslo: FEI-5382-97773

**VYUŽITIE ŠTANDARDU WEB COMPONENTS PRE  
NÁVRH WEBOVEJ APLIKÁCIE  
BAKALÁRSKA PRÁCA**

Študijný program:	Aplikovaná informatika
Číslo študijného odboru:	2551
Študijný odbor:	9.2.9 Aplikovaná informatika
Školiace pracovisko:	Ústav automobilovej mechatroniky
Vedúci záverečnej práce:	doc. Ing. Oto Haffner, PhD.
Konzultant:	Ing. Erich Stark, PhD.

**Bratislava 2021**

**Dominik Fullajtár**



## ZADANIE BAKALÁRSKEJ PRÁCE

Študent: **Dominik Fullajtár**  
ID študenta: **97773**  
Študijný program: **aplikovaná informatika**  
Študijný odbor: **informatika**  
Vedúci práce: **Ing. Oto Haffner, PhD.**  
Konzultant: **Ing. Erich Stark, PhD.**  
Miesto vypracovania: **Ústav automobilovej mechatroniky**

Názov práce: **Využitie štandardu Web Components pre návrh webovej aplikácie**

Jazyk, v ktorom sa práca vypracuje: **slovenský jazyk**

### Špecifikácia zadania:

Cieľom práce je navrhnuť a implementovať praktické riešenie s použitím webových technológií, ktoré bude využívať štandard web components. Tento štandard umožňuje tvorbu zapuzdrených komponentov využívajúce API prehliadača.

Úlohy:

1. Naštudujte problematiku Web Components (Custom Elements, Shadow DOM, ES Modules, HTML Template).
2. Vytvorte prehľad existujúcich knižníc pre tvorbu web componentov a ich rozdiely.
3. Vyberte vhodný framework pre realizáciu praktickej časti.
4. Navrhните a implementujte webové riešenie, kde budú tieto komponenty v rámci stránky komunikovať so serverovou časťou cez REST API.
5. Spracujte užívateľskú a technickú dokumentáciu.

### Zoznam odbornej literatúry:

1. Patel, S. K. (2015). Learning web component development. Birmingham, England: Packt Publishing

Riešenie zadania práce od: **15. 02. 2021**

Dátum odovzdania práce: **04. 06. 2021**

**Dominik Fullajtár**  
študent

**Dr. rer. nat. Martin Drozda**  
vedúci pracoviska

**Dr. rer. nat. Martin Drozda**  
garant študijného programu

# SÚHRN

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE  
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Študijný program :	Aplikovaná informatika
Bakalárska práca:	Využitie štandardu Web Components pre návrh webovej aplikácie
Autor:	Dominik Fullajtár
Vedúci záverečnej práce:	doc. Ing. Oto Haffner, PhD.
Konzultant ak bol určený:	Ing. Erich Stark, PhD.
Miesto a rok predloženia práce:	Bratislava 2021

Bakalárska práca sa zaoberá využitím natívnych komponentov, spĺňajúcich štandard Web Components pri vývoji webovej aplikácie. Konkrétne sa zameriame na vývoj fakturačnej aplikácie. V úvodnej časti práce sú vysvetlené pojmy objednávka a faktúra a ich náležitosti vzhľadom na Slovenský zákonník. V úvode sú taktiež spomenuté základné funkcionálne požiadavky na aplikáciu a rozbor existujúcich riešení. Ďalšia časť je venovaná použitým technológiám a ich využitiu pri tvorbe aplikácie. Nasleduje popis návrhu našej aplikácie. Posledná časť sa zaoberá celkovou implementáciou aplikácie pričom dôraz je kladený na spôsob tvorby komponentov s využitím Stencil.js. V tejto záverečnej časti je detailne opísaná výsledná webová aplikácia.

Kľúčové slová: web components, aplikácia, node.js, stencil.js, faktúra

# ABSTRACT

SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA  
FACULTY OF ELECTRICAL ENGINEERING AND INFORMATION  
TECHNOLOGY

Study Programme:	Applied Informatics
Bachelor Thesis:	Use of standard Web Components for web application design
Autor:	Dominik Fullajtár
Supervisor:	doc. Ing. Oto Haffner, PhD.
Consultant:	Ing. Erich Stark, PhD.
Place and year of submission:	Bratislava 2021

The bachelor thesis deals with the use of native components that meet the Web Components standard in the development of a web application. We will focus specifically on the development of billing application. The introductory part of the thesis explains the terms order and invoice and their requirements with respect to the Slovak law-book. The introduction also mentions the basic functional requirements for the application and analysis of existing solutions. The next part is devoted to the technologies used and their use in creating the application. The following is a description of the design of our application. The last part deals with the overall implementation of the application, with emphasis on the method of creating components using Stencil.js. The final part describes in detail the resulting web application.

Key words: web components, application, node.js, stencil.js, invoice

## **Vyhlásenie autora**

Podpísaný Dominik Fullajtár čestne vyhlasujem, že som Bakalársku prácu Využitie štandardu Web Components pre návrh webovej aplikácie vypracoval na základe poznatkov získaných počas štúdia a informácií z dostupnej literatúry uvedenej v práci.

Uvedenú prácu som vypracoval pod vedením doc. Ing. Oto Haffner, PhD.

V Bratislave dňa 11.06.2021

.....

podpis autora

# **Pod'akovanie**

Ďakujem môjmu konzultantovi Ing. Erich Stark, PhD. za všetky jeho cenné a odborné rady, hlavne za všetok čas venovaný pomoci pri vypracovaní tejto bakalárskej práce.

# Obsah

Zoznam použitých skratiek a označení .....	7
Úvod.....	1
1 Úvod do problematiky .....	2
1.1 Objednávka.....	2
1.2 Faktúra.....	3
1.2.1 Povinné náležitosti faktúry tak, ako ich definuje zákon o DPH: .....	3
1.2.2 Lehota na vyhotovenie faktúry .....	4
1.2.3 Uchovávanie faktúr .....	5
1.3 Fakturačný Systém .....	6
1.4 Analýza súčasného stavu dostupných aplikácií .....	6
1.4.1 SuperFaktúra .....	7
1.4.2 Profitco.....	8
1.4.3 Odoo.....	9
2 Použité technológie .....	11
2.1 Node.js.....	11
2.2 NestJS.....	11
2.3 TypeORM.....	11
2.4 JavascRipt .....	12
2.4.1 Web components.....	12
2.5 TypeScript .....	15
2.6 MVC.....	15
2.6.1 Model.....	15
2.6.2 View .....	16
2.6.3 Controller .....	16
2.7 Express.js.....	16
2.8 Express Handlebars .....	17
2.9 Tailwind CSS .....	18
2.10 Stencil.js .....	18
2.11 Použité JavaScript Knižnice .....	19
2.11.1 Chart.js .....	19



2.11.2	Datatables.js .....	19
3	Návrh riešenia.....	20
3.1	Popis aplikácie.....	20
3.2	Prípady použitia.....	21
3.3	Wireframe.....	22
3.4	Objektový model .....	24
4	Implementácia riešenia .....	26
4.1	Databáza .....	26
4.2	API .....	26
4.3	Autentifikácia používateľa .....	27
4.4	Web Components .....	27
4.4.1	Postup tvorby komponentu v StencilJS .....	28
4.4.2	Štýlovanie .....	32
4.5	Prehľad vytvorených komponentov .....	32
4.5.1	Button.....	32
4.5.2	Input a label .....	34
4.6	Výsledná webová aplikácia .....	34
4.6.1	Vstup do aplikácie .....	35
4.6.2	Prehľad.....	35
4.6.3	Faktúry .....	36
4.6.4	Objednávky .....	39
4.6.5	Výdavky.....	39
4.6.6	Zamestnanci .....	39
4.6.7	Položky .....	39
4.6.8	Moja firma .....	39
	Záver .....	40
5	Zoznam použitej literatúry.....	41

## Zoznam obrázkov a tabuliek

Obrázok 1: SuperFaktúra – nová faktúra .....	7
Obrázok 2: Profitco - hlavná stránka .....	8
Obrázok 3: Profitco - zobrazenie faktúry .....	9
Obrázok 4: Odoo - hlavná stránka .....	10
Obrázok 5: Odoo - nová faktúra .....	10
Obrázok 6: Grafická reprezentácia Shadow DOM [23] .....	14
Obrázok 7: Vizualizácia MVC architektúry pomocou flow diagramu .....	16
Obrázok 9: Návrh komunikácie medzi segmentmi aplikácie .....	20
Obrázok 10: Diagram prípadov použitia.....	22
Obrázok 11: Wireframe domovskej stránky .....	23
Obrázok 12: Wireframe zoznamu faktúr .....	24
Obrázok 13: Entitno-relačná databáza .....	25
Obrázok 14: Príkaz - npm init stencil .....	28
Obrázok 15: Nastavenia komponentu .....	28
Obrázok 16: Príkaz - stencil generate .....	29
Obrázok 17: Súborová štruktúra generovaných komponentov.....	29
Obrázok 18: Vlastnosti modal komponentu .....	30
Obrázok 19: Metódy modal komponentu .....	31
Obrázok 20: Náhľad modal komponentu (zvýraznený červeným orámovaním).....	31
Obrázok 21: Zobrazenie obrázku modal komponentom.....	32
Obrázok 22: Metódy button komponentu .....	33
Obrázok 23: Výsledný button komponent .....	33
Obrázok 24: Formulár tvorený elementmi.....	34
Obrázok 25: Formulár tvorený komponentmi .....	34
Obrázok 26: Prihlasovanie do aplikácie.....	35
Obrázok 27: Vytvorenie firmy .....	35
Obrázok 28: Domovská stránka - Prehľad.....	36
Obrázok 29: Faktúry .....	37
Obrázok 30: Vytvorenie novej faktúry .....	38

# Zoznam použitých skratiek a označení

API	Aplication programming interface
CSS	Cascading Style sheets
DOM	Document Object Model
HTML	Hypertext Markup Language
I/O	Input/Output
JS	JavaScript
ORM	Object-relational mapping
REST	Representational state transfer
UI	User Interface
URI	Uniform Resource Identifier
URL	Uniform Resource Locat
UX	User Experience

# Úvod

V posledných rokoch sa rozmáha trend pri vývoji webových aplikácií využívať front-end frameworky ako napríklad React, Angular, Vue.js a ďalšie. Tieto frameworky sú orientované na tvorbu komponentov, ktoré sú následne využívané ako časti aplikácie. Pri veľkom množstve týchto frameworkov plynie množstvo nevýhod, pričom najpodstatnejšími sú vzájomná nekompatibilita komponentov vytvorených týmito frameworkmi. Pri využívaní vznikajú závislosti na podpore týchto frameworkov, čo môže viesť k problémom, pokiaľ vybraný framework stratí podporu.

Návrh na riešenie týchto problémov prichádza so štandardom Web Components. Ten ponúka možnosť vytvorenia natívnych JS komponentov. Fungujú na báze štandardných technológií implementovaných vo webovom prehliadači. Vďaka tomu sú kompatibilné so všetkými JavaScriptovými frameworkami a je možné očakávať ich dlhodobú podporu.

Pre ich demonštráciu sme sa rozhodli vytvoriť webovú aplikáciu, kde zaoberáme problematikou fakturačných systémov.

V terajšej dobe informatizácie je dôležitou témou digitalizácia podnikania. My sme si zvolili konkrétne fakturáciu. Naším cieľom je zjednodušiť a sprehľadniť prácu s financiami hlavne malým podnikateľom a živnostníkom, ktorí si nemôžu finančne dovoliť platiť za komplexné nástroje a zamestnanca, ktorý by sa im staral o financie.

Kvôli Slovenskej legislatíve je taktiež potrebné okrem vystavovania faktúr, uchovávať faktúry až desať rokov po vystavení, preto vidíme potenciál v uchovávaní faktúr v prehľadnej elektronickej forme, kde je možné vyhľadať požadovanú faktúru za pár sekúnd. Aplikácia musí byť uspokojená na potrebu vytlačiť požadované doklady v prípade potreby.

# 1 Úvod do problematiky

V tejto kapitole upresníme pojmy objednávka a faktúra. Ako sú definované podľa slovenských zákonov, ich súčasti a význam v podnikaní.

## 1.1 Objednávka

Objednávku posielajú odberateľ dodávateľovi na základe ponuky. Objednávka je listina ktorou sa objednávateľ zaviazuje k odberu tovaru alebo služieb dodávateľovi. Záväznosť pre dodávateľa a aj odberateľa nadobúda po tom ako ju dodávateľ potvrdí.

Objednávka musí obsahovať následné náležitosti podľa §3 zákona 102/2014:

- obchodné meno a sídlo alebo miesto podnikania predávajúceho alebo osoby, v mene ktorej predávajúci koná,
- telefónne číslo predávajúceho a ďalšie údaje, ktoré sú dôležité pre kontakt spotrebiteľa s predávajúcim, najmä adresu jeho elektronickej pošty a číslo faxu, ak ich má,
- adresu predávajúceho alebo osoby, v mene ktorej predávajúci koná, na ktorej môže spotrebiteľ uplatniť reklamáciu tovaru alebo služby, podať sťažnosť alebo iný podnet, ak sa táto adresa líši od adresy uvedenej v písmene b),
- celkovú cenu tovaru alebo služby vrátane dane z pridanej hodnoty a všetkých ostatných daní alebo ak vzhľadom na povahu tovaru alebo služby nemožno cenu primerane určiť vopred, spôsob, akým sa vypočíta, ako aj náklady na dopravu, dodanie, poštovné a iné náklady a poplatky, alebo, ak tieto náklady a poplatky nemožno určiť vopred, skutočnosť, že do celkovej ceny môžu byť zarátané takéto náklady alebo poplatky; ak ide o zmluvu uzavretú na dobu neurčitú alebo dohodu o predplatnom, predávajúci informuje spotrebiteľa o celkovej cene za zúčtovacie obdobie a o cene za jeden mesiac, ak je dĺžka zúčtovacieho obdobia odlišná, a ak túto cenu nemožno určiť vopred, o spôsobe, akým sa vypočíta [21].

## 1.2 Faktúra

Pojem „faktúra“ pochádza z latinského slova „facere“ ktoré znamená robiť, urobiť. Je definovaná ako výmenný vzťah medzi dvoma subjektmi, ktorý sa vzťahuje na dodanie tovaru alebo služby [19].

Podľa § 71 faktúrou je každý doklad alebo oznámenie, ktoré je vyhotovené v listinnej forme alebo elektronickej forme podľa tohto zákona alebo zákona platného v inom členskom štáte upravujúceho vyhotovenie faktúry, elektronickej faktúrou je faktúra, ktorá obsahuje údaje podľa § 74 a je vydaná a prijatá v akomkoľvek elektronickej formáte; elektronickej faktúru možno vydať len so súhlasom príjemcu tovaru alebo služby

Za faktúru sa považuje aj každý doklad alebo oznámenie, ktoré mení pôvodnú faktúru a osobitne a jednoznačne sa na ňu vzťahuje. Faktúrou podľa prvej vety nie je opravný doklad podľa § 25a [1].

Pri vystavovaní faktúr musíme dávať pozor na to, aby obsahovala všetky potrebné informácie. Ak sa tak nestane, môže to mať dopad na odpočítanie dane platiteľom.

### 1.2.1 Povinné náležitosti faktúry tak, ako ich definuje zákon o DPH:

- meno a priezvisko (alebo názov) zdaniteľnej osoby, adresu jej sídla, miesta podnikania, prevádzkarne, bydliska alebo adresu miesta, kde sa obvykle zdržiava, a jej identifikačné číslo pre daň, pod ktorým tovar alebo službu dodala,
- meno a priezvisko (alebo názov) príjemcu tovaru alebo služby, adresu jeho sídla, miesta podnikania, prevádzkarne, bydliska alebo adresu miesta, kde sa obvykle zdržiava, a jeho identifikačné číslo pre daň, pod ktorým mu bol dodaný tovar alebo služba,
- poradové číslo faktúry,
- dátum, keď bol tovar alebo služba dodaná alebo dátum, keď bola platba prijatá, ak tento dátum možno určiť a ak sa odlišuje od dátumu vyhotovenia faktúry,
- dátum vyhotovenia faktúry,
- množstvo a druh dodaného tovaru alebo rozsah a druh dodanej služby,
- základ dane pre každú sadzbu dane, jednotkovú cenu bez dane a zľavy a rabaty, ak nie sú obsiahnuté v jednotkovej cene,

- uplatnenú sadzbu dane alebo oslobodenie od dane; pri oslobodení od dane sa uvedie odkaz na ustanovenie tohto zákona alebo smernice Rady 2006/112/ES z 28. novembra 2006 o spoločnom systéme z pridanej hodnoty v platnom znení alebo slovná informácia „dodanie je oslobodené od dane“,
- výšku dane spolu v eurách, ktorá sa má zaplatiť, okrem výšky dane uplatnenej podľa osobitnej úpravy v § 66,
- slovnú informáciu „vyhotovenie faktúry odberateľom“, ak odberateľ, ktorý je príjemcom tovaru alebo služby, vyhotovuje faktúru podľa § 72 ods. 5,
- slovnú informáciu „prenesenie daňovej povinnosti“, ak osobou povinnou platiť daň je príjemca tovaru alebo služby,
- údaje o dodanom novom dopravnom prostriedku podľa § 11 ods. 12,
- slovnú informáciu „úprava zdaňovania prirážky – cestovné kancelárie“, ak sa uplatní osobitná úprava podľa § 65,
- slovnú informáciu „úprava zdaňovania prirážky – použitý tovar“, „úprava zdaňovania prirážky – umelecké diela“ alebo „úprava zdaňovania prirážky – zberateľské predmety a starožitnosti“, a to v závislosti od tovaru, pri ktorom sa uplatní osobitná úprava podľa § 66,
- meno a priezvisko (alebo názov) daňového zástupcu podľa § 69a alebo § 69aa, adresu jeho sídla alebo bydliska a jeho osobitné identifikačné číslo pre daň, ak zahraničná osoba je zastúpená daňovým zástupcom podľa § 69a alebo § 69aa,

### **1.2.2 Lehota na vyhotovenie faktúry**

Každá faktúra musí byť vyhotovená do 15 dní od:

- a) dňa dodania tovaru alebo služby,
- b) dňa prijatia platby pred dodaním tovaru alebo služby alebo do konca kalendárneho mesiaca, v ktorom bola platba prijatá,
- c) konca kalendárneho mesiaca, v ktorom bol dodaný tovar oslobodený od dane podľa § 43,
- d) konca kalendárneho mesiaca, v ktorom bola dodaná služba alebo prijatá platba pred dodaním služby s miestom dodania podľa § 15 ods. 1 v inom členskom štáte,

- e) konca kalendárneho mesiaca, v ktorom nastala skutočnosť rozhodná pre vykonanie opravy základu dane podľa § 25 ods. 1.

### 1.2.3 Uchovávanie faktúr

- 1) Plateľ je povinný uchovávať
  - a. kópie faktúr, ktoré vyhotovil alebo ktoré vyhotovil v jeho mene a na jeho účet odberateľ alebo tretia osoba, a prijaté faktúry vyhotovené zdaniteľnou osobou alebo treťou osobou v jej mene a na jej účet po dobu desiatich rokov nasledujúcich po roku, ktorého sa týkajú,
  - b. prijaté faktúry vzťahujúce sa na investičný majetok uvedený v § 54 ods. 2 písm. b) a c) do konca obdobia na úpravu odpočítanej dane podľa § 54 a 54a,
  - c. dovozné doklady a vývozné doklady potvrdené colným orgánom do konca kalendárneho roka, v ktorom uplynie desať rokov od skončenia roka, ktorého sa týkajú.
- 2) prijaté faktúry vzťahujúce sa na tovary a služby, pri ktorých je osobou povinnou platiť daň, po dobu desiatich rokov nasledujúcich po roku, ktorého sa týkajú.
  - a. kópie faktúr podľa § 72 ods. 2, ktoré vyhotovila alebo ktoré vyhotovil v jej mene a na jej účet odberateľ alebo tretia osoba, po dobu desiatich rokov nasledujúcich po roku, ktorého sa týkajú,
  - b. prijaté faktúry vzťahujúce sa na tovary a služby, pri ktorých je osobou povinnou platiť daň, po dobu desiatich rokov nasledujúcich po roku, ktorého sa týkajú.
- 3) Právnická osoba, ktorá nie je zdaniteľnou osobou, je povinná uchovávať prijaté faktúry vzťahujúce sa na tovary a služby, pri ktorých je osobou povinnou platiť daň, po dobu desiatich rokov nasledujúcich po roku, ktorého sa týkajú.
- 4) Každá osoba, ktorá predá nový dopravný prostriedok do iného členského štátu, a každá osoba, ktorá kúpi nový dopravný prostriedok z iného členského štátu, je povinná uchovať faktúru o predaji alebo kúpe nového dopravného prostriedku po dobu desiatich rokov nasledujúcich po roku, v ktorom došlo k predaju alebo kúpe.
- 5) Zdaniteľná osoba, ktorá uchováva faktúry elektronicky, je povinná umožniť daňovému úradu na účely kontroly prístup k týmto faktúram, ich sťahovanie a používanie.



- 6) Elektronickým uchovávaním faktúry sa rozumie uloženie údajov vykonané prostredníctvom elektronického zariadenia na spracovanie vrátane digitálnej kompresie a uchovávanie údajov použitím drôtových, rádiových, optických alebo iných elektromagnetických prostriedkov.
- 7) Ak je faktúra vyhotovená alebo prijatá v cudzom jazyku, sú platiteľ a zdaniteľná osoba, ktorá nie je platiteľom, povinní na požiadanie daňového úradu na účel kontroly zabezpečiť jej preklad do slovenského jazyka; ustanovenie osobitného predpisu<sup>29aa</sup>) týmto nie je dotknuté [20].

### 1.3 Fakturačný Systém

Fakturačný systém má slúžiť hlavne malým a stredným podnikateľom, na elektronizáciu vystavených alebo prijatých faktúr. Fakturačný Systém prispeje k zlepšovaniu prehľadu financií pre podnikateľov. Ide o digitalizáciu základných príjmov a výdavkov firmy. K základnej funkcionalite patrí:

- registrácia používateľa a jeho firmy,
- pridávanie prijatých faktúr,
- pridávanie vystavených faktúr pre historické zobrazenie,
- vyhľadanie konkrétnej prijatej alebo vystavenej faktúry v systéme, cez jej číslo, meno vystaviteľa, splatnosti, celkovej sumy,
- filtrovanie nevyplatených faktúr,
- pridávanie značiek na faktúry
- zobrazenie zoznamu zamestnancov s ich kontaktnými údajmi a ich spravovanie,
- nahrávanie dokladov do systému pre účtovníčku,
- upozornenie na zaplatenie DPH pri faktúrach zo zahraničia

### 1.4 Analýza súčasného stavu dostupných aplikácií

Analýza súčasného stavu dostupných aplikácií je spravenie si rozhľadu medzi existujúcimi spracovaniami a riešeniami fakturácie. Najdôležitejšie je všimnúť si a nájsť nedokonalosti ktorých sa dopustili pri návrhu, aby sme sa im vyhli a prišli s lepším riešením problému. Ale taktiež aj zaznamenať dobré vlastnosti a funkcionalitu aby sme sa ňou

inšpirovali a ideálne ju ešte ďalej rozšírili. Všetky webové aplikácie ktoré sme našli fungovali ako predplatená služba, našťastie mali skúšobné verzie s obmedzenou funkcionalitou na vymedzený čas zdarma.

### 1.4.1 SuperFaktúra

Názov: SuperFaktúra

URL: <https://www.superfaktura.sk/>

The screenshot displays the 'Nová faktúra' (New Invoice) form in the SuperFaktúra web application. The form is organized into several sections:

- Header:** The application logo 'SuperFaktúra' is at the top left. A notification bar indicates 'Do konca skúšobného obdobia Vám zostáva 30 dní' (End of trial period, you have 30 days left). A 'Kupit SuperFaktúru' button is at the top right.
- Navigation:** A horizontal menu includes 'Prehľad', 'Faktúry', 'Nákupy', 'Kontakty', and 'Nástroje'. Below this, a sub-menu for 'Faktúry' lists 'Zálohové faktúry', 'Dodávky tovaru', 'Prijaté objednávky', 'Cenné ponuky', 'Právnické', 'Dodávky', and 'Koncepty'.
- Form Fields:**
  - Všeobecné informácie:** Includes fields for 'Faktúra číslo' (2021001), 'Dátum vystavenia' (08.02.2021), 'Odberateľ' (Výber...), 'Výrobiteľský symbol' (2021001), 'Dátum dodania' (08.02.2021), 'Názov faktúry' (Faktúra 2021001), and 'Spadá do' (14 dní).
  - Poznámka nad položkami:** A text area for additional notes.
  - Názov a popis položky:** A table with columns 'Názov a popis položky', 'Počet', 'Jednotka', 'Cena', and 'Celkom'. It includes a 'Zlozka' (Tax) field.
  - Poznámka:** A text area for additional notes.
  - Summary:** A table with columns 'Zlozka' (0 %) and 'Celková suma' (0,00 €).
  - Contact Information:** Fields for 'Faktúru vystavil' (Name), 'Telefón', 'Web', 'Email', and 'Tagy'.
- Footer:** Contains contact information for 'SuperFaktúra, s.r.o.', 'KLIENSKÝ SERVIS' (Customer Service), and links to the 'App Store' and 'Google Play'.

Obrázok 1: SuperFaktúra – nová faktúra

#### 1.4.1.1 Výhody

SuperFaktúra je už pokročilejšia aplikácia. Ponúka možnosti exportu faktúry do PDF, prepojenie s bankou, API s e-shopom a importovanie faktúr z niektorých iných aplikácií. Rozloženie polí pre zápis údajov do faktúry sa značne podobá štandardnému formátu faktúry.

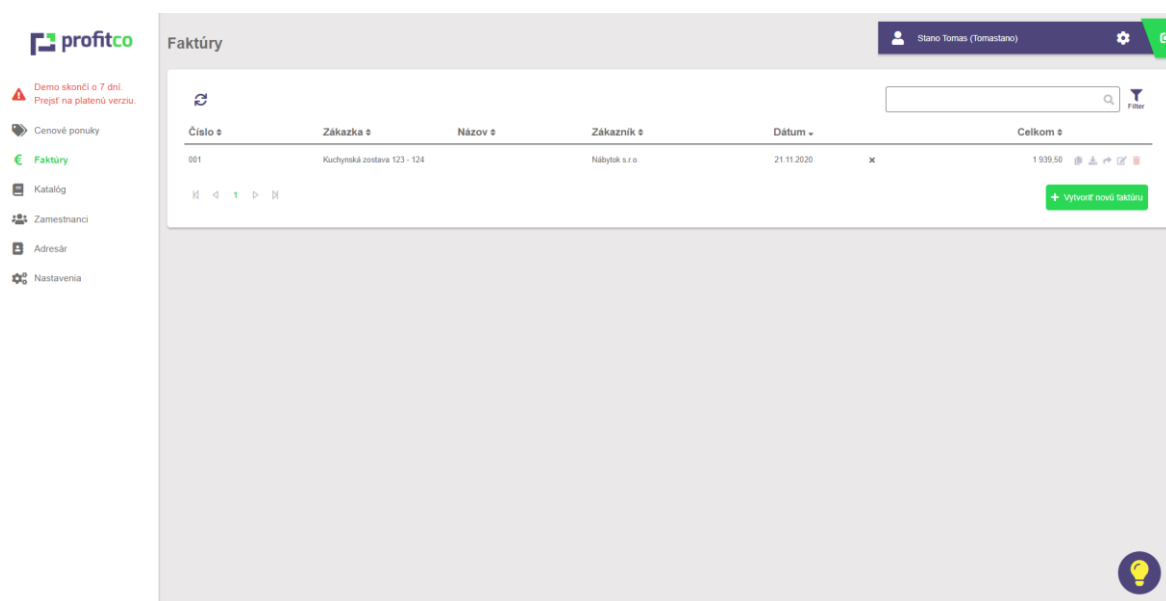
### 1.4.1.2 Nevýhody

Žiadne výrazné nedostatky sme nezaznamenali, aplikácia je plne funkčná a vhodná aj pre nového používateľa, jediné čo by mohlo byť zmenené je vzhľad, avšak to je iba subjektívny pohľad.

### 1.4.2 Profitco

Názov: Profitco

URL: <https://www.profitco.sk/>



Obrázok 2: Profitco - hlavná stránka

Obrázok 3: Profitco - zobrazenie faktúry

#### 1.4.2.1 Výhody

Táto aplikácia na prvý pohľad zaujme moderným a jednoduchým vzhľadom. V ľavej časti vidíme menu, s vhodným členením kategórií, ktoré vytvára jednoduchú a prehľadnú navigáciu.

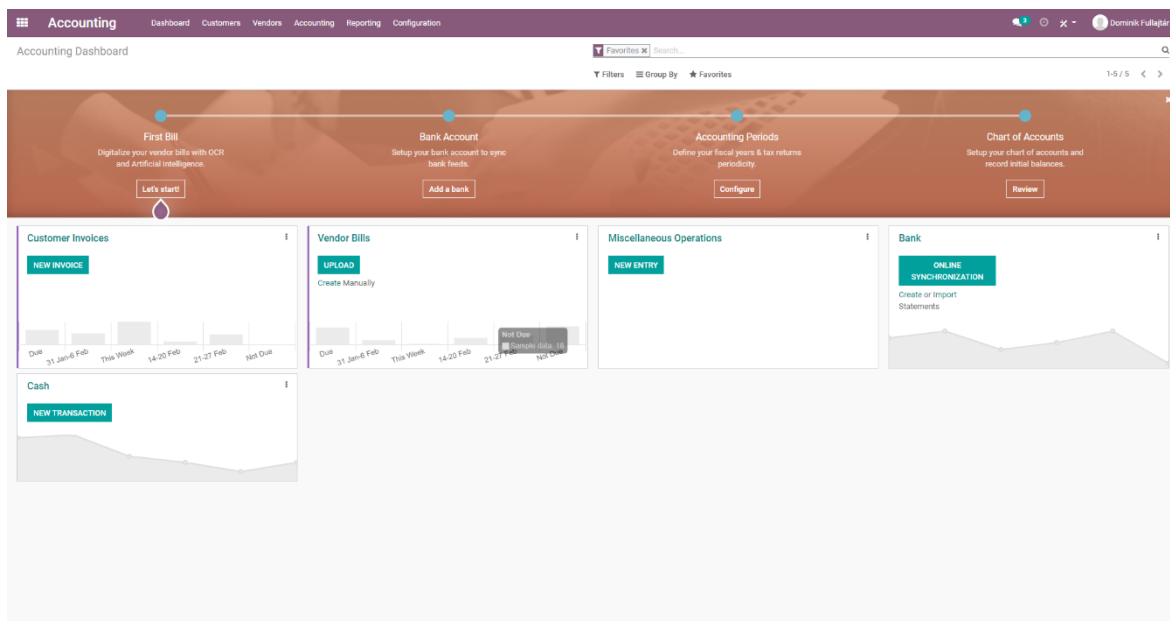
#### 1.4.2.2 Nevýhody

Detail vytvárania faktúry je síce dostačujúci ale rozloženie políček nezodpovedá klasickej papierovej faktúre, z tohto dôvodu by mohlo byť používanie tejto aplikácie pre nového používateľa zo začiatku problematické a musel by si zvyknúť na toto spracovanie. Taktiež sú políčka faktúry rozdelené do pod kategórií „Fakturačné údaje“, „Položky faktúry“, „Zákazník“, „Dodávateľ“. V tomto je taktiež rozdiel od klasickej vzorovej faktúry v ktorej sa všetky informácie nachádzajú na jednej strane a rozširuje to problémy pre nového používateľa. Tomuto prístupu sa pokúsime vyhnúť pretože chceme aby bola naša aplikácia priateľská aj pre nových používateľov.

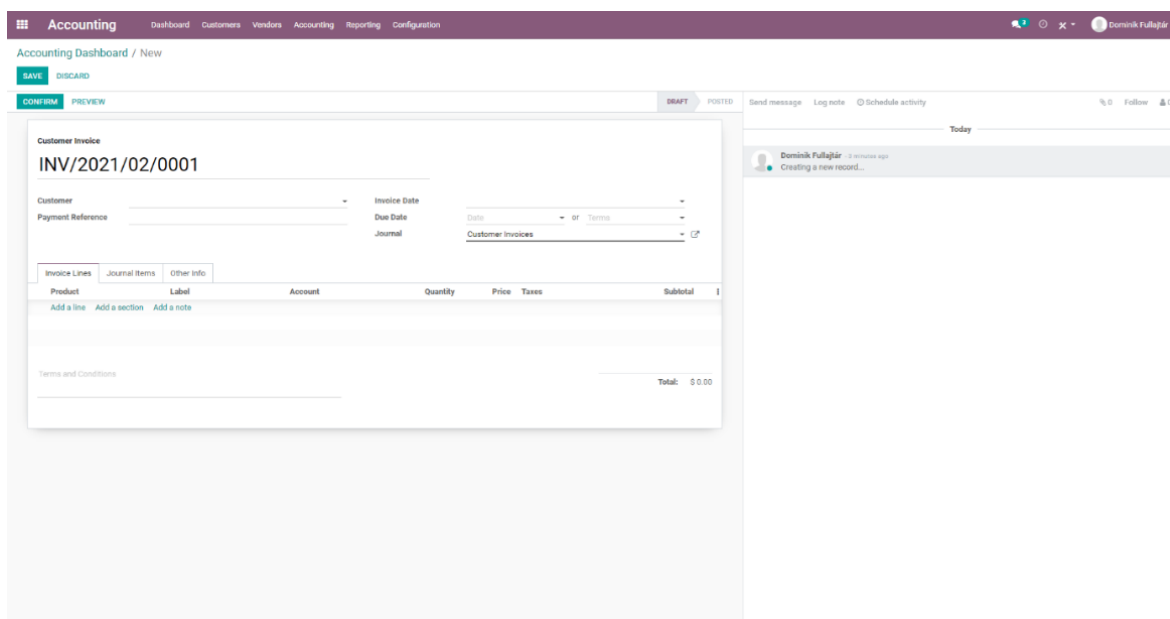
#### 1.4.3 Odoo

Názov: Odoo

URL: <https://www.odoo.com/>



Obrázok 4: Odoo - hlavná stránka



Obrázok 5: Odoo - nová faktúra

### 1.4.3.1 Výhody

Možnosť posielania priamych správ medzi používateľmi, je možné sledovať aktivitu prihlásených používateľov a taktiež sa ukladá záznam zmien v systéme.

### 1.4.3.2 Nevýhody

Zatiaľ nedostupné v slovenskom jazyku.

## 2 Použité technológie

Pred implementáciou riešenia je vhodné oboznámiť sa s technológiami ktoré máme v pláne využiť. Tieto technológie sme vybrali na základe našich zručností a trendov pri tvorbe webových aplikácií.

### 2.1 Node.js

Node.js je platformou založenou na JavaScript runtime, pre jednoduchú tvorbu rýchlych a škálovateľných sieťových aplikácií. Využíva udalosťami riadený, neblokujúci I/O model, čím sa stáva nenáročným a efektívnym. Tento model je vhodný pre real-time aplikácie a v súčasnosti patrí k jedným z najpopulárnejším JavaScript technológiám. Jeho hlavnými výhodami sú veľká NPM komunita s množstvom modulov pripravených na použitie a využitie rovnakého jazyka pre backend aj frontend aplikácie [22].

### 2.2 NestJS

NestJS je framework, vytvorený pre budovanie efektívnych aplikácií na strane servera. Je vyvíjaný v TypeScript avšak podporuje aj JavaScript. Kombinuje elementy objektovo orientovaného programovania, funkcionálneho programovania a reaktívne funkcionálne programovanie [2].

NestJS je dobrým počiatočným bodom pre začínajúcich vývojárov, ktorí sa chcú naučiť využívať moderné webové frameworky pretože používa TypeScript ktorý je v súčasnosti veľmi populárny a je len nadstavbou JavaScriptu. Keďže NestJS využíva na pozadí Express.js, vytvára to možnosť používania balíčkov z ekosystému Node.js [24].

### 2.3 TypeORM

Knižnica TypeORM ako už z názvu vyplýva slúži na objektovo relačné mapovanie, podporujúca TypeScript. Výnimočnou oproti ostatným ju robí to, že podporuje Active Record a Data Mapper modely [3].

## 2.4 JavaScript

JavaScript je skriptovací jazyk ktorý je jednou z hlavných troch súčastí tvorby webových stránok a je taktiež nevyhnutný pri tvorbe webových aplikácií. Popri HTML a CSS vytvárajúcich štruktúru a štýl webu, JavaScript mu pridáva funkcionality a dáva možnosť interakcie návštevníkovi rôznymi spôsobmi. JavaScript je jazyk primárne určený pre použitie na strane klienta, to znamená, že je vykonávaný v browseri návštevníka webu. Samozrejme, že sa nájdu výnimky ako frameworky NestJS alebo NodeJS ktoré umožňujú jeho využitie aj na strane servera [6]. JavaScript umožňuje implementovať komplexné funkcionality webu, všetka funkcionality ktorá je mimo rozsah statického zobrazovania stránky je dosahovaná JavaScriptom ako napríklad upozornenia, interaktívne mapy, interakcia s DOM a mnoho ďalších. Je „nenáročný, interpretovaný“ programovací jazyk. Browser prijme JavaScript kód v jeho pôvodnej textovej forme podľa ktorej bude vykonávaný. Moderné „interpretori“ JavaScriptu využívajú techniku nazývanú „just-in-time“ kompilovanie kódu, ktoré má zvýšiť výkon. Zdrojový kód JavaScriptu je skompilovaný do rýchlejšieho binárneho formátu počas toho ako je používaný, vďaka čomu je „beh“ kódu rýchlejší. Aj napriek tomuto faktoru je však stále považovaný za „interpretovaný“ jazyk pretože jeho kompilácia nastáva počas „behu kódu“ a nie pred jeho spustením [8].

### 2.4.1 Web components

V posledných rokoch sa dizajn UI rýchlo rozvíjal. Tento postup bol možný hlavne vďaka frameworkom ako React a Angular s ich prístupom skladania komponentov do výsledného UI. Napriek týmto inováciám, tieto frameworky vytvorili do určitej miery izolované technológie. Vývojári využívajúci React môžu zdieľať ich komponenty iba s React aplikáciami. Podobne na tom sú Angular komponenty a ich aplikácie. Táto izolácia priniesla presný opak toho, o čom majú byť webové technológie, majú byť otvorené a navzájom kompatibilné. Pre vyriešenie tohto rozdelenia a zároveň zachovania dizajnu na vysokej úrovni vznikol štandard Web Components [25].

Web components je set APIs webového prehliadača, ktoré umožňujú vytvoriť nové, znovu použiteľné, zapuzdrené HTML komponenty (s novými HTML tagmi) využiteľné vo webových stránkach a aplikáciách. Takéto komponenty postavené na štandarde, fungujú

naprieč všetkými modernými prehliadačmi a dajú sa využívať s akoukoľvek JavaScript knižnicou alebo frameworkom. Web components je tvorený technológiami :

- Custom Elements
- ShadowDOM
- HTML templates [9].

#### **2.4.1.1 Custom elements**

Custom elements poskytujú programátorovi možnosť vytvorenia vlastného DOM (Document Object Model) elementu. Je možné dať informácie parseru ako správne skonštruovať element a ako má reagovať na zmeny.

Custom elements sa dajú rozdeliť na 2 skupiny, a to: Customized built-in elements a Autonomous custom elements.

Customized built-in elements sa definujú iba mierne odlišne od Autonomous custom elementov ale majú úplne iné využitie. Customized built-in elements existujú aby mohli byť vlastnosti z už existujúcich HTML elementov znovu použité, alebo teda zdedené a navyše rozšírené novou, vlastnou funkcionalitou, taktiež dovoľujú inštaláciu vlastného správania konštruktora a pridanie životného cyklu (lifecycle hooks).

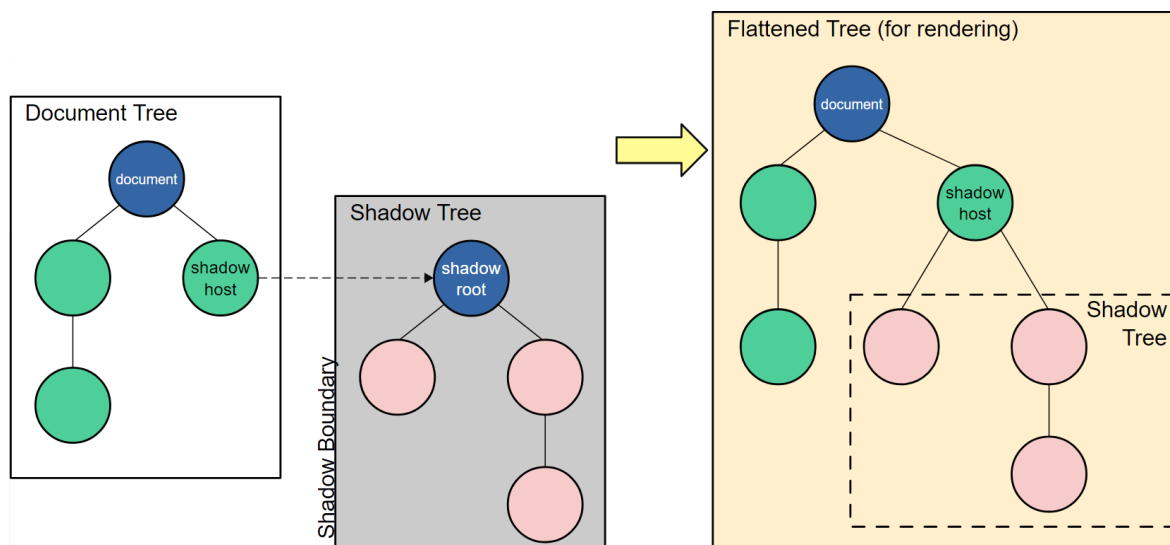
Autonomous custom elements sa rozlišujú v tom, že nededia vlastnosti už existujúcich HTML elementov, sú budované „od nuly“ a z tohto dôvodu je náročnejšie napodobiť funkcionalitu už existujúcich HTML elementov [9].

#### **2.4.1.2 Shadow DOM**

Shadow DOM je strom uzlov, ktorého koreňom je shadow root. Shadow root je vždy pripojený k inému stromu uzlov cez jeho host'a. Z toho vyplýva, že shadow root neexistuje osamote, iba naviazaný na iný strom.

Táto technológia dovoľuje programátorovi efektívne izolovať (zapuzdriť) časť Document Object Modelu, vrátane všetkého čo môže byť použité ako CSS selektor a príslušných štýlov [9].





Obrázok 6: Grafická reprezentácia Shadow DOM [23]

### 2.4.1.3 Template

`<template>` element nám dovoľuje vytvoriť znovu použiteľné šablóny kódu, vo vnútri normálneho HTML súboru, ktoré nebudú okamžite renderované ale môžu byť použité neskôr napríklad cez JS [9].

### 2.4.1.4 Porovnanie prístupu tvorby komponentov

Keďže technológia Web Components je natívnou pre JavaScript, môžeme ich vytvárať aj bez použitia akýchkoľvek knižníc alebo kompilátora. Tento prístup je akceptovateľný pre novšie verzie prehliadačov s podporou Web Components avšak nie je zatiaľ ideálny. Keďže niektoré z prehliadačov nemusia interpretovať správanie vytvorených komponentov rovnakým spôsobom. Ďalšou z nevýhod môže byť veľkosť a efektivita, ktorá nemusí byť optimálna. Problémy taktiež nastávajú pri čitateľnosti kódu alebo štylovaní komponentov. V štylovaní môžu nastať problémy hlavne pri používaní slot tagov, slúžiacich na pridávanie child elementov do komponentu. Tieto slot tagy pri využití nebudú nahradené vloženým elementom ale vnorené do neho čo spôsobuje komplikácie.

Na druhej strane využívanie knižníc a kompilátorov umožňuje rozšírenú funkcionálnu, prehľadnejšie spracovanie kódu tvoriaceho komponent. Kompilátory môžu obsahovať aj polyfill, ktorý zaisťuje funkcionálnu aj na prehliadačoch ktoré nepodporujú danú funkcionálnu. Zabezpečujú taktiež uniformné správanie naprieč rôznymi prehliadačmi. Kompilátor ako napríklad StencilJS môže v produkcii komponenty

minifikovať a spraviť ich menšími z pohľadu veľkosti a rýchlejšími. Ďalej uľahčujú využitie Shadow DOM a môžu podporovať samostatné súbory na aplikáciu štýlov alebo rozširujú funkcionality testovania. Pridávajú nové možnosti využitia pridaných udalostí a mnoho novej funkcionality ktorá závisí už priamo na danej knižnici alebo kompilátore. To všetko pri dodržaní štandardu Web Components.

## 2.5 TypeScript

TypeScript je open-source programovací jazyk ktorý je postavený na JavaScripte, jednom z najpoužívanějších nástrojov na svete, pridaním definícií statických typov. Typy poskytujú spôsob akým sa dá opísať štruktúra objektu, poskytujúc lepšiu dokumentáciu a dovoľujú TypeScriptu validovať, či kód pracuje správne [10].

Z tohoto dôvodu sme ho zvolili ako hlavný programovací jazyk pre backend v NestJS a frontend v StencilJS.

## 2.6 MVC

MVC (Model-View-Controller) je softwarová architektúra ktorá rozdeľuje aplikáciu do troch hlavných prepojených logických častí:

- Model
- View
- Controller

Každá z týchto častí je vytvorená k tomu aby mala na starosti špecifický aspekt aplikácie a majú rôzne účely.

### 2.6.1 Model

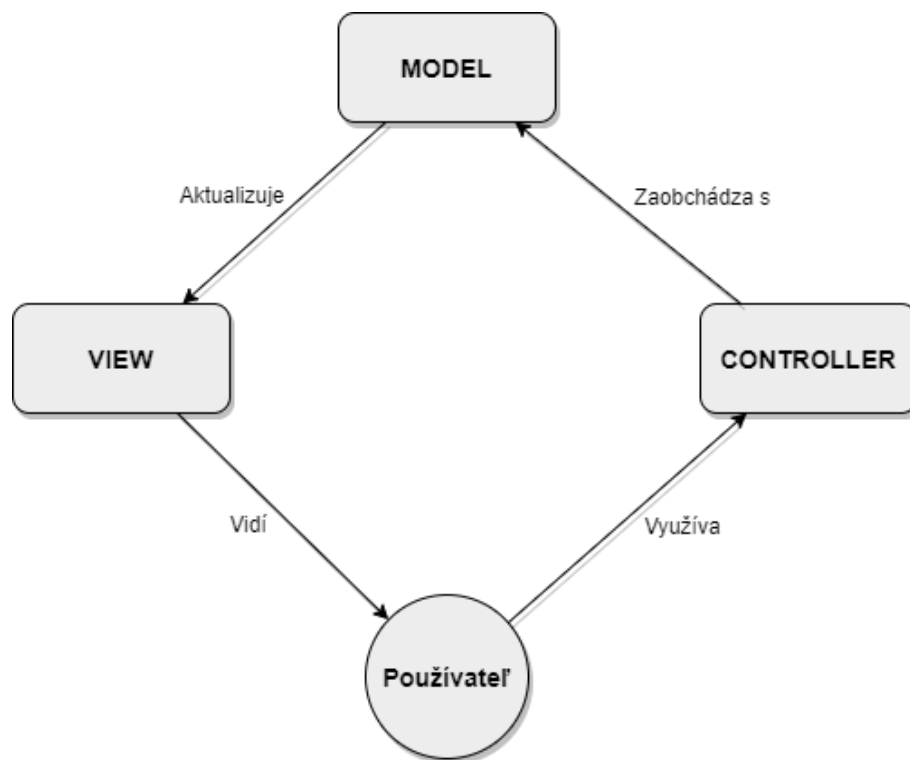
Model je centrálnou časťou MVC architektúry. Obsahuje všetku logiku vzťahujúcu sa k dátam aplikácie, s ktorou používateľ interaguje. Ako napríklad schémy, prepojenie projektu a databáza aplikácie, priamo riadi dáta celej aplikácie. Je nezávislá od UI. Prijíma vstupy používateľa cez controller [11][14].

### 2.6.2 View

View je akoukoľvek reprezentáciou zobrazovania informácií, môže byť vo forme grafu, tabuľky. Graficky reprezentuje model v konkrétnom formáte, v našom prípade vo forme HTML stránky zobrazujúcej dáta a obsahujúcej UI.

### 2.6.3 Controller

Controller spracováva vstupy a požiadavky od používateľa, ktoré validuje. Následne na základe týchto údajov interaguje s Model objektom, ktorý dáta spracuje a Controller odosiela reakciu na prijatú požiadavku [12].



Obrázok 7: Vizualizácia MVC architektúry pomocou flow diagramu

## 2.7 Express.js

Express.js je backend framework pre webové aplikácie, vydaný pod MIT licenciou. Pôvodnú a prvú verziu vytvoril TJ Holowaychuk dňa 22. mája, 2010. Je používaný pre tvorbu web aplikácií v čo najkratšom čase. Keďže je využíva JavaScript, je jeho využívanie jednoduchšie, a to hlavne pre začínajúcich programátorov keďže nie je potrebné sa učiť ďalší programovací jazyk. Je nenáročný a pomáha jednoducho vytvoriť REST API webovej

aplikácie. Spracováva smerovanie (routing), relácie (sessions), HTTP požiadavky a spracovanie chýb. Má udalosťami riadenú architektúru, vďaka čomu je môže spracovať veľké množstvo požiadaviek od klienta naraz.

Výhody Express.js:

- Rýchlejší vývoj aplikácie na strane servera, pretože obsahuje množstvo často používaných funkcií, ktoré môže vývojár využiť. Toto umožňuje zrýchlenie tempa a šetrenie času počas vývoja aplikácie pretože si tieto funkcie nemusí vývojár navrhovať sám.
- Middleware poskytuje systematickú organizáciu funkcií v Express.js
- Poskytuje pokročilé routing mechanizmy, ktoré napomáhajú vytvoriť prehľadný a spravovateľný kód na strane servera.
- Uľahčuje hľadanie a opravovanie chýb, pretože poskytuje ladiace mechanizmy, ktoré sú schopné presne určiť miesto na ktorom vo webovej aplikácii dochádza k chybe [16].

## 2.8 Express Handlebars

Od verzie 3.x, Express prestal byť generickým template enginom, z tohto dôvodu bolo potrebné siahnuť po template engine tretej strany, našou voľbou bol Express Handlebars.

Template engine nám dovoľuje používať statické šablóny ktoré reprezentujú view v architektúre MVC. Pri behu kódu, template engine nahradí premenné v šablóne skutočnými hodnotami, a šablónu transformuje do HTML súboru, ktorý bude následne odoslaný klientovi. Tento proces sa uskutočňuje na strane servera a zjednodušuje navrhovanie HTML stránky.

Hlavnými myšlienkami Express Handlebars sú:

- pridať koncept „layout“ ktorý bol odstránený od verzie Express 3.x,
- pridať koncept „partials“,
- efektívny I/O systém, počas vývoja sú šablóny vždy načítavané z disku, avšak pri produkcii sú šablóny vždy uložené v cache pamäti, čo zvyšuje rýchlosť ich spracovávania,

- možnosť jednoducho prekompilovať šablóny a „partials“ pre použitie u klienta, umožňujúc zdieľanie a znovu použitie šablón [13][14][15].

## 2.9 Tailwind CSS

Tailwind je utility, mobile-first CSS framework. Ponúka triedy s prakticky všetkými CSS atribútmi, využíva dosť širokú ale zároveň obmedzenú paletu farieb. Do Tailwindu je možné pridávať vlastné triedy, selektory a atribúty pretože Tailwind je kompilovaný z klasického CSS súboru. Do skompilovaného `tailwind.css` súboru je prenesené všetko z pôvodného súboru, plus pridané všetky triedy Tailwindu. Z tohto dôvodu je možné používať aj funkcionality CSS nad rámec Tailwindu ako napríklad vlastné číslovanie elementov. Z dôvodu poskytovania obrovského množstva tried, nadobúda CSS súbor Tailwindu veľkú veľkosť, čo je v produkcii neprípustné, preto Tailwind automaticky odstráni všetky nevyužité triedy, keď sa projekt pripraví do produkcie. Takto sa z veľkého (3.5 MB) súboru zredukuje veľkosť na väčšinou menej ako 10kB súbor, bez zazipovania. Podporuje samozrejme aj responzívny dizajn, pomocou predpôň, ktoré je možné pridať pred ktorúkoľvek utilitnú triedu [17].

Keďže Tailwind funguje na princípe štylovania pomocou pridávania utilitných tried, môžu nastať problémy s duplicitami a neprehľadným HTML kódom. Ako najväčšiu nevýhodu tohto prístupu sa asi môže považovať následné menenie štylovania, kedy by bolo potrebné nájsť v celom projekte každý jeden výskyt elementu u ktorého je požadovaná zmena a upraviť ho. V tomto prípade je vhodné využiť funkcionality komponentov, ktorým sa touto prácou venujeme. Pomocou štandardu Web Components si môžeme vytvoriť ucelený HTML element kde použijeme triedy z Tailwind, a následne tento element využívať v aplikácií. Týmto docielime že všetko štylovanie môžeme upraviť na jednom mieste.

## 2.10 Stencil.js

Stencil je kompilátor, ktorý generuje Custom Elements v štandarde Web Components. Stencil využíva Virtual DOM, asynchrónne renderovanie, reaktívny data binding, TypeScript a JSX šablóny.

Keďže sa drží štandardu Web Components, komponenty ním vygenerované, fungujú s ktorýmkoľvek frameworkom, alebo môžu byť využívané bez frameworku. Keďže Web

Components sú podporované samotným JavaScript-om. Stencil taktiež okrem klasickej funkcionality Web Componentov, pridáva možnosti ako napríklad prerendering a prácu s objektmi ako s vlastnosťami.

Na rozdiel od priameho využívania Custom Elements stencil prináša možnosť využívať pridané API ktoré zjednoduší tvorbu efektívnych komponentov. API ako napríklad Virtual DOM, JSX a asynchrónne renderovanie uľahčujú tvorbu užitočných komponentov, zatiaľ čo zachovávajú úplnú kompatibilitu so štandardom Web Components.

Pre zlepšenie vývoja komponentov, Stencil ponúka automatické načítanie po zmene v kóde a server zabudovaný v kompilátore na účely vývoja [18].

## **2.11 Použité JavaScript Knižnice**

Pre klientovu stranu aplikácie sme využili niektoré JavaScript knižnice pre zlepšenie UX pri používaní našej aplikácie. Všetky použité knižnice spadajú pod MIT licenciu, sú to open source, zdarma knižnice a ich využívaním neporušujeme žiadne licenčné podmienky.

### **2.11.1 Chart.js**

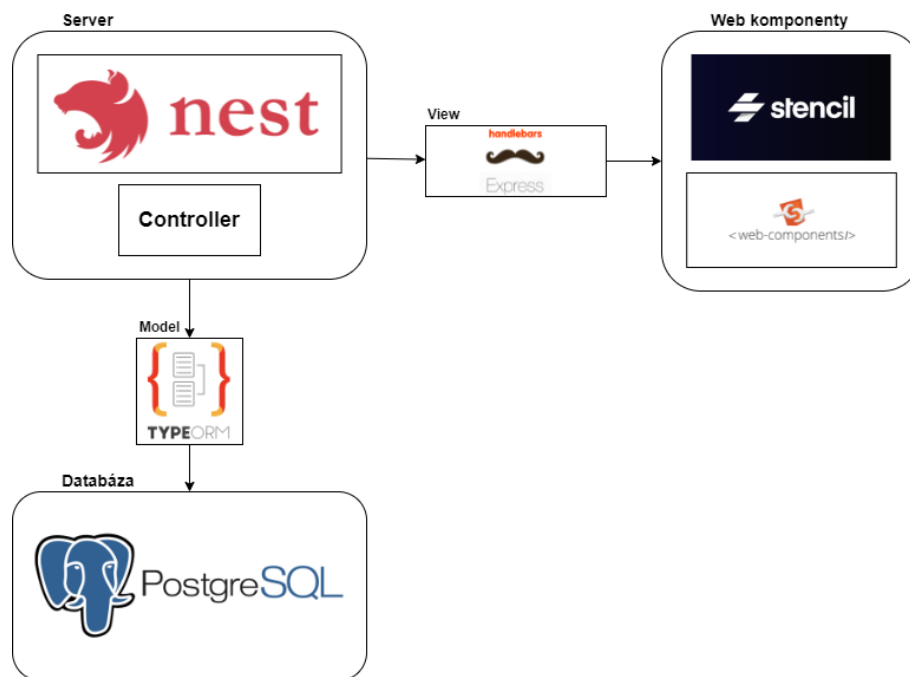
Chart.js je knižnica využívaná na vykresľovanie grafov a tabuliek zo zadaných dát do `<canvas>` elementu. Podporuje množstvo rôznych grafov, s možnosťou použitia vlastných farieb a popisov. Tieto tabuľky sú responzívne.

### **2.11.2 Datatables.js**

Datatables.js je efektívna knižnica využívaná na pokročilú funkcionality HTML tabuliek. My sme ju zvolili hlavne pre jej možnosti stránkovania, vyhľadávania a zoradovania údajov.

### 3 Návrh riešenia

Pri návrhu riešenia sme museli najprv navrhnuť akou funkcionalitou bude aplikácia disponovať, následne sme museli pre túto funkcionalitu navrhnuť vhodnú databázu, oba návrhy sú opísané v tejto kapitole 3.1 a 3.2.



Obrázok 8: Návrh komunikácie medzi segmentmi aplikácie

#### 3.1 Popis aplikácie

Cieľom tejto bakalárskej práce je vytvoriť webovú aplikáciu, ktorá bude využívať webové komponenty, spadajúce pod štandard web components.

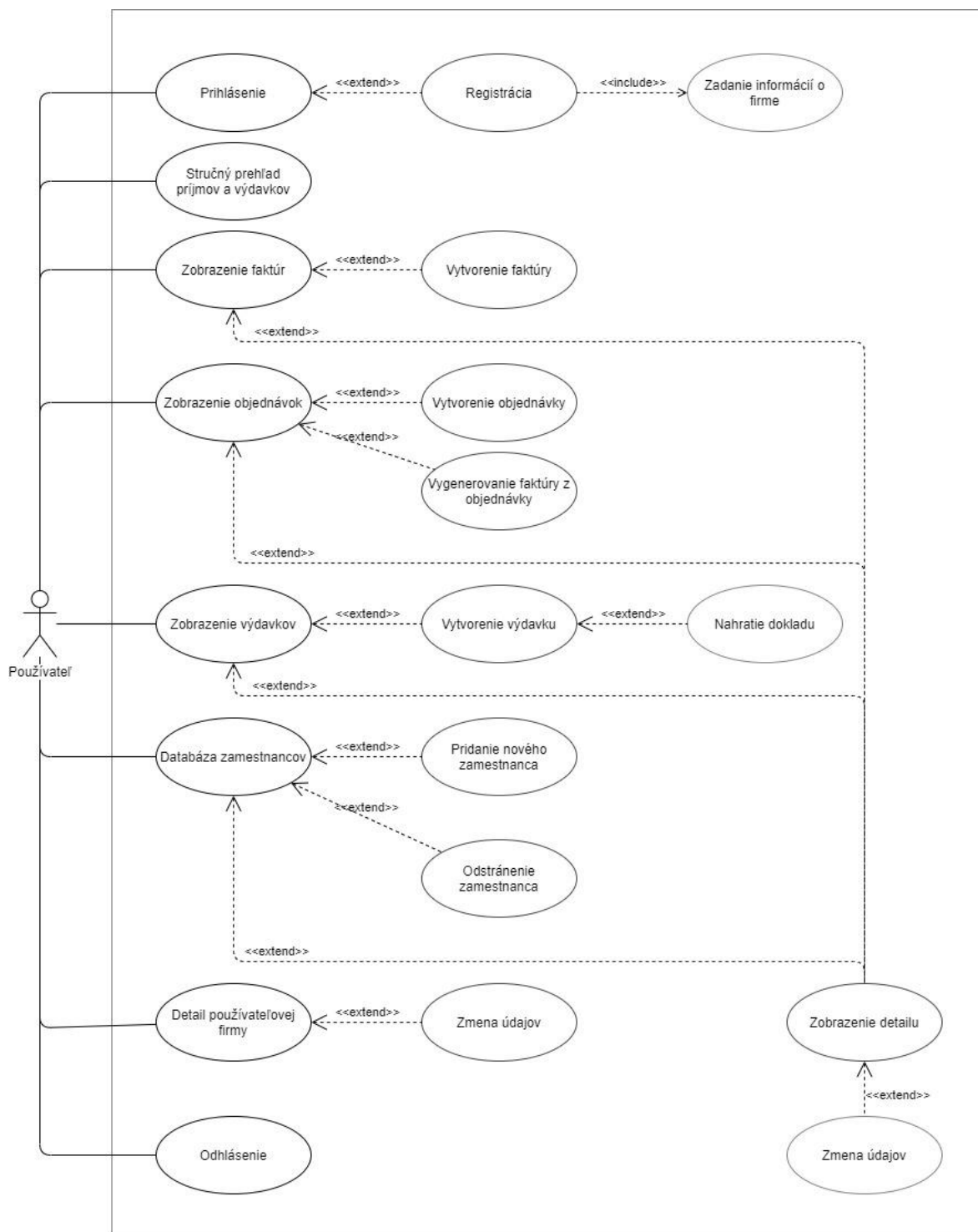
Ako praktickú časť budeme vyvíjať webovú aplikáciu, jej funkciou bude prehľad príjmov a výdavkov, pre malých podnikateľov. Z tohto dôvodu je vhodné zvoliť intuitívne rozhranie, ktoré bude jednoducho použiteľné hlavne pre živnostníka, ktorý podniká na vlastnú päsť.

Návrh a implementáciu sme rozdelili do týchto častí. Backend, v ktorom sme najprv riešili návrh databázy, následne API našej webovej aplikácie, s ktorou bude komunikovať klientská časť. Frontend, v ktorom sme riešili UI a UX aplikácie. Tvorba Web Components a ich následná integrácia do klientskej časti aplikácie.

## 3.2 Prípady použitia

Po navštívení aplikácie, je používateľ vyzvaný k tomu aby sa prihlásil do aplikácie, ak však používateľ ešte nie je registrovaný, má možnosť kliknúť na „*Chcem sa registrovať*“ a následne zadať meno a heslo pod ktorými sa chce zaregistrovať do aplikácie. Po registrácii je presmerovaný na stránku kde vyplní informácie o svojej firme, tie môže dodatočne ešte upravovať. Po úspešnom prihlásení je používateľ presmerovaný na podstránku „*Prehľad*“, kde sa mu zobrazí pomocou grafov, stručný prehľad všetkých príjmov a výdavkov. Na obrázku č. 8. sú zobrazené všetky prípady použitia pre používateľa.



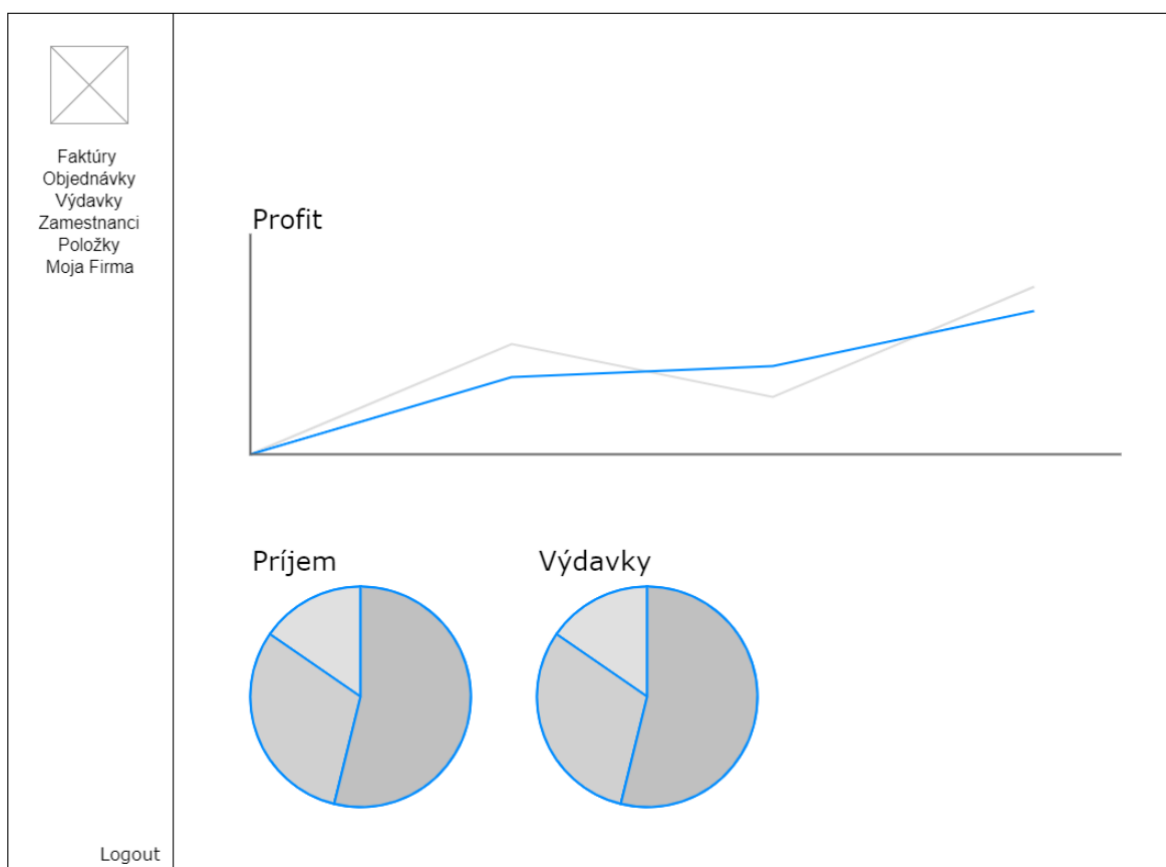


Obrázok 9: Diagram prípadov použitia


### 3.3 Wireframe

Po návrhu prípadov použitia a databázy, sme mohli prejsť k navrhovaniu vzhľadu a rozloženia elementov webovej aplikácie. Keďže už máme navrhnuté prípady použitia,

môžeme bez problémov navrhnuť štruktúru navigácie. Taktiež návrh databázy nám môže dať predstavu, aké údaje budeme zobrazovať a tomu prispôbiť vzhľad stránky. Pre každú podstránku aplikácie sme si pripravili príslušný wireframe, týmto prístupom sme si pomohli pri návrhu dizajnu. Pomohlo nám to aj vyhnúť sa nežiadaným UX chybám, ktoré by sme museli následne upravovať v už fungujúcej aplikácii, čo by nám skomplikovalo prácu. Nižšie na obrázkoch č. 10 a č. 11 sú zobrazené návrhy domovskej stránky a prehľad faktúr. Na demonštráciu sme zvolili iba nasledovné dva, pretože patria k najpodstatnejším podstránkam aplikácie a ukážka všetkých podstránok by bola rozsahovo nevhodná.



Obrázok 10: Wireframe domovskej stránky



Faktúry  
Objednávky  
Výdavky  
Zamestnanci  
Položky  
Moja Firma

Faktúry

Číslo	Dátum	Zákazník	Zákazka	Suma

Logout

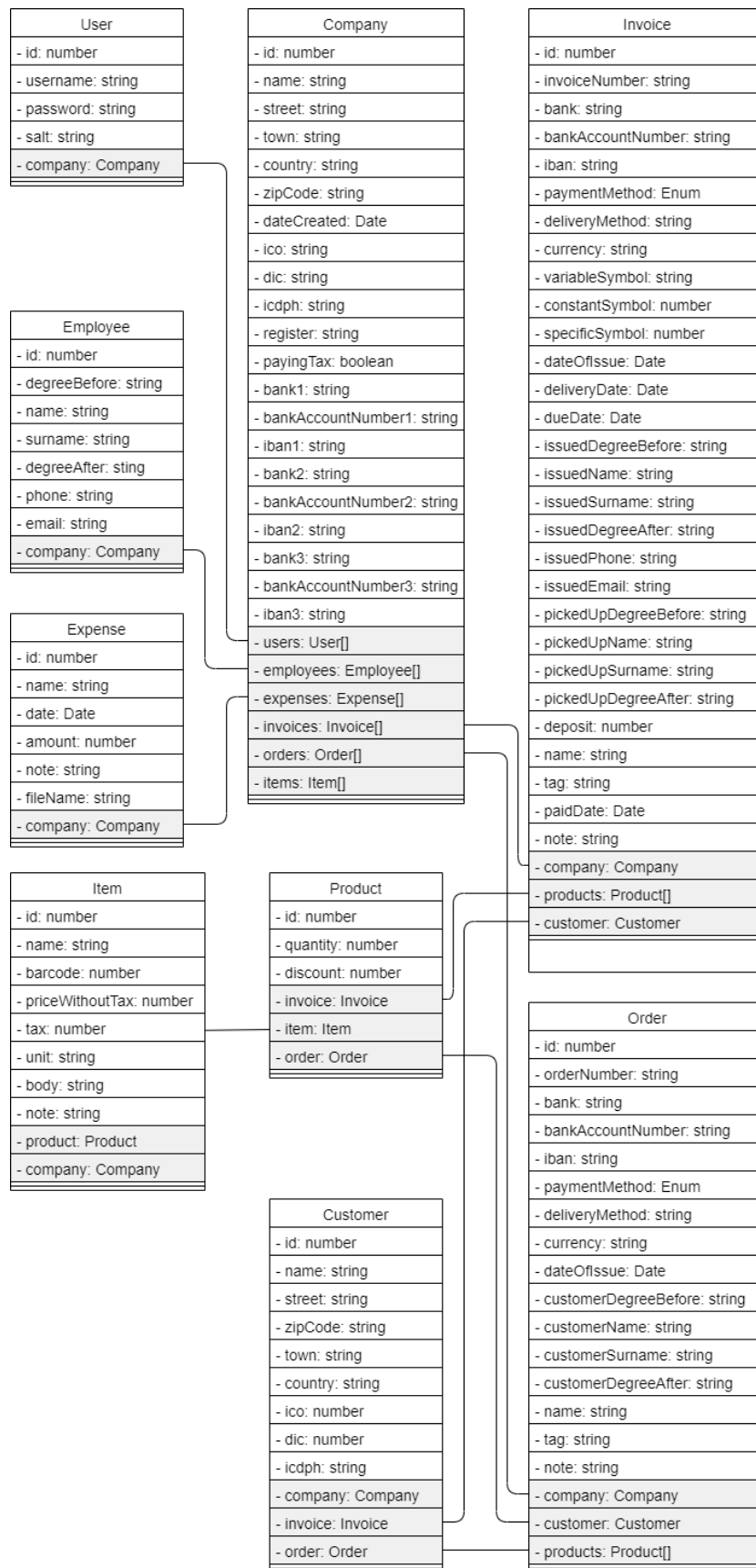
<< 1 >>

Nová faktúra

Obrázok 11: Wireframe zoznamu faktúr

### 3.4 Objektový model

Na základe vytvoreného objektového modelu, definovaného entitnými triedami, sa transformuje cez TypeORM do databázy generovanej databázy.



Obrázok 12: Entito-relačná databáza

## 4 Implementácia riešenia

V tejto kapitole bude opísané aký postup a metodiku sme zvolili pri implementácii riešenia.

### 4.1 Databáza

Na komunikáciu a automatickú tvorbu vzťahov sme využili TypeORM. Pre každú Entitu sme vytvorili vlastnú triedu, ktorá je v databáze interpretovaná ako samostatná tabuľka. V tejto triede sme definovali vlastnosti, ich typy a vzťahy s inými Entitami ktoré sú v tabuľkách databázy interpretované ako stĺpce tabuľky. Z týchto Entitných tried bola následne vygenerovaná celá databáza, ktorej diagram možno vidieť na obrázku č. 11.

### 4.2 API

Pre tvorbu API sme zvolili DTO (Data Transfer Object). Výhodou tohto dizajnového vzoru je, že dáta s viacerými atribútmi odošle naraz, vďaka čomu je odosielanie dát uskutočnené jedným volaním. Týmto prístupom znížime počet volaní a zaťaženie serverovej strany aplikácie. Tento dizajnový vzor je pre náš typ aplikácie vhodný kvôli tomu že pri vytváraní objednávok a faktúr, budeme odosielať serveru veľa dát, takýmto spôsobom systém odosielania a spracovávania spravíme efektívnejším[4].

API aplikácie je štruktúrované do troch hlavných častí. Prvou z častí je Controller, ten je zodpovedný za spracovanie HTTP požiadaviek, prichádzajúcich od klienta. Controller rozozná akou metódou spracovať požiadavku na základe URI, a metódy odosielania požiadavky GET alebo POST. Požiadavku obsluži za pomoci vrstvy Service, ktorá obsahuje takmer všetku spravujúcu logiku. Táto vrstva pri potrebe interakcie s údajmi v databáze, bez ohľadu na to či ide o zápis nového záznamu, úpravu existujúceho záznamu alebo jeho vymazanie, pristupuje k ďalšej vrstve, ktorou je Repository. Táto vrstva je zodpovedná za spravovanie dát príslušnej Entity a priamo pristupuje k tabuľke dát tejto príslušnej Entity.

Každý z Entít našej aplikácie prislúchajú na ich správu vlastné triedy Repository, Service a väčšina má aj vlastnú triedu Controller. Triedu Controller nemá iba Entita Product a to z toho dôvodu že všetka jej obsluha je volaná cez „InvoiceController“ a

„OrderController“, keďže Product je časťou objednávky alebo faktúry ktorá ukladá počet a zľavu produktu.

### 4.3 Autentifikácia používateľa

Autentifikáciu používateľa sme riešili použitím Sessions, ktorých funkcionality sme do aplikácie pridali pomocou modulu „express-session“. Do Session sme pri úspešnom prihlásení uložili ID používateľa, na základe tohto ID má používateľ prístup k správe jeho firmy. Pri odhlásení používateľa sa Session vynuluje, čím sa klientovi odoprie prístup k používaniu aplikácie a pri pokuse o navštívenie stránky ku ktorej má prístup iba prihlásený používateľ, bude klient presmerovaný na prihlasovaciu stránku.

### 4.4 Web Components

Do webových komponentov sme chceli zabudovať drobné prvky aplikácie. Integrácia všetkých využitých elementov do komponentov neprichádzala do úvahy, pretože nie je veľký význam tráviť čas pri tvorení webového komponentu, ktorý nakoniec v aplikácii použijeme iba raz a jeho funkcionality nebude žiadnym spôsobom vylepšená. Preto po analýze našej aplikácie, sme uznali za vhodné vytvoriť komponenty hlavne zo súčastí formulárov, ktoré sú hlavnou súčasťou našej aplikácie a to konkrétne z nasledovných elementov:

- Achor
- Button
- Input
- Label
- Select
- Textarea
- Kombinácia Input a Label

Neskôr sme pri implementácii výdavkov narazili na problém so zobrazovaním obrázkov výdavkov, ktoré aby boli pri zobrazení čitateľné, zaberali príliš veľkú plochu na stránke. To ju robilo zle používateľnou pretože používateľ musel zrolovať až na spodok stránky aby uložil zmeny. Tento problém sa dá avšak ukážkovo vyriešiť pomocou vlastného modal komponentu, ktorý sme neskôr vytvorili.

### 4.4.1 Postup tvorby komponentu v StencilJS

Vytvorenie komponentu je vďaka StencilJS funkcionalite jednoduché a ľahko pochopiteľné. Pred začatím implementácie samotného komponentu je potrebné si najprv vygenerovať potrebné súbory, o tento akt sa postarajú stencil príkazy. V adresári kde chceme vytvoriť StencilJS projekt, je potrebné si inicializovať StencilJS pomocou príkazu `npm init stencil`, po zadaní tohto príkazu dostaneme možnosť zvoliť si či chceme vytvoriť ionic PWA (Progresívnu Webovú Aplikáciu), Stencil aplikáciu alebo kolekciu komponentov. My sme si zvolili poslednú možnosť aby sme vytvorili komponenty pre našu už existujúcu webovú aplikáciu.

```
? Pick a starter » - Use arrow-keys. Return to submit.

  ionic-pwa      Everything you need to build fast, production ready PWAs
  app           Minimal starter for building a Stencil app or website
> component     Collection of web components that can be used anywhere
```

Obrázok 13: Príkaz - `npm init stencil`

Následne po zadaní názvu projektu, bol vygenerovaný StencilJS projekt na vývoj balíčku komponentov, s prvým vzorovým komponentom s názvom *my-component*. V tomto vzorovom komponente si môžeme prezrieť štruktúru komponentu. Pred triedou vzorového komponentu máme dekorátor `@Component()`, v ktorom máme možnosť upraviť nastavenia komponentu ako napríklad nastavenie tagu ktorým bude component volaný v HTML súbore, cestu k štýlovaciemu súboru, možnosť povolenia Shadow DOM alebo cestu k *assets* súboru, v ktorom môžeme uložiť napríklad SVG ikony potrebné ku komponentu.

```
4  @Component({ opts: {
5    tag: 'my-component',
6    styleUrls: ['my-component.css'],
7    shadow: true,
8  })
```

Obrázok 14: Nastavenia komponentu

Následne môžeme vidieť triedu vzorového komponentu, ktorý má tri vlastnosti, ich hodnoty sa priradia pomocou atribútov pri volaní komponentu v HTML súbore. Tieto vlastnosti sú využité v metóde `getText()`, ktorá spája hodnoty týchto troch vlastností a vráti

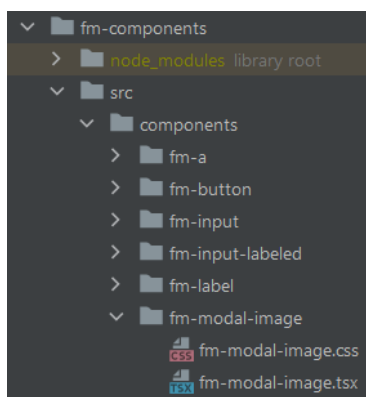
vo forme *string*. Túto metódu volá komponent počas vykonávania metódy *render()*. Metóda *render()* je zodpovedná za vykreslenie obsahu komponentu na stránku, pokiaľ nastane v komponente zmena, ktorú treba vykresliť, táto metóda musí byť opätovne zavolaná aby sa zmena zobrazila aj v klientovi.

Pre vygenerovanie vlastného komponentu, je potrebné zadať príkaz stencil *generate nazov-komponentu* alebo v prípade, pokiaľ na zariadení nie je globálne nainštalovaný Stencil príkazom *npx stencil generate nazov-komponentu*. Je potrebné, aby sa v názve komponentu nachádzala pomlčka, inak vyhodí chybu, že validný názov komponentu musí obsahovať pomlčku. Následne dostaneme možnosť si vybrať, ktoré súbory chceme pre daný komponent vytvoriť.

```
? Which additional files do you want to generate? »
Instructions:
  ↑/↓: Highlight option
  ←/→/[space]: Toggle selection
  a: Toggle all
  enter/return: Complete answer
(*) Stylesheet (.css)
( ) Spec Test (.spec.tsx)
( ) E2E Test (.e2e.ts)
```

Obrázok 15: Príkaz - stencil generate

Pre náš modal komponent, na ktorom demonštrujeme postup vytvorenia komponentu v StencilJS, sme si zvolili generovanie CSS súboru. Pretože pre tento komponent využijeme funkcionality Shadow DOM, testovacie súbory sú pre nás nepotrebné. Komponent sme nazvali *fm-modal-image* a súbory komponentu sú vygenerované v adresári *src/components/nazov-komponentu*.



Obrázok 16: Súborová štruktúra generovaných komponentov



V tomto momente je možné začať s vývojom vlastného komponentu. Naším cieľom bolo vytvoriť modal komponent, ktorý zobrazuje ako náhľad obrázkov v požadovanej veľkosti a po kliknutí na tento obrázok sa zobrazí modal okno, s obrázkom v plnej veľkosti pre dobrú čitateľnosť.

Funkcionálne požiadavky na modal komponent s obrázkom:

- náhľad obrázku, ktorému môžeme meniť veľkosť ale jeho pomer strán bude zachovaný,
- po kliknutí na náhľad, sa zobrazí modal okno s obrázkom, ktorý bol v náhľade,
- modal okno bude zaberat' maximálne 80 % šírky a výšky obrázku, pri menších obrázkoch sa im veľkosťou prispôsobí,
- komponent bude zobrazovať primárne vertikálne orientované obrázky, preto musí byť modal okno rolovateľné na výšku,
- po kliknutí na ľubovoľnú časť obrazovky sa modal okno zatvorí.

Kvôli zobrazovaniu obrázkov, sme si definovali vlastnosti *src* a *alt*, ktoré sú atribútmi `<img/>` elementov pre nájdenie zdroja obrázku a jeho popisu. Pre implementáciu funkcionality otvárania a zatvárania modal okna, sme si definovali vlastnosť pomocou dekorátora `@State()`, hodnotu tejto vlastnosti vie používateľ meniť iba z vnútra komponentu a nefunguje ako atribút. Pri zmene tejto hodnoty je volaná metóda *render()*, toto správanie využijeme na otváranie a zatváranie modal okna takže mu dáme názov *isOpen* a hodnoty budú typu *boolean*.

```
10      /**
11       * Default image attribute behaviour
12       */
13      @Prop() src: string;
14
15      /**
16       * Default image attribute behaviour
17       */
18      @Prop() alt: string;
19
20      @State() isOpen: boolean;
```

Obrázok 17: Vlastnosti modal komponentu

Na základe boolovskej hodnoty budeme cez ternárny operátor elementom vo vnútri komponentu meniť triedy. Na tieto triedy aplikujeme rôzne štýly, ktorými budeme v požadovaných elementoch meniť viditeľnosť. Hodnotu *isOpen*, môžeme zmeniť za pomoci metód a *onClick* Eventov.

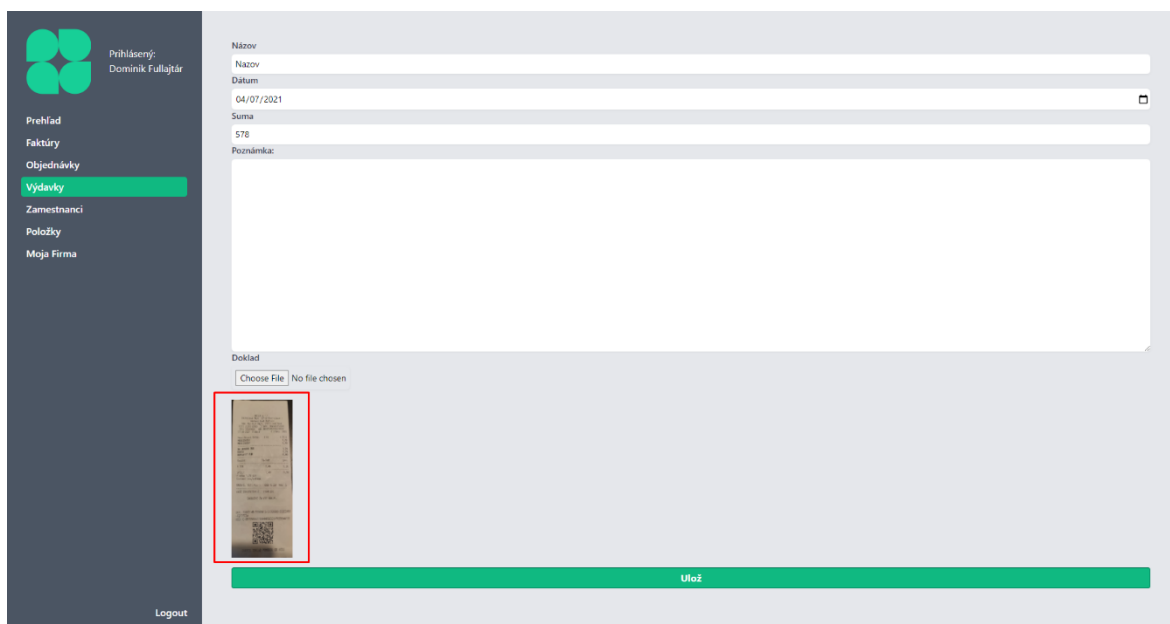
```

22  show() {
23    this.isOpen = true;
24  }
25
26  hide(){
27    this.isOpen = false;
28  }
29
30  render() {
31    return [
32      <img onClick={this.show.bind(this)} class="preview" src={this.src} alt={this.alt}/>,
33      <div onClick={this.hide.bind(this)} class={this.isOpen ? 'backdrop is-open' : 'backdrop'}>
34        <div class={this.isOpen ? 'modal is-open' : 'modal'}>
35          <img class="view" src={this.src} alt={this.alt} />
36        </div>
37      </div>
38    ];
39  }

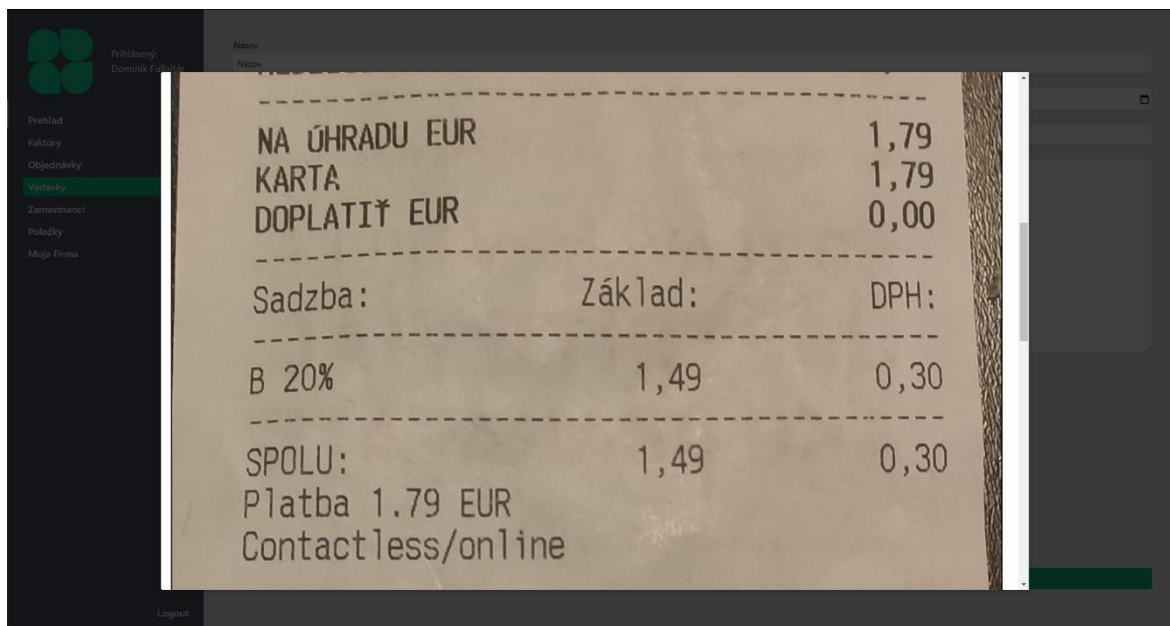
```

Obrázok 18: Metódy modal komponentu

Po naštýlovaní komponentu v príslušnom súbore *fm-modal-image.css* je komponent dokončený. Aby sme komponent mohli využiť v našej aplikácii, musíme si komponent skompilovať príkazom *npm run-script build*. V priečinku *dist* sa nachádzajú skompilované elementy, ktoré môžeme pridať do našej aplikácie. Tieto komponenty potom importujeme do HTML ako *<script/>* typu *module*.



Obrázok 19: Náhľad modal komponentu (zvýraznený červeným orámovaním)



Obrázok 20: Zobrazenie obrázku modal komponentom

#### 4.4.2 Štýlovanie

Štandardne sa na štýlovanie komponentov za využitia StencilJS využíva Shadow DOM, na izolovanie súčastí komponentov od globálneho štýlovania a následne sa pre štýlovanie konkrétneho komponentu používa príslušný súbor CSS alebo SASS. Pre naše účely to však nebolo vhodné, keďže využívame TailwindCSS. TailwindCSS obsahuje triedy štýlov ktoré sa pridávajú priamo do HTML a tým sa aplikujú iba na konkrétny element. Týmto prístupom sa vyhneme prepisovaniu štýlov komponentov a Shadow DOM sa pre nás tým stáva zbytočným. Keby sme využili Shadow DOM, mohlo by to vytvoriť duplicity tried, keďže triedy z globálneho CSS by sme museli prepísať aj do príslušných štýlovacích súborov komponentov. Pre demonštračné účely sme ale Shadow DOM využili v modal komponente.

### 4.5 Prehľad vytvorených komponentov

Okrem modal komponentu sme vytvárali v StencilJS aj iné často využívané prvky aplikácie. Tie najpodstatnejšie sú opísané v tejto kapitole.

#### 4.5.1 Button

V tomto komponente nevyužívame ShadowDOM ale globálny štýlovací tailwind súbor, čo nám dopomohlo v rozšírení funkcionality pre štýly. Vďaka sémantickej štruktúre

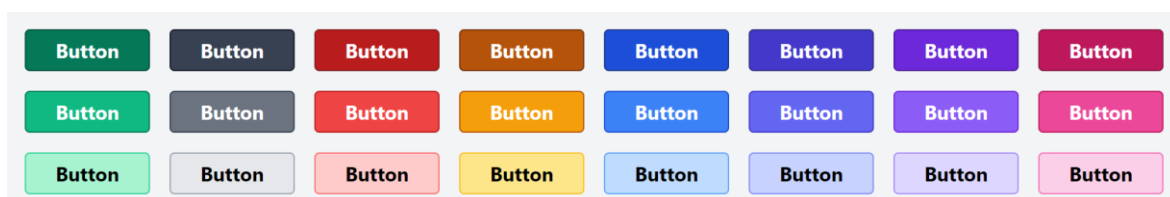
tailwindu sme pomocou atribútu *color* pridali možnosť meniť farebnú schému tlačidla. A pomocou atribútu *shade* jeho odtieň, ktorý je definovaný ako enumerácia číselných hodnôt. dáva to možnosť rýchleho zmenenia farebnej schémy bez duplicitných tried. Toto správanie bolo docielené využitím metódy *stringifyClass()*, ktorá spracováva hodnoty týchto atribútov a jej návratovou hodnotou je reťazec tailwind tried pre požadované štylovanie. Na obrázku č. 20 môžeme vidieť, že pri renderovaní tohto komponentu je zavolaná metóda *stringifyClass()*, ktorej návratová hodnota nastaví triedy button elementu. Na obrázku je tiež vidieť, akým spôsobom sa dá definovať enumerácia vlastností s počiatočnou hodnotou.

```

72  ▶ @Prop() shade: 100|200|300|400|500|600|700|800|900 = 500;
73
74  private getBackgroundShade(): number{
75    if (this.shade > 700){
76      return 700;
77    }
78    return this.shade;
79  }
80
81  private getBorderShade(): number{
82    if (this.shade > 700){
83      return 900;
84    }
85    return this.shade+200;
86  }
87
88  private stringifyClass(){
89    return "bg-"+this.color+"-"+this.getBackgroundShade()+
90      " hover:bg-"+this.color+"-"+this.getBorderShade()+
91      " focus:bg-"+this.color+"-"+this.getBorderShade()+
92      " border-"+this.color+"-"+this.getBorderShade()+
93      " text-"+this.text+
94      " w-full font-bold py-1 px-2 border rounded focus:outline-none focus:ring-2" +
95      " focus:ring-"+this.color+"-"+this.getBorderShade()+
96      " focus:ring-opacity-50";
97  }
98
99  render() {
100    return <button class={this.stringifyClass()} disabled={this.disabled} form={this.form}
101  }

```

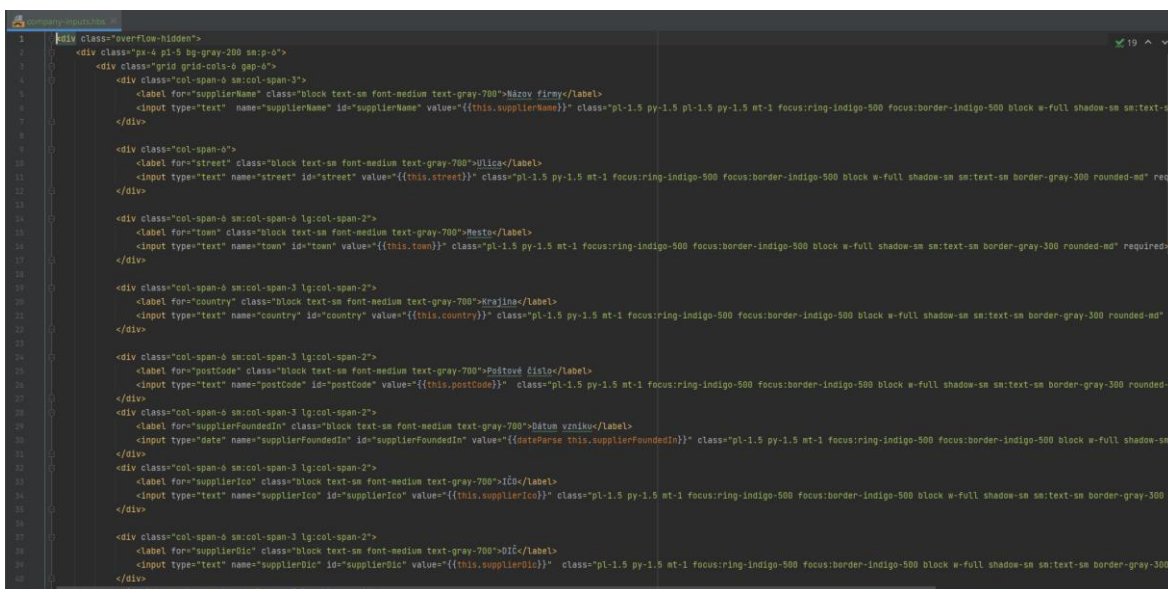
Obrázok 21: Metódy button komponentu



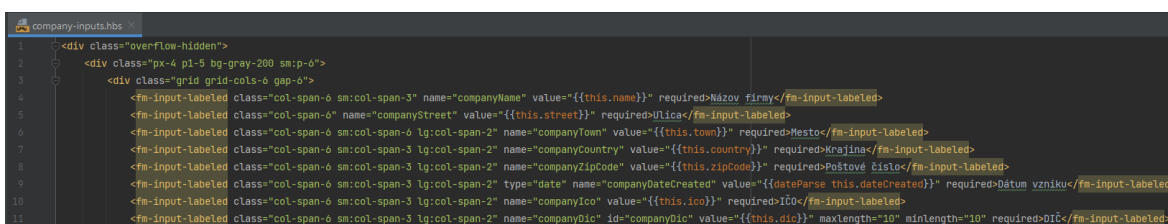
Obrázok 22: Výsledný button komponent

## 4.5.2 Input a label

Ďalším vhodným samostatným komponentom bola kombinácia inputu a label. Pri veľkom množstve využívaných inputov, bol tento krok výhodným už len z dôvodu zlepšenia čitateľnosti kódu. Popis do elementu label vo vnútri komponentu je dopĺňaný pomocou tagu `<slot />`. Jeho funkciou je renderovanie dynamických child elementov vo vnútri komponentov, pre nás slúži primárne pre textové reťazce. Na obrázkoch č. 22 a č. 23 je zobrazené porovnanie využitia elementov oproti komponentov, zobrazené časti kódu majú identickú funkcionálnu a vzhľad na stránke, pri čom na obrázku č. 23 je potrebných takmer štvornásobne menší počet riadkov a je zlepšená čitateľnosť HTML kódu. Tento komponent obsahuje iba atribúty potrebné pre základnú funkcionálnu a vzhľad na stránke.



Obrázok 23: Formulár tvorený elementmi




Obrázok 24: Formulár tvorený komponentmi

## 4.6 Výsledná webová aplikácia

Táto kapitola sa venuje výslednej aplikácii. Opisuje jej funkcionálnu a vzhľad na stránke.

### 4.6.1 Vstup do aplikácie

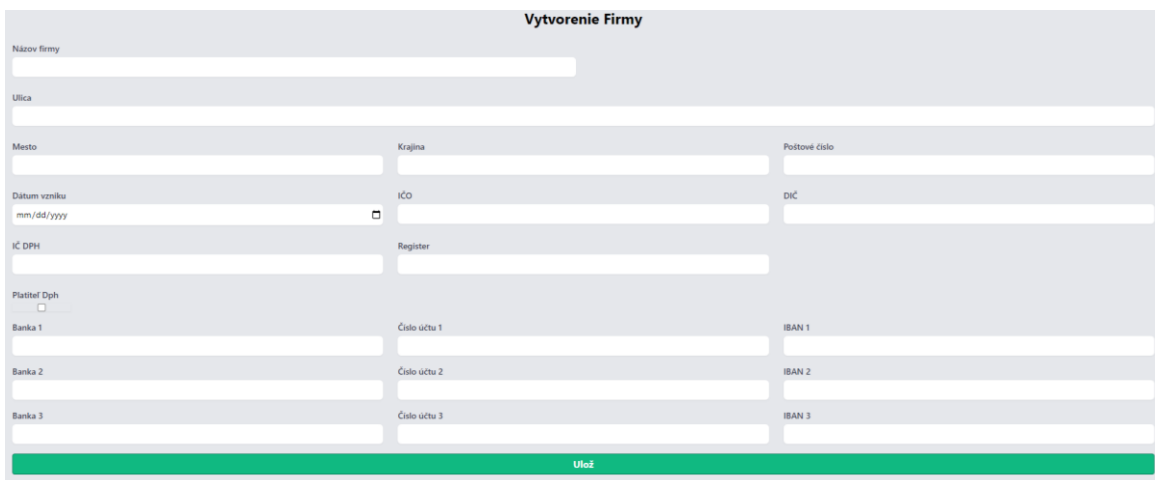
Do aplikácie môžu vstúpiť iba registrovaní používatelia, pokiaľ používateľ nie je registrovaný, môže tak urobiť po kliknutí na link *Chcem sa registrovať*.



The login form is displayed on a light gray background. It features two white input fields for 'Login' and 'Heslo' (password). Below these fields is a link labeled 'Chcem sa registrovať' in a brown, italicized font. At the bottom of the form is a green button with the text 'Prihlásiť' in white.

Obrázok 25: Prihlasovanie do aplikácie

Po registrácii je používateľ nútený zadať informácie o jeho firme, potrebné pre správnu funkciu aplikácie. Pokiaľ používateľ nedokončí vytváranie svojej firmy a bude odhlásený, po opätovnom prihlásení bude znovu vyzvaný, aby vyplnil informácie ohľadom jeho firmy.



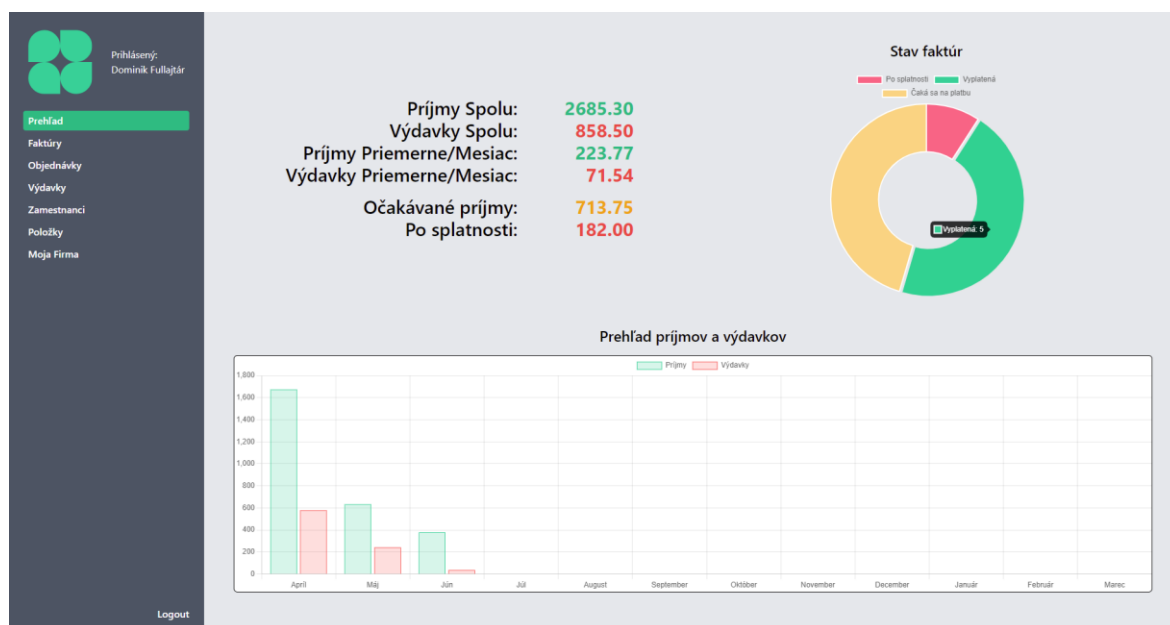
The 'Vytvorenie Firmy' form is a complex layout with multiple input fields. It includes fields for 'Názov firmy', 'Ulica', 'Mesto', 'Krajina', 'Poštové číslo', 'Dátum vzniku' (with a date format 'mm/dd/yyyy'), 'IČO', 'DIČ', 'IČ DPH', 'Register', 'Platiteľ Dph' (with a checkbox), 'Banka 1', 'Číslo účtu 1', 'IBAN 1', 'Banka 2', 'Číslo účtu 2', 'IBAN 2', and 'Banka 3', 'Číslo účtu 3', 'IBAN 3'. A green 'Uložiť' button is located at the bottom right of the form.

Obrázok 26: Vytvorenie firmy

### 4.6.2 Prehľad

Prihlásený používateľ je presmerovaný na stránku *Prehľad*, na tomto mieste môže vidieť všetky príjmy a výdavky za momentálny daňový rok. Tieto príjmy a výdavky sú zobrazené v grafe kde sú rozdelené do mesiacov ale aj ako súčet za celý rok a ich mesačný


priemer. Na stránke je aj zobrazený stav faktúr, pričom tie môžu byť vyplatené po splatnosti, vyplatené načas alebo neuhradené faktúry pri ktorých dátum splatnosti ešte neuplynul. Nachádza sa tu aj položka *Očakávané príjmy*, ktorá predstavuje súčet hodnôt všetkých doposiaľ nevyplatených faktúr.



Obrázok 27: Domovská stránka - Prehľad

### 4.6.3 Faktúry


Stránka *Faktúry* slúži na správu faktúr. Používateľ na tomto mieste môže vidieť podstatné informácie o faktúrach a po kliknutí na jeden z riadkov tabuľky, je presmerovaný na úplné zobrazenie faktúry, ktorú môže podľa potreby ľubovoľne upravovať. Pre jednoduchšie vyhľadanie požadovanej faktúry sa dá použiť input element s názvom Hľadať, zadaný reťazec aplikuje filter, platný na všetky dáta zobrazené v tabuľke. Pre uľahčenie vyhľadávania je možné pridávať faktúram tagy a názvy. Pod tabuľkou je umiestnené tlačidlo *Vytvor Faktúru*, ktoré používateľa presmeruje na vytváranie novej faktúry.

 Prihlásený: Dominik Fullajtár	Zobraz: 20 záznamov							Hľadať:
	Číslo	Dátum vystavenia	Zákazník	Zákazka	Suma (bez DPH)	Vystavil	Tag	Stav
	5/2021	2021-06-08	Kameň		182.00	Fullajtár	Hruška, Jablko, Pomaranč	Po splatnosti
	7/2021	2021-06-08	Gaspar		187.75	Fullajtár		Čaká sa na platbu
	8/2021	2021-06-08	Asimov		182.00	Fullajtár	Maliny	Čaká sa na platbu
	9/2021	2021-06-08	Gaspar		178.25	Fullajtár	Potraviny	Vyplatená
	10/2021	2021-06-08	Ivanovič		182.00	Fullajtár		Čaká sa na platbu
	11/2021	2021-06-08	Fabian		32.00	Fullajtár		Čaká sa na platbu
	1/2021	2021-06-06	Mrkva	Moja prvá zákazka	130.00	Fullajtár	Jablko	Po splatnosti
	6/2021	2021-05-06	Drevo		200.70	Fullajtár		Vyplatená
	4/2021	2021-05-03	Anon		286.55	Fullajtár		Vyplatená
	3/2021	2021-05-03	Gaspar		346.05	Fullajtár		Vyplatená
	2/2021	2021-04-01	Gaspar		1673.75	Fullajtár	Banán	Vyplatená
Záznamy 1 až 11 z celkom 11								Predchádzajúca 1 Nasledujúca
Vytvor Faktúru								

Obrázok 28: Faktúry

Číslo faktúry je automaticky vygenerované na základe roku a počte už vytvorených faktúr v tomto roku. Podľa potreby ho však používateľ môže prepísať. Informácie o dodávateľovi sú vytvorené vopred podľa vyplnených informácií, keď si používateľ vytváral firmu. Informácie o banke sú vopred predvolené, ak si ich používateľ napísal pri vytváraní firmy. V prípade potreby si ho môže ručne vyplniť. Informácie o banke sa zadávajú textovým vstupom, alebo vybratím jednou z troch predvolených bánk, ktoré si môže používateľ zadať pri vytváraní firmy. Stránka podporuje Tlač Faktúry, pričom všetky nežiadúce elementy sú odstránené, a vytlačená bude iba štandardná forma faktúry vo formáte A4. Pod faktúrou je políčko *Dátum vyplatenia*, kde je možné zadať dátum, kedy bola faktúra vyplatená. Pokiaľ sa toto políčko nechá prázdne, tak systém berie túto faktúru ako neuhradenú čo sa v tabuľke faktúr zobrazuje hodnotou *Čaká sa na platbu*.





Prihlásený:  
Dominik Fullajtár

Prehľad

**Faktúry**

Objednávky

Výdavky

Zamestnanci

Položky

Moja Firma

Faktúra číslo: 14/2021

Fullajtár s.r.o.  
Uličná 3/5  
972 51 Handlová  
IČO: 456456  
DIČ: 456456  
Platiteľ DPH: Nie  
IČDPH: 456456  
Register: 456456  
[Zmeniť údaje](#)

ÚČET:

Banka:

IBAN:

Forma úhrady:

Spôsob Dopravy:

Mena:

Variabilný symbol:

Konštantný symbol:

Špecifický symbol:

Odberateľ:

IČO:

DIČ:

IČDPH:

Dátum vystavenia:

Dátum dodania:

Dátum splatnosti:

P. č.	Kód položky	Názov položky	Množstvo	MJ	Cena za j. bez DPH	Cena za j. s DPH	Spolu bez DPH	DPH	DPH %	Zlava v %	Celková cena
1	<input type="text" value="Kód položky"/>	<input type="text" value="Názov"/>	<input type="text" value="Množstvo"/>	<input type="text" value="Jednot"/>	<input type="text" value="0"/>	<input type="text" value="---"/>	<input type="text" value="---"/>	<input type="text" value="---"/>	<input type="text" value="20"/>	<input type="text" value="0"/>	<input type="text" value="---"/>
Spolu:											X

**Pridaj Položku**

Vyhotovil:

Telefón:

Email:

Prevzal:

Suma bez DPH:

Celková fakturovaná suma:

Uhradené zálohami:

K úhrade:

Zákazka

Tag

Dátum vyplatenia

Poznámka:

**Ulož**

Logout

Obrázok 29: Vytvorenie novej faktúry

#### **4.6.4 Objednávky**

Objednávky disponujú takmer rovnakou funkcionalitou ako faktúry, s možnosťou generovania faktúr z objednávok. Vďaka tejto funkcionalite sa všetky informácie predvyplnia, a používateľ musí zadať iba informácie, ktoré objednávka neobsahuje.

#### **4.6.5 Výdavky**

Výdavky sú opäť zobrazené v tabuľke. Pozostáva z názvu, dátumu vzniku, sumy, poznámky a môže byť priložená aj fotografia dokladu.

#### **4.6.6 Zamestnanci**

V tejto časti aplikácie je databáza zamestnancov. Nevyužíva sa v iných časti aplikácie, a má informačnú funkciu pre firmu používateľa.

#### **4.6.7 Položky**

Zobrazuje zoznam všetkých doteraz predaných položiek spolu s ich cenami. Firma používa tento zoznam, aby vedela, za akú sumu má rovnaký produkt predat' ďalším zákazníkom. Má informačnú funkciu.

#### **4.6.8 Moja firma**

Tu sa môžu upravovať informácie o firme, ktoré boli zadané pri vytváraní firmy. Keďže takéto niečo by sa za normálnych okolností nemalo stať, slúži primárne na zmenu bankových účtov firmy, ktoré sú dopĺňané do objednávok a faktúr. Používateľ si môže zadať hodnoty troch bankových účtov.

# Záver

Cieľom tejto práce bolo oboznámiť sa so štandardom Web Components, rôznymi dostupnými knižnicami a prístupmi pri ich vývoji. Následne ich využitím vo webovej aplikácii. Taktiež sme sa museli oboznámiť s právnou stránkou ohľadne objednávkových a fakturačných dokladov. Všetky tieto poznatky sme následne museli aplikovať pri vývoji webovej aplikácie. Väčšinu našich cieľov sa nám podarilo splniť.

Začiatkom riešenia bola analýza dostupných riešení, ďalším krokom bol návrh funkcionality nášho riešenia. Tomu nasledovala implementácia REST API našej aplikácie. Počas tejto časti sme sa zoznámili s frameworkom NestJS, HTTP serverovým frameworkom Express a ich možnosťami pri vývoji aplikácií na strane servera. Tiež sme sa lepšie oboznámili s funkcionalitou TypeScript a objektovo-relačným mapovaním. Po dokončení API, sme sa posunuli na tvorbu klientskej časti aplikácie, kde sme sa podrobnejšie venovali často využívaným prvkom aplikácie a ich implementovaní do komponentov. Vytvárali sme ich za pomoci StencilJS, ktorý nám dopomohol vytvoriť balíček efektívnych komponentov, kompatibilných so všetkými frameworkami podporujúcimi JavaScript.

Výsledkom našej práce je fakturačný systém, jeho funkcionality spĺňajú takmer všetky funkcionálne požiadavky ktoré sme si určili na začiatku práce. Terajší stav aplikácie by bol nevhodný na nasadenie do produkcie, z dôvodu nedostatočnej funkcionality oproti niektorým konkurenčným riešeniam. Napriek tomu, by sa pravdepodobne v budúcnosti dal využiť po implementácii nových funkcionalít. Niektoré z vhodných pridaných možností by bol prístup viacerých používateľov k jednej firme a implementácia rolí pre zamestnancov. Alebo prídanie počtu inventárnych položiek v sklade, ktorých počet by sa znižoval po vytvorení faktúry s pridaním upozornenia pokiaľ by nebol dostatočný počet položiek v sklade.

## 5 Zoznam použitej literatúry

1. Podnikateľské centrum. 2021. *Čo je to faktúra | Podnikanie* | Podnikateľské centrum.sk. [online] Dostupné z: <<https://podnikatelskecentrum.sk/co-je-to-faktura/>> [cit. 2020-12-15].
2. NestJS - A progressive Node.js framework. 2021. *NestJS - A progressive Node.js framework*. [online] Dostupné z: <<https://nestjs.com/>> [cit. 2020-12-15].
3. Typeorm.io. 2021. *TypeORM - Amazing ORM for TypeScript and JavaScript (ES7, ES6, ES5). Supports MySQL, PostgreSQL, MariaDB, SQLite, MS SQL Server, Oracle, WebSQL databases. Works in NodeJS, Browser, Ionic, Cordova and Electron platforms.* [online] Dostupné z: <<https://typeorm.io/>> [cit. 2021-01-09].
4. **FOWLER, Martin.** *Patterns of Enterprise Application Architecture*. [s.l.]: Addison Wesley, 2002. S. 560. ISBN: 0-321-12742-0.
5. Auth0 - Blog. 2021. *A Brief History of JavaScript*. [online] Dostupné z: <<https://auth0.com/blog/a-brief-history-of-javascript/>> [cit. 2020-12-21].
6. Springboard Blog. 2021. *The History of JavaScript: Everything You Need to Know - Springboard Blog*. [online] Dostupné z: <<https://www.springboard.com/blog/data-science/history-of-javascript/>> [cit. 2020-12-21].
7. The State of the Octoverse. 2021. *The State of the Octoverse*. [online] Dostupné z: <<https://octoverse.github.com/>> [cit. 2021-01-11].
8. Developer.mozilla.org. 2021. *JavaScript First Steps - Learn web development | MDN*. [online] Dostupné z: <[https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First\\_steps](https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps)> [cit. 2020-12-27].
9. Html.spec.whatwg.org. 2021. *HTML Standard*. [online] Dostupné z: <<https://html.spec.whatwg.org/multipage/custom-elements.html>> [cit. 2021-02-06].
10. Typescriptlang.org. 2021. [online] Dostupné z: <<https://www.typescriptlang.org/>> [cit. 2021-02-09].
11. **Burbeck, Steve** (1992) *Applications Programming in Smalltalk-80:How to use Model–View–Controller (MVC)*
12. freeCodeCamp.org. 2021. *How the Model View Controller Architecture Works – MVC Explained*. [online] Dostupné z: <<https://www.freecodecamp.org/news/model-view-architecture/>> [cit. 2021-03-01].

13. npm. 2021. *express-handlebars*. [online] Dostupné z:  
<<https://www.npmjs.com/package/express-handlebars/v/3.0.0>> [cit. 2020-12-18].
14. Handlebarsjs.com. 2021. *Handlebars*. [online] Dostupné z:  
<<https://handlebarsjs.com/>> [cit. 2020-12-23].
15. Expressjs.com. 2021. *Using template engines with Express*. [online] Dostupné z:  
<<https://expressjs.com/en/guide/using-template-engines.html>> [cit. 2021-02-01].
16. Besant Technologies. 2021. *What is Express.js? / Why should use Express.js? / Features of Express.js*. [online] Dostupné z:  
<<https://www.besanttechnologies.com/what-is-expressjs>> [cit. 2021-01-28].
17. Tailwindcss.com. 2021. *Tailwind CSS - Rapidly build modern websites without ever leaving your HTML.* [online] Dostupné z: <<https://tailwindcss.com/>> [cit. 2021-05-05].
18. Stenciljs.com. 2021. *Stencil*. [online] Dostupné z: <<https://stenciljs.com/>> [cit. 2021-03-19].
19. KROS. 2021. *Všetko, čo potrebujete vedieť o faktúrach | KROS*. [online] Dostupné z: <<https://www.kros.sk/blog/vsetko-co-potrebujete-vediet-o-fakturach/>> [cit. 2021-04-14].
20. Slov-lex. 2021. 222/2004 Z.z. - *Zákon o dani z pridanej hodnoty - SLOV-LEX*. [online] Dostupné z: <<https://www.slov-lex.sk/pravne-predpisy/SK/ZZ/2004/222/20180101#paragraf-76>> [cit. 2021-05-17].
21. Slov-lex. 2021. 102/2014 Z.z. - *Zákon o ochrane spotrebiteľa pri pr...* - SLOV-LEX. [online] Dostupné z: <<https://www.slov-lex.sk/pravne-predpisy/SK/ZZ/2014/102/20190101#paragraf-3>> [cit. 2021-05-17].
22. **Stark, E.**, 2006. *Tvorba virtuálneho laboratória s využitím JavaScript-u na strane servera*. Diplomová práca. Slovenská technická univ. v Bratislave FEI ÚIM (FEI).
23. Developer.mozilla.org. 2021. *Using shadow DOM - Web Components | MDN*. [online] Dostupné z: <[https://developer.mozilla.org/en-US/docs/Web/Web\\_Components/Using\\_shadow\\_DOM](https://developer.mozilla.org/en-US/docs/Web/Web_Components/Using_shadow_DOM)> [cit. 2021-06-10].
24. **Bell, J., Magolan, G., Housley, P., de Peretti, A. and Guijarro, D.**, 2018. *Nest.js: A Progressive Node.js Framework*, s.8.
25. **CHIARELLI, A.**, 2020. *EXPLORING WEB COMPONENTS*. [S.l.]: BPB PUBLICATIONS, s.1. ISBN: 9789389423976

# Prílohy

Príloha A: Štruktúra elektronického nosiča .....	II
--	----

## **Príloha A: Štruktúra elektronického nosiča**

\bakalarska\_praca.pdf

\projekt.zip

\navod\_na\_spustenie.txt