Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Návrh a implementace systému pro tvorbu rozpočtu elektrikářských prací

Václav Branda

Diplomová práce

2020

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2019/2020

ZADÁNÍ DIPLOMOVÉ PRÁCE
(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Václav Branda**
Osobní číslo:	**I18235**
Studijní program:	**N2646 Informační technologie**
Studijní obor:	**Informační technologie**
Téma práce:	**Návrh a implementace systému pro tvorbu rozpočtu elektrikářských prací**
Zadávající katedra:	**Katedra softwarových technologií**

Zásady pro vypracování

V teoretické části práce budou popsány současné trendy a nástroje v oblasti plánování rozpočtů. Dále pak veškeré technologie, které budou finálně pro realizace práce vybrány. V rámci vlastní implementace bude třeba navrhnout vhodný databázový model a navrhnout samotný software rozpočtu elektrikářských prací s využitím frameworku JAVA Spring a technologie Vaadin. Systém musí být řešen i z pohledu různých uživatelských rolí.

Rozsah pracovní zprávy: **50-60 stran**
Rozsah grafických prací: **–**
Forma zpracování diplomové práce: **tištěná**

Seznam doporučené literatury:

CARNELL, John. Spring microservices in action: a multiplatform approach to building chatbots. Shelter, Island, NY: Manning Publications Co., 2017. ISBN 16-172-9398-9.
CARNELL, John. Beginning spring boot 2: applications and microservices with the spring framework. New York, NY: Springer Science Business Media, 2017. ISBN 978-148-4229-309.

Vedoucí diplomové práce: **Ing. Jan Fikejz, Ph.D.**
Katedra softwarových technologií

Datum zadání diplomové práce: **5. listopadu 2019**
Termín odevzdání diplomové práce: **15. května 2020**

Univerzita Pardubice Česká republika
L.S.

Ing. Zdeněk Němec, Ph.D.
děkan

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

V Pardubicích dne 15. listopadu 2019

Prohlašuji:

Tuto práci 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., autorský zákon, 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. 9/2012, bude práce zveřejněna v Univerzitní knihovně a prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 26. 8. 2020

Václav Branda

PODĚKOVÁNÍ

Své poděkování bych chtěl věnovat panu Ing. Janu Fikejzovi, Ph.D. za odborné vedení této práce a hlavně za poskytnutí příležitosti tuto práci dokončit i přes časový deficit způsobený zraněním.

ANOTACE

V této Diplomové práci je popsán vznik aplikace pro tvorbu rozpočtů elektrikářských prací. V práci jsou také uvedeny i důvody jejího vzniku, její uplatnění a případné již existující alternativy. Diplomová práce se tak dělí na dvě části. Začátek práce obsahuje samotné důvody tvorby této aplikace. Dále je zde provedena zevrubná analýza již existujících systémů a to nejen v oblasti elektrikářských prací ale i v dalších odvětvích, kde je používání rozpočtových aplikací nutností. Druhá část práce se pak zabývá samotnou tvorbou aplikace. Od specifikace požadavků, výběr používaných technologií přes návrh databázového modelu a UI diagramu, po samotnou implementaci a následné testování s případným nasazením.

KLÍČOVÁ SLOVA

Rozpočet, ceník, nabídka, elektrikář, webová aplikace

TITLE

Design and implementation of a system for budgeting electrical work

ANNOTATION

This diploma dissertation describes creation of an application for creating budgets for electrical work. In this paper are stated reasons for its creation, its usage and eventual existing alternatives. This thesis is presented in two parts. First one is describing reasons why this application was created with analysis of existing systems which are not only for electrical work but in other industries where is usage of budget applications needed. Next part is about creation standalone application. It goes through specification of requirements, choosing used technologies, designing a database model and an UI diagram, to the implementation itself and testing code with potential deployment.

KEYWORDS

Budget, price list, offer, electrician, web application

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM ZKRATEK

CSS	Cascading Style Sheets
HTML	HyperText Markup Language
JSON	JavaScript Object Notation
OS	Operační systém
XML	eXtensible Markup Language

ÚVOD

Nejenom u nás ale i po celím světě existuje mnoho firem, které poskytují své služby v oblasti elektrikářských prací. Bohužel ne všechny tyto firmy používají nějaký sofistikovaný softwarový nástroj k tvorbě a správě nabídek. A právě z tohoto důvodu vznikla tato práce. Jejím cílem je navrhnout a vytvořit softwarové řešení, které zjednoduší práci všem koncovým uživatelům, ať se již jedná o firmu či jednotlivce. Nezáleží na tom, zda jste malá firma o deseti zaměstnancích, nebo nadnárodní společnost. Vždy budete hledat způsob, jak proces tvorby nabídek zjednodušit, zrychlit a co nejvíce optimalizovat. A k tomu by vám měla sloužit právě tato aplikace.

Obdobných aplikací je sice na trhu celá řada, jejich kvalita je však značně variabilní. Předností nové aplikace oproti konkurenci by mělo být jednoduché a intuitivní rozhraní, které si běžný uživatel osvojí i bez nutnosti speciálního zaškolení. Proto bude věnován velký prostor jak analýze, tak následně vlastnímu návrhu aplikace. Kvalita a efektivita je rovněž podpořena využitím moderních technologií a vývojových nástrojů, které jsou už na tvorbu takovýchto systému připraveny. Součástí aplikace by také měla být obsáhlá databáze prací a úkonů, které právě elektrikáři provozují. Hlavním přínosem databáze široké škály elektrikářských aktivit je to, že každá aktivita je doplněna o normu času. Báze dat by měla být samozřejmě plně upravitelná a každý zákazník by si ji měl mít možnost upravit podle vlastních individuálních potřeb.

Představení aplikace a její samotná tvorba jsou však jen částí této práce. V teoretické části práce budou představeny vybrané softwarové nástroje pro tvorbu rozpočtů, a to nejen z elektrikářské oblasti. Bude zde uvedeno, jak fungují, kde je lze využít a jaké jsou na ně kladeny požadavky při jejich používání. Dále zde bude porovnáno několik vybraných zástupců aplikací, které by mohly nově vytvořenému systému konkurovat. Budou zde kompletně představeny a probrány jejich klady a zápory. Tento přístup se využije také při samotném vývoji, kde již bude známo na které funkcionality klást větší důraz a čeho je potřeba se vyvarovat.

V závěru práce bude vyhodnocena celková tvorba systému. Dojde k vyhodnocení, jestli výsledná aplikace odpovídá požadavkům a zda naplnila plánované očekávání.

1 SOFTWAROVÉ NÁSTROJE PRO TVORBU ROZPOČTŮ

Jak již název této práce napovídá, bude se její obsah zaobírat z valné části softwarem pro tvorbu rozpočtů. Než zde, ale bude představena samotná tvorba nové aplikace, bylo by asi dobré si tuto oblast nejdříve trochu přiblížit. Nejdříve zde tedy budou tyto softwarové nástroje alespoň trochu představeny a bude zde řečeno, k čemu se využívají, v jakých oborech jsou rozšířené nejvíce a jaká je jejich cílová skupina uživatelů. Bude zde také představen vznik těchto systému a jejich postupný vývoj až do dnes. S tím také souvisí aktuální moderní trendy v jejich tvorbě a designu, které budou představeny na závěr této kapitoly.

1.1 Výhody softwarových nástrojů

Softwarové nástroje pro tvorbu rozpočtů jsou obvykle počítačové programy, které mají za úkol uživateli proces tvorby rozpočtů usnadnit. Je zde několik argumentů, které právě tento elektronický způsob tvorby vyzdvihují. Naproti tomu jsou tu však jedinci, kteří na klasický přístup tužka papír nedají dopustit. Jedná se však jen o výjimky, které do několika let vymizí. Je tu hned několik důvodů, proč jsou softwarové nástroje tak oblíbené. Tím největším důvodem je asi jejich jednoduchost. Odpadá zde totiž potřeba skladování dat v tištěné podobě, s tím související jejich organizace a hlavně zdlouhavé hledání potřebných informací. Pokud jsou data uložena elektronicky, vyhledávání v nich je velmi jednoduché a tvorbu rozpočtu to tak značně urychlí.

Dalším plusem softwarových nástrojů je tvorba tiskových sestav. Ty jsou zde obvykle dovedeny k dokonalosti a nabízí tak uživateli výborný výstup, ve kterém jsou informace jednoduše a přehledně zobrazeny v grafiky vyladěné šabloně. Takovýto výstup by bylo velice složité, možná i nemožné udělat ručně. S takovouto kvalitou výsledného dokumentu už nic nebrání uživateli předat takovouto nabídku rovnou svému zákazníkovi, bez nutnosti dalších úprav.

Důležitým aspektem při volbě softwarového nástroje pro tvorbu rozpočtů je to, zda obsahuje interní ceník všech úkonů a prací, které by uživatel mohl potřebovat. V dnešní době, kdy jsou informace nedocenitelné, je přítomnost takovéto rozsáhlé databáze, dá se říct nutností. Uživatelé si tak nemusí vytvářet ceníky vlastní a oceňovat tak každou položku ze své nabídky. Stačí jim využít poskytnutý ceník, který je obvykle možno si upravit k obrazu svému. Vydavatelé softwaru si však za takovýto luxus nechávají zaplatit a za tyto ceníky dodávané společně se softwarem si uživatel připlatí nemalé peníze. V některých případech se jedná

dokonce i o službu, kdy uživatel platí měsíční či roční poplatky za to, že vydavatel databázi ceníku pravidelně aktualizuje. [5]

1.2 Oblast využití a cílová skupina uživatelů

V době, kdy jsou informační technologie tak rozšířené, a hlavně dostupné je jasné, že používaní softwaru v oblasti tvorby rozpočtů a nabídek se nevyhne žádnému oboru práce. Nicméně ne pro všechny z nich je to tak úplně výhodné. Každý podnikatel, který v dnešní potřeboval stanovit cenu svých služeb, na to použil nějaký softwarový nástroj. Ne vždy se však jednalo o nějaký ze sofistikovaných nástrojů, o kterých je zde řeč. To ale nemusí být na škodu, některé z oborů ke stanovení ceny svých služeb takovéto nástroje nepotřebují a stačí jim jen jednoduchý tabulkový editor. Jedná se většinou o obory, kde jsou ceny za danou službu statické, často se nemění a nejsou závislé na délce práce či jiné měrné jednotce. Příkladem může být například holičství nebo nějaké ubytovací zařízení.

Existují ale obory, ve kterých je použití nějakého specializovaného nástroje na tvorbu nabídek a rozpočtů takřka nutností. Hlavním zástupcem je obor stavebních prací. V tomto oboru je nejdříve potřeba vyhotovit rozpočet prací, který je následně předán zákazníkovi ke schválení. Samotné práce jsou realizovány až ve chvíli, kdy zákazník zná přibližnou cenu práce a nabídku odsouhlasí. Ceník stavebních prací také obvykle není malý a stanovit tak cenu zakázky někdy bývá pěkný oříšek. Proto využití sofistikovaného softwaru je v tomto oboru docela zásadní.

Stavební průmysl ale není jediným oborem, ve kterém se tyto nástroje využívají. Existuje jistě mnoho oborů, ve kterých je zapotřebí nejdříve pro danou zakázku vytvořit nabídku práce. Obecně však záleží na daném podnikateli, zda zvolí nějaký specializovaný software nebo ne. Pro malé podnikatele, kteří netvoří obrovské nabídky každý den, je to možná jen drahý komfort, který nevyužijí. Na druhou stranu je to obrovský šetřič času, bez kterého by třeba velké firmy nedokázaly fungovat.[5]

1.3 Historie a moderní trendy

Softwarové nástroje pro tvorbu rozpočtů tu nebyly odjakživa. Na přelomu tisíciletí, kdy došlo k velkému rozmachu stavebních technologií, se začaly objevovat první známky jejich využití. Nejdříve se však jednalo pouze o informační technologie obecně, které se začaly používat v řemeslných oborech pro plánování, řízení a sledování ekonomické situace firem. Postupným vývojem se dostaly i do oblasti ekonomické, kde byl už jenom krůček k programům pro tvorbu

rozpočtů a kalkulací. Ty časem vytlačily klasické způsoby rozpočtování pomocí katalogu, tužky a kalkulačky.

Společně s tím, jak se vyvíjely informační technologie, tak se analogicky měnila i kvalita softwarů pro tvorbu rozpočtů. Starší programy byly obvykle tvořeny pro platformu MS-DOS. Aktuálně ale většina rozpočtových systému běží na operačním systému Windows. Tato platforma nabízí velkou variabilitu a dává vývojářům značnou volnost při tvorbě programů. Je tedy možné najít značné rozpětí funkcionalit těchto programů. Může tak být řeč o jednoduchých programech typu tabulkový editor, přes programy složitější využívající databázové úložiště, až po komplexní systémy komunikující s ostatními aplikacemi, které řídí provoz dané firmy.

Nicméně pokrok jde dál a aktuální moderní trendy se zabývají sdílením informací a komunikace přes internet. Proto je kladen velký důraz na to, aby data nebyla uložena na jednom zařízení, ale aby k nim byl přístup přes síť, ať už jde o internet nebo jen síť firemní. Nejen v oblasti rozpočtování ale i v jiných oborech se začíná také postupně ustupovat od desktopových aplikací a pomalu se přechází na jejich webové varianty. Čímž je ušetřena hlavně cena za distribuci aplikace, která tak nemusí běžet na každém zařízení, ale je dostupná online. [5]

2 REŠERŠE EXISTUJÍCÍCH SOFTWARŮ

Tato práce se kromě teoretické stránky věci zaměřuje zejména na tvorbu nové aplikace pro tvorbu rozpočtů. Nejedná ale bohužel o nějaký nový nápad a podobných aplikací už na trhu několik je. Tím pádem je potřeba před samotnou tvorbou programu prověřit konkurenci a provést rešerši již dostupných řešení. Je zde vybráno několik aplikací, které by mohli být přímými konkurenty nově vznikajícího programu. Proto tyto aplikace budou prozkoumány a navzájem porovnány v několika hodnocených parametrech. Mezi ně patří nabízené možnosti, přítomnost ceníku, uživatelské rozhraní, a nakonec cenová politika. Důvodů takovéhoto průzkumu trhu je několik. Největším z nich je asi snaha zjistit, zda už neexistuje podobná aplikace, která by mnohonásobně převyšovala plánovanou aplikaci ve všech ohledech. Tím pádem by byl projekt zbytečný a skončil by dříve, než by začal. Dalším důvodem je snaha o nalezení problémů a chyb ostatních aplikací, kterým by se dalo předcházet. Výsledná aplikace by tak byla v něčem lepší než konkurence a měla by tak větší šanci na úspěch.

2.1 KONCES

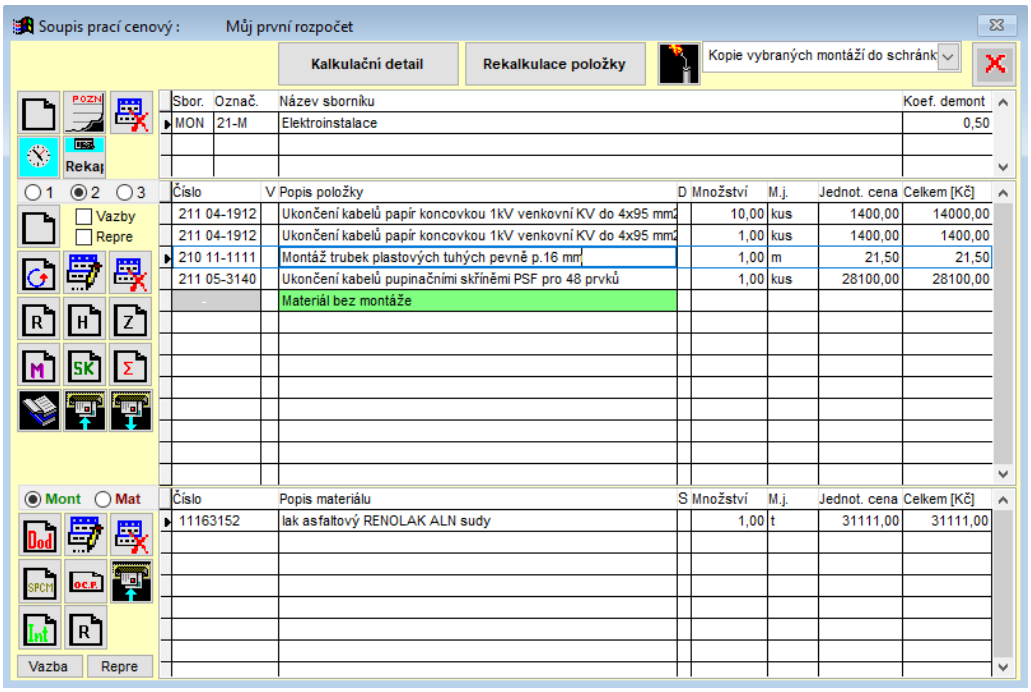
První z vybraných aplikací porovnávaných v této práci je program KONCES pro rozpočty a kalkulace. Jak říká jeho instalační příručka, tento program je určen projektantům investorů a montážním firmám pro zpracování cenových nabídek. Tento program je poměrně starý a jeho podpora skončila zřejmě někde u operačního systému Windows 7. Nicméně na svých webových stránkách je stále dostupná jeho objednávka, spolu s demoverzí, která byla využita právě při tvorbě této rešerše. A ačkoli podpora tohoto programu skončila u předminulé verze Windows, demoverze na aktuálním operačním systému běží bez problémů. [6]

I přes to, že je KONCES starým nástrojem, nabízí celou řadu funkcionalit, které byste od rozpočtového programu čekaly. Jsou zde tvorby rozpočtů, které je možno archivovat v aplikaci nebo exportovat do souboru. Dále jsou zde tvorby zakázek, u kterých je možno vyplnit mnoho polí a danou zakázku tak dokonale popsat a vložit ji tak do systému. K zakázkám je také možno přiložit několik dokumentů, takže je zde možnost k zakázce připojit i nabídky vytvořené zde v systému. Pokud je potřeba dostat data ze systému, aplikace obsahuje tiskové sestavy, které lze rovnou vytisknout nebo uložit do textového dokumentu. Jelikož ale byla zkoušena jen demoverze samotného programu, nebyly tak otestovány všechny funkce a hodnocení může být tak zkreslené.

To, v čem KONCES vyniká je přítomnost rozsáhlého ceníku prací a materiálů, za který by se nemusel stydět žádný rozsáhlý sofistikovaný program. Obsahem ceníku ale nejsou jen práce a materiály z elektrikářského oboru. Zastoupení zde má většina oborů týkající se oblasti stavebnictví. Dále kromě ceníku aplikace obsahuje číselníky, které se uživateli mohou hodit při tvorbě nabídek. Jedná se například o regionální číselník krajů a obcí, nebo o číselník bank a jejich předčíslí účtů. Ceníky a číselníky jsou plně modifikovatelné, takže si je uživatel může s libostí upravovat. Problémem s těmito ceníky je ale jejich aktualizace, kterou společnost nevydala už několik let.

Tam kde aplikace KONCES ztrácí oproti konkurenci je uživatelské rozhraní, jehož ukázka je zobrazena na obrázku 1. Zde se již zub času projevil a aplikace tak vypadá, jako kdyby byla určena pro operační systémy minulého tisíciletí. Kdyby se ale jednalo jen o problémy grafického designu, tak by to bylo ještě přijatelné. Zde jsou ale i problémy, které narušují uživatelský prožitek z používání celé aplikace. Příkladem můžou být, opakující se obrázek pozadí, fixní pozice a velikost oken, nerozeznatelné obrázky ikon a v neposlední řadě neintuitivní rozmístění navigačních prvků. V době, kdy byla aplikace vydána, se jednalo nejspíše velice dobrý program. Nicméně jak už bylo řečeno, zub času se na ní podepsal a aktuálním standardům moderních aplikací nestačí.

Posledním aspektem zkoumání zde bude cena. Program KONCES je dodáván na bázi předplatného. To trvá jeden rok a uživatele v základu stojí 3600 Kč. Nicméně objednávka tohoto softwaru lze ještě rozšířit o přistup více uživatelů a doplňující databázi ceníků. Za plnou verzi tohoto softwaru tak uživatel zaplatí přes 6000 Kč ročně. [7]



Obrázek 1: Ukázka programu KONCES. Zdroj: [7]

2.2 OCEP

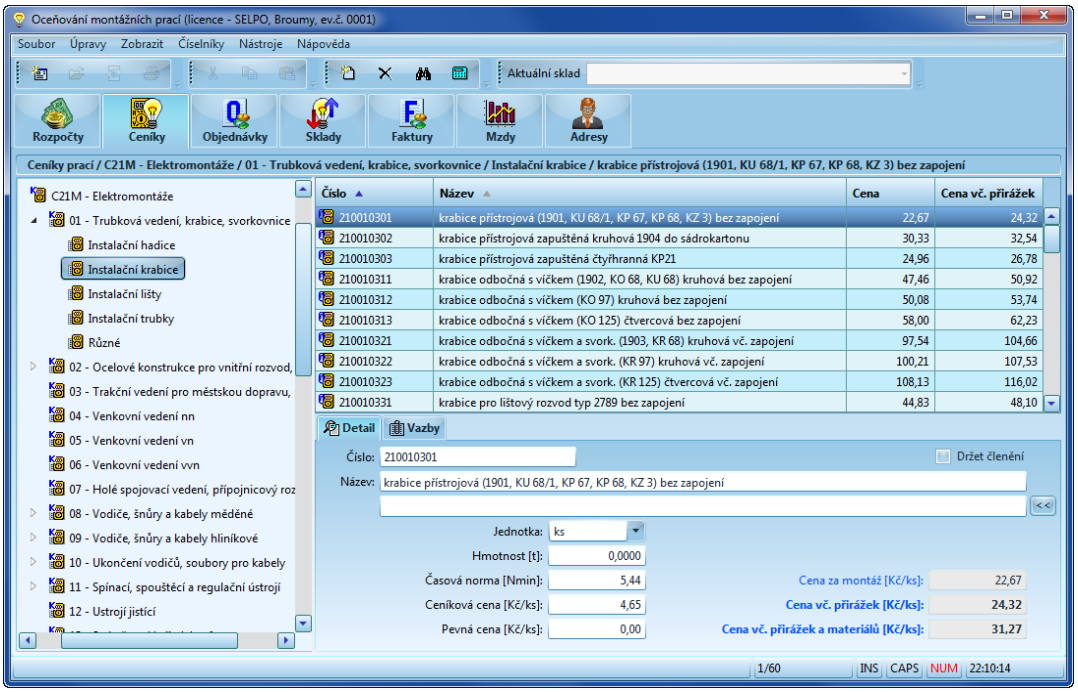
Druhým ze zkoumaných programů je program od společnosti OCEP. Tato společnost podobně jako firma KONCES poskytuje demoverzi svého programu. Nicméně tato demoverze není volně dostupná, a společnost tuto verzi poskytuje na základně odeslaného emailu se žádostí. Po odeslání této žádosti však doposud nepřišla žádná odpověď a demoverze poskytnuta také nebyla. Jako podklady pro tuto rešerši tak budou použity jen informace z webových stránek společnosti a nalezené obrázky či recenze samotné aplikace.

Program OCEP nabízí v základu stejnou funkcionalitu jako předchozí aplikace. To znamená torbu kalkulací, rozpočtů a cenových nabídek. Oproti konkurentům se však zaměřuje jen na obory elektro a voda, topení, plyn. To programu umožňuje se více zaměřit na tyto obory a nabídnout mnohem více specifické funkce, než dokáže konkurence. Tak jako v programu KONCES tak i zde je možno tvořit nabídky a cenové kalkulace, které lze ukládat nebo exportovat. Zde je možnost exportu do XLS souboru, nebo rovnou nabídku vytisknout pomocí tiskové sestavy. Kromě tvorby nabídek společnost nabízí i rozšiřující moduly pro fakturace, objednávky, sklady a mnoho dalšího. Tím se pak stane z aplikace velice robustní a nedocenitelný nástroj pro vedení firmy. [8]

Společnost OCEP stejně jako předchozí firma nabízí ceník prací a materiálů. Na rozdíl od nich ale tento ceník sama udržuje a pravidelně ve spolupráci s projektanty a montážními firmami tento ceník aktualizují. Navíc tento ceník rozšířily o nové položky z oblasti montáže slaboproudých zařízení. Tím pádem zákazník dostane asi ten nejlepší ceník na našem trhu. Vše je samozřejmě plně editovatelné a uživatel si tak může ceník upravit nebo jednoduše naimportovat svůj vlastní. [8]

Jelikož není k dispozici alespoň demoverze aplikace, tak uživatelské rozhraní nemůže být moc posuzováno. Podle ukázky aplikace na obrázku 2, lze ale s určitostí říct, že se jedná o lepší grafický design než u aplikace předchozí. Všechno zde vypadá přehledně a jednoduše. Nicméně aplikace zřejmě obsahuje až moc funkcionalit, z čehož vyplívá, že nějaký malí kurz pro obsluhující personál by nebyl na škodu.[8]

Společnost OCEP nabízí svoji aplikaci také s podobnou cenovou politikou. Aplikace je tak prodávána jako služba na jeden rok. Cena ročního předplatného se určuje podle zakoupené verze a připojených doplňujících modulů. Minimální verze bez doplňujících modulů stojí bez mála 5000 Kč na rok. Cena neomezené verze se všemi moduly se vyšplhá až k 50000 Kč za rok. [8][9]



Obrázek 2: Ukázka programu OCEP. Zdroj: [8]

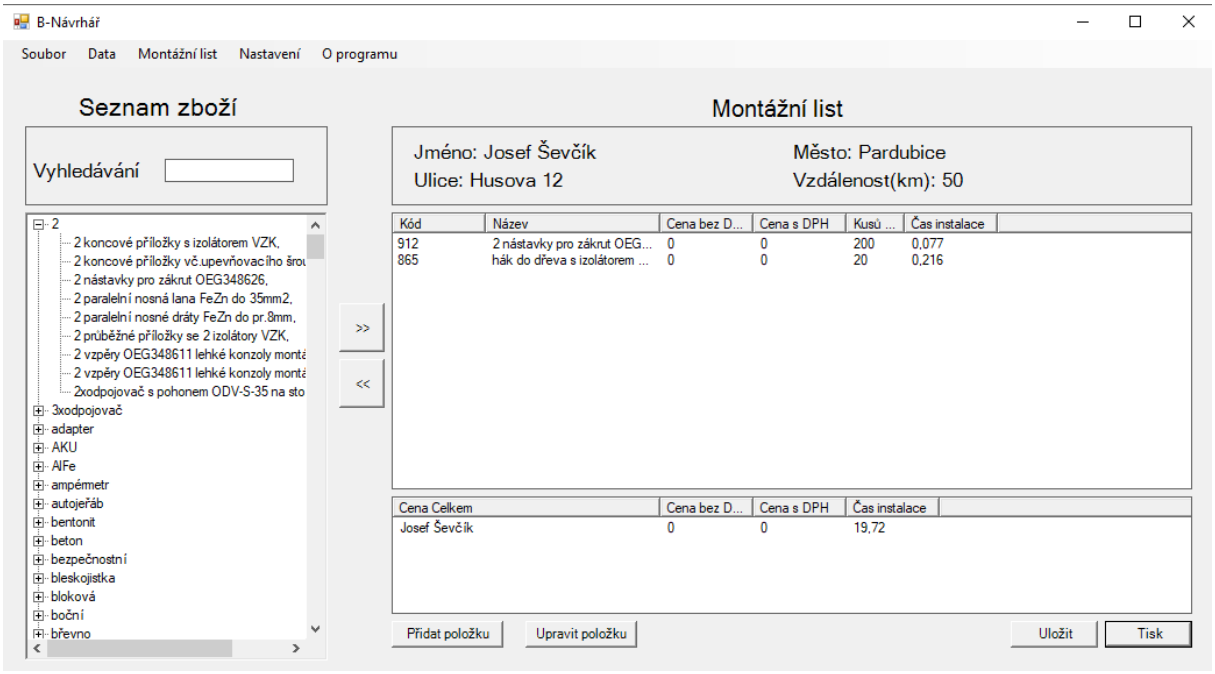
2.3 Aplikace ze vzorové práce

Kromě této práce byla již na Univerzitě Pardubice vytvořena práce podobná. Jejím výsledkem byla shodou okolností rovněž aplikace pro tvorbu rozpočtů a cenových kalkulací. Z tohoto důvodu zde bude prozkoumán i tento software. Díky zveřejňování vypracovaných akademických prací nebude problém otestovat plnou verzi programu. Průzkum tak nebude založen jen na popisu aplikace od samotných vydavatelů, ale půjde o zevrubný test všech funkcionalit.

Při průzkumu samotné aplikace, která je zobrazena na obrázku 3, je vidět, že velkým počtem funkcionalit neoplývá. Ovládá klasikou tvorbu nabídky, kterou obsahovali i předchozí aplikace. Jednotlivé nabídky jde sice exportovat do souboru, nicméně jejich seznam si aplikace nedrží, a pokud chce uživatel nějakou nabídku znova editovat, musí si ji do aplikace nahrát opět nahrát. Měnit u nabídky jde ale jen seznam položek. Změna zákazníka nebo jeho adresy zde chybí. Dalším mínusem aplikace je absence souhrnu a ceny nabídky. Celková cena je zobrazena až v tiskové sestavě, kterou aplikace také obsahuje. Tento výstup by si ale zasloužil po grafické stránce doladit. [10]

S aplikací je dodáván také ceník prací a materiálu. Ten je ale potřeba nejdříve naimportovat ručně do aplikace. Položky ceníku jde sice v aplikaci přidávat a odebírat. Nicméně jejich editace

zde asi opět chybí. Položek v ceníku je zde ale opravdu dost, což bude asi největší klad aplikace. Co se týče ceny, tak zde nemůže být aplikace porovnávána s konkurencí, jedná se totiž o akademický projekt, který zřejmě ještě nebyl nasazen v praxi a nebyla tak stanovena jeho cena.



Obrázek 3: Ukázka aplikace ze vzorové práce. Zdroj: [10]

2.4 Souhrn a porovnání

Pokud by měly být porovnány tyto vybrané aplikace a měl by z nich vyvstat jeden vítěz, zřejmě by to byla aplikace OCEP. Jednoznačně převyšuje ostatní dva konkurenty ve všech směrech. V rámci funkcionality či uživatelského rozhraní, nabízí mnohem více kvalitnějších služeb. Pokud jde o ceník prací a materiálu, tak jako jediná společnost poskytuje aktuální ceník s pravidelnými aktualizacemi. Nakonec je tady i grafická stránka, ve které aplikace OCEP také vede, nicméně má zde asi co zlepšovat.

O druhou pozici by se asi podělili obě zbývající aplikace. Každá má své mínusy a plusy. Kvalitativně jsou však asi na stejné úrovni. Aplikace KONCES má asi rozsáhlejší stránku funkcionalit. Nicméně se na ní podepsal zub času a grafické zpracování už není, co bývalo v době jejího vydání. Poslední aplikace zase pokulhává s funkcionalitami a také s uživatelským rozhraním. Na druhou stranu má lepší vizuální kabát než aplikace KONCES, nicméně ani klasický formulářový styl aplikace nikoho neohromí.

3 NÁVRH A IMPLEMENTACE WEBOVÉ APLIAKCE

Po tom, co byla probrána taková teoretičtější část této práce, kde byly představeny aktuální možnosti v tomto oboru, může být přistoupeno k samotné tvorbě nové webové aplikace. Ta by měla přebírat všechny kladné aspekty již existujících aplikací a vyvarovat se chyb, ať už jde o chyby nové nebo o chyby, které se vyskytují u konkurence.

Postup práce by měl být shodný s postupem práce u nějakého profesionálního projektu, na kterém pracuje celý tým složeny z vývojářů, analytiků a v neposlední řadě testerů. Rozdíl je jen v tom, že v tomto případě je tým jednočlenný. Při řízení projektu se budeme řídit klasickou základní metodikou vodopádem. Kde je práce rozdělena na přesně definované fáze, které se provádějí v sekvenčním pořadí. Tato metodika je sice již zastaralá a dávno překonaná novějšími postupy. Pro tento projekt je ale dostačující. Její velká nevýhoda při změně požadavků v době vývoje je eliminována přesně daným zadáním práce. Z tohoto důvodu tedy není nutné se zabývat nějakými složitějšími postupy. Vodopád tedy rozděluje práci na přesně dané fáze. Konkrétně jsou těmito fázemi sběr požadavků, návrh, samotná implementaci a nakonec testování. A právě všechny tyto fáze budou dopodrobna probrány v následujících kapitolách. [1][12]

3.1 Sběr požadavků

Prvním krokem, kterým se započne celý projekt, by měl být sběr požadavků. Tuto fázi projektu není vhodné podceňovat a je jí potřeba věnovat dostatek času. Neboť důvod neúspěchu značné části projektů tkví právě ve špatném a zejména neúplném sběru požadavků. Metod získávání požadavků je hned několik. Většinou platí, že bychom se neměly zaměřit jen na jeden způsob sběru požadavků, ale měla by být použita jejich kombinace, kde by se jednotlivé metody měly navzájem doplňovat. Obvykle se jedná o kombinaci konzultace se zákazníkem a analýzy zadání. A právě analýza zadání bude z akademických důvodů použita při tvorbě naší aplikace.

Při sběru požadavků by měla být dodržována základní pravidla. Prvním z nich je rozdělení požadavků na funkční a nefunkční. Funkčním požadavkem můžeme označit to, co by měl systém dělat. Na rozdíl od nefunkčních požadavků, které se spíše zabývají různými omezujícími podmínkami uvalenými na daný systém. Další pravidlo se vztahuje na jejich organizaci. Výsledkem této první etapy by měl být číslovaný seznam, kde související požadavky jsou u sebe, a nepřeskakuje se z jednoho témata na druhé. Samotné názvy požadavků by měly být také nějak strukturovány. Obecně se používá forma <id> <systém> bude <funkce>.

Tato struktura není nějaké dogma, které by mělo být za každou cenu dodržováno. Je to spíše takové doporučení, které zpřehlední výsledný výstup sběru požadavků. [1]

FP1	Systém bude umět vytvořit nabídku.
FP2	Systém bude umět editovat nabídku.
FP3	Systém bude umět smazat nabídku.
FP4	Systém bude umět exportovat nabídku do PDF.
FP5	Systém bude umět vytvořit nabídku pro vybraného zákazníka.
FP6	Systém bude umět přidat poznámku k nabídce.
FP7	Systém bude umět zaevidovat zákazníka.
FP8	Systém bude umět upravit zákazníka.
FP9	Systém bude umět smazat zákazníka.
FP10	Systém bude umět zamezit smazání zákazníka, pro kterého jsou vytvořeny nabídky
FP11	Systém bude umět evidovat položku ceníku.
FP12	Systém bude umět upravit položku ceníku,
FP13	Systém bude umět smazat položku ceníku.
FP14	Systém bude umět přidat nebo odebrat položku ceníku do nabídky.
FP15	Systém bude umět spravovat uživatelské nastavení (adresa, DPH, cena za hodinu).
FP16	Systém bude umět vytvořit pro každého uživatele separátní účet.
FP17	Systém bude umět zamezit přístupu k datům jiných uživatelů.
FP18	Systém bude umět autentizaci a autorizaci.
FP19	Systém bude umět spravovat uživatele.
FP20	Systém bude umět smazat uživatele a všechny jeho data.

Tabulka 1: Seznam funkčních požadavků

NP1	Systém bude provozován jako webová aplikace.
NP2	Systém bude dostupný ze všech operačních systémů.
NP3	Systém bude umožňovat přístup několika uživatelům najednou.
NP4	Front end bude vybudován na frameworku Vaadin.
NP5	Systém bude napojen na databázi typu Mysql.
NP6	Front end bude responzivní.
NP7	Systém bude nasazen na server Tomcat.

Tabulka 2: Seznam nefunkčních požadavků

Ve dvou předchozích tabulkách jsou vidět již kompletní seznamy funkčních a nefunkčních požadavků. Jak je jistě vidět, byla zde snaha o dodržování všech zmíněných pravidel. Požadavky jsou číslovány, obsahují prefix **FP**, a dodržují stanovené schéma názvu požadavku. Posledním atributem bylo slučování souvisejících požadavků do kategorií. A právě tyto kategorie zde budou v následujících kapitolách důkladněji probrány.

3.1.1 Tvorba nabídky

Nejzákladnější aspekt co by asi aplikace pro tvorbu rozpočtů měla umět je určitě samotná tvorba rozpočtu, respektive v našem případě nabídky. Nabídka sama o sobě není nic jiného než jen seznam položek z ceníku spolu se souhrnem ceny za danou nabídku. O ceníku a jeho zde bude samostatná kapitola. Zde budou probrány hlavně požadavky na správu nabídek. Aplikace by tedy měla umět nabídku vytvořit, uložit, editovat již vytvořenou nabídku a případně nabídku smazat. Ze zadání vyplívá, že je více než pravděpodobné, že uživatel nebude vytvářet jen jednu nabídku, ale že jich bude mít na svém účtu více. Proto bylo do požadavků přidána položka, která říká, že k nabídce musí být možno přidat textovou poznámku. Tato poznámka by měla soužit, jak k odlišení podobných nabídek, tak i k přidání doplňujících informací k nabídce. Samotná tvorba nabídky je sice úžasná funkcionalita a sama o sobě by i stačila. Do doby, než si uživatel uvědomí, že potřebuje nabídku nějakým způsobem dostat ze systému a předložit jí svému zákazníkovi. Proto byl do seznamu požadavků přidán další bod, a to možnost nabídku exportovat do PDF souboru. PDF soubor se všemi informacemi o nabídce si už uživatel může stáhnout, vytisknout případně ho poslat zákazníkovi.

3.1.2 Správa zákazníků

Jako další kategorie funkčních požadavků, bude představena skupina požadavků věnující se správě zákazníků. Ze zadání vyplývá, že nabídky mají být personalizované a mají být určeny danému zákazníkovy, který je v systému uložen. Prvním funkčním požadavkem z této kategorie je tedy možnost tvorby a uložení samotného zákazníka. Každá položka v databázi zákazníků by měla odpovídat nějakému skutečnému zákazníkovi z reálného světa. Ti mohou časem měnit svá jména nebo jiné informace. Proto i editace zákazníků v systému musí být uvedena v seznamu požadavků. Nesmí zde chybět ani mazání zákazníku se systému, neboť i reální zákazníci mohou zaniknout nebo přestat využívat služby uživatele aplikace. Do požadavků je ale ze zadání přidaná výjimka, která jasně říká, že nemůže být smazán zákazník, pro kterého jsou vytvořeny nabídky. Pokud by zde tento doplňující požadavek nebyl, tak by se mohlo stát, že v systému budou nabídky, které nemají určeného zákazníka a tím vlastě postrádají smysl.

3.1.3 Ceník a jeho položky

Jak již bylo zmíněno v předchozích kapitolách, nabídka by se měla skládat zejména z položek ceníku. Společně se zadáním projektu byl poskytnut ceník elektrikářských prací, který by měl být součástí projektu, měl by být naimportován do databáze a měl by být dostupný přes aplikaci k tvorbě nabídek. Tím, že jsou poskytnuta data pro import položek ceníku, jsou celkem přesně určeny i jednotlivé atributy položek ceníku. Proto není nutné toto implicitně uvádět do seznamu požadavků. Ze zadání také vyplívá, že každý uživatel by měl mít možnost si svůj ceník upravit dle libosti. Proto do seznamu požadavků přibyly opět položky na vytvoření, úpravu a smazání. Tentokrát se ale jedná o položky ceníku. Z doposud vytvořených požadavků však vyplývá jen to, že by měla být vytvořena jak správa nabídek, tak i správa položek ceníku. Není zde ale nic uvedeno o vztahu mezi těmito dvěma entitami. Aby bylo jasné, že by nabídka měla obsahovat právě seznam položek ceníku, je potřeba toto uvést i v seznamu požadavků. Tuto informaci doplníme o to, že by seznam položek ceníku u nabídky měl být editovatelný a mělo by být možno do něho přidávat či z něho odebírat položky. Samozřejmostí je přidání požadavku na mazání položky ceníku. Musíme ovšem doplnit stejné pravidlo jako u zákazníka a to, že by nemělo být možno smazat položku ceníku, pokud je položka připojena k existující nabídce.

3.1.4 Uživatelský účet a jeho nastavení

Doposud se zde řešily základní požadavky na funkčnost a na to co by výsledný systém měl umět. Je ale potřeba se na dané zadání podívat s trochu větším odstupem a zaměřit se na věci, které zde nejsou implicitně uvedeny, ale pro fungující systém jsou nutností. Jedním z takovýchto téma je řízení uživatelského přístupu. Ze zadání nepřímo vyplívá, že by uživatel aplikace měl mít separátní účet a měl by mít přístup jen ke svým datům. Tento přístup k datům by měl být samozřejmě nějakým způsobem zabezpečený, aby nedošlo ke ztrátě či zneužití dat. Proto by přístup do samotné aplikace neměl být možný, bez předchozí autentizace a autorizace. Aby mohl takovýto systém fungovat v praxi je nutné nějakým způsobem zajistit správu uživatelských účtů. Nějaký nadřazený uživatel by měl mít přístup k vytváření, editaci i mazání běžných uživatelských účtů. Proto kromě požadavků na zabezpečení systému je potřeba do seznamu zahrnout i požadavky na správu uživatelů.

Kromě údajů, kterými by se měl uživatel přihlásit do aplikace, by v systému mělo být možno vyplnit i jiné uživatelské údaje. Jako jsou adresa, cena za hodinu a použité DPH. Právě tyto údaje by se měly využít při tvorbě nabídky, zejména při kalkulaci ceny. Takže i toto uživatelské nastavení bylo přidáno jako jedna z položek do seznamu požadavků.

3.1.5 Nefunkční požadavky

Jak již bylo řečeno, z nefunkčních požadavků nelze poznat, co by měl daný systém obsahovat za funkcionality. Seznam nefunkčních požadavků spíše obsahuje určitá omezení a pravidla, ke kterým by mělo být přihlíženo v rámci vývoje a které mohou nějakým způsobem limitovat nebo naopak zaručovat chování či kvalitativní atributy systému. Ať už jde o použití daných technologií, zaručení výkonu či dostupnosti systému nebo třeba i požadavek na úroveň zabezpečení. [1]

Prvním z nefunkčních požadavků, uvedených v přiložené tabulce, je požadavek na samotný typ aplikace. Ze zadání je jasné že se by se mělo jednat o webovou aplikaci. Je to celkem pochopitelné. V poslední době se vytváří stále více webových aplikací, zatímco vývoj desktopových aplikací je pomalu na ústupu. Toto je způsobeno hlavně tím, že webové aplikace mají velkou výhodu v rámci údržby a distribuce k uživateli. Webová aplikace běží na jednom místě a je dostupná uživatelům přes internet. Zatím co desktopová aplikace musí běžet lokálně na zařízení uživatele, což komplikuje instalaci a zejména distribuci aktualizací. Někdo může

namítat, že u webových aplikací je zde nutnost stálého připojení do sítě, tento problém se ale s rostoucí dostupností internetu stále více eliminuje. [14]

Další nefunkční požadavek navazuje tak trochu na ten první. Aby cílová skupina uživatelů mohla být co největší je potřeba eliminovat omezení, které plynou z uživatelova zařízení. Tím pádem je potřeba zajistit, aby zobrazení aplikace nebylo závislé na operačním systému uživatele. Ono už to tak trochu vyplívá, z požadavku na webovou aplikaci, která tuto podmínku splňuje, tím že běží ve webovém prohlížeči, který by měl být nainstalován na většině používaných OS. Důvodem proč je tato podmínka implicitně zařazena do požadavků, je spíše podpora webových prohlížečů, které jsou dostupné jen pro konkrétní OS. Příkladem může být webový prohlížeč Microsoft Edge nebo Safari.

V zadání je věnovaná část i požadavkům na použité technologie. Důvodů, proč jsou takovéto požadavky uvedeny v zadání, může být několik. Zejména se jedná o nutnost dodržování technologií, z důvodu nasazení na zákazníkovo produkční prostředí, které vývojář není schopen ovlivnit. V této práci jsou to však jen důvody akademické, nicméně do požadavků musí být stejně uvedeny.

3.2 Použité technologie

Po stanovení nastává další důležitý krok při vývoji softwaru. Tímto krokem je samozřejmě výběr použitých technologií. Toto rozhodnutí nelze dělat jen tak od boku a je potřeba se nad ním velmi zamyslet. Výběr špatných technologií by mohl mít za následek zkomplikování vývoje, prodražení projektu nebo i jeho případné zrušení při velkých komplikacích. Nejdříve ze všeho je nutné si ujasnit, jaké typy technologií budou potřeba vybrat. V případě tohoto projektu se jedná zejména o programovací jazyk, případně nějaký framework, technologii pro ukládání dat a v neposlední řadě místo a technologie produkčního prostředí. Důležité je také zjistit jaké budou požadavky na výkon, stabilitu a udržovatelnost aplikace. Tyto hodnoty totiž mají na výběr technologií zásadní vliv.

3.2.1 Java

Jako hlavní programovací jazyk byl zvolen jazyk Java. Je to objektově orientovaný jazyk aktuálně ve vlastnictví firmy Oracle. I přes to je však volně dostupný. Což je možná společně s jeho jednoduchostí důvod, proč je tak oblíbený, a tak často používaný. Tento jazyk totiž běží aktuálně na mnoha typech zařízení, jejichž počet přesáhl hranici 3 bilionů. Hlavní důvody

výběru tohoto jazyk však jsou jiné. Java byla vybrána zejména proto, že jo ve velmi čistý strukturovaný jazyk, který dovoluje znovu použitelnost kódu, což snižuje nákladnost vývoje. Dále je pro tento jazyk dostupné velké množství knihoven, které také ulehčí a zpříjemní práci. [15][16]

3.2.2 Spring boot

Cílem tohoto výběru technologií bylo vybrat takové nástroje, které práci na projektu značně zjednoduší a ušetří práci na vývoji aplikace. Java je sama o sobě už v základu dost jednoduchá, ale dost věcí si člověk musí vytvořit sám a trávit čas na tom, aby znovu vynalezl kolo. Proto byl do výběru technologií zařazen Spring boot. Spring je v základu framework dovolující psát webové aplikace, které budou rychlé bezpečné a budou napsané velice jednoduše. Spring boot je potom už jen před konfigurovaná šablona Springu, která ještě více ulehčuje vývoj. Dalším plusem Springu je jednoduché připojení datových zdrojů jako jsou databáze. Jak již bylo řečeno, součástí springu je také Hibernate, který tomu s využitím ORM také velmi napomáhá. Konfigurace zabezpečení je ve springu také poměrně jednoduché a nabízí mnoho možností, jak přístup do aplikace omezit. [3][4][17][18]

3.2.3 Vaadin

Dnešní moderní trendy při tvorbě webových aplikací se však za posledních pár let změnily. Od původních statických informačních webů psaných jen pomocí HTML s využitím PHP pro funkcionální stránku věci, se přechází spíše na weby fungující jako desktopové aplikace běžících ve webovém prohlížeči, tvořené z valné většiny pomocí JavaScriptu. Hlavní problém, který se JavaScriptové aplikace snaží eliminovat, je nutnost načítat celou stránku při každé změně. Sníží se tak načítací čas, což uživatelé vždy ocení. Přes sít se tedy neposílají celé vykreslené stránky, ale jen zkomprimovaný kód prováděný až v prohlížeči. Program se sice provádí až na straně uživatele a někteří uživatelé můžou namítnout, že to je zbytečné využívání výkonu uživatelského zařízení. V dnešní době, kdy technologie pořad postupují dopředu a osobní počítače se dodávají se stále větším výkonem, to ale nebude zas takový problém.

Aby nově vytvořená aplikace mohla konkurovat i těm nejlepším konkurentům, musí být tedy její frontend vytvořen pomocí JavaScriptu. Tvorba dvou oddělených aplikací je jedno z možných řešení, jak toho docílit. Existuje zde ale i alternativa, která umožní vývoj celé aplikace pomocí Javy, ale frontendovou část následně převede do JavaScriptu. Touto

funkcionalitou oplývá právě framework Vaadin, který bude použit pro tvorbu právě nově vzniklé aplikace. Vaadin tedy nabízí Javě nástroje, kterými lze jednoduše tvořit HTML elementy, zejména pak formulářové prvky, které jsou následně při kompilaci převedeny do JavaScriptu. Toto je možno jen díky technologii GWT, která právě poskytuje potřebné knihovny, kompilátory a vývojové servery, které jsou k tomu zapotřebí. Takovýto převod jednoho programového jazyka do druhého ale není výkonnostně jednoduchá záležitost, a proto by se Vaadin a GWT obecně neměly používat na rozsáhlé projekty. [19][20][21]

3.2.4 Mysql

Po výběru programovacích jazyků a používaných frameworků, je nutno se zaměřit na technologii používanou pro ukládání dat. Samotná aplikace nebude nijak rozsáhlá ani datový model nebude nijak komplikovaný, proto zde není potřeba přemýšlet o nějaké NoSQL databázi. Bylo by zde spíše vhodné poohlédnout se po nějaké relační databázi.

Z existujících relačních databází byla za vítěze pro novou aplikaci, zvolena databáze Mysql. Tato databáze je využívána nejčastěji pro webové aplikace. Je tedy značně rozšířená a najít webový hosting s Mysql databází, kde by výsledná aplikace běžela, by neměl být problém. Důvodem, proč je tak rozšířená, by mohla být jednoduchost, dostupnost a také její zaměření na rychlost vyhledávání. Což je právě výhodné i pro tvořenou aplikaci. [2][23]

3.3 Vývojové nástroje

Používané technologie jsou jedna věc, druhou věcí jsou však vývojové nástroje. Jejich výběr je leckdy důležitější než výběr samotných technologií. Pokud vývojář nemá dostačující nebo dokonce žádné vývojové nástroje, vývoj se může zpomalit nebo dokonce zastavit, a to má samozřejmě vliv na projektové náklady a s tím spojené věci. Pokud tomu je naopak a vývojář má kvalitní softwarovou výbavu, která mu práci usnadňuje, vývoj se zrychlí a práce odsýpá tak jak by měla.

3.3.1 IntelliJ IDEA

Nejdůležitějším vývojovým nástrojem je pro vývojáře asi jeho IDE pro psaní kódu. IntelliJ IDEA je vývojové prostředí od společnosti JetBrains. Toto prostředí je v základní verzi zdarma a poskytuje většinu jeho výhod při psaní Java aplikace. Mezi výhody patří chytré napovídání kódu, formátování, detekce duplicit, detekce chyb, nabídky řešení chyb, podpora

neskutečného množství klávesových zkratek a mnoho dalšího. Pro vývoj springové aplikace je ale vhodnější verze Ultimate. Ta nabízí veškerou podporu při psaní nejen ve Springu, ale i v jiných frameworcích. Tato rozšířená verze ale už není zadarmo a je potřeba si za ni v měsíčních poplatcích připlatit. JetBrains však tuto verzi nabízí studentům a akademickým pracovníkům zdarma. [22]

3.3.2 DataGrip

IntelliJ IDEA obsahuje i nějaké nástroje pro správu databází, není to ale nic převratného a vývojář by ocenil nějaký plnohodnotný nástroj, který by mu opět ulehčil práci. Proto byl zvolen program DataGrip opět od společnosti JetBrains. Podobně jako IntelliJ IDEA nabízí plnou řadu funkcí, které si jako vývojář zamilujete. Jako jeden z mála nástrojů není věrný jen jednomu typu databáze, kterých je zde celá řada a které lze přes tento nástroj spravovat. Jednou z nich je právě databáze vybraná pro tento projekt. Tím pádem nebylo při výběru databázového správce co řešit. DataGrip je ale bohužel tak jako IntelliJ IDEA Ultimate zpoplatněn. Nicméně je zde opět přístup k bezplatné verzi pro studenty. [24]

3.3.3 Mysql Workbench

Společnost JetBrains nabízí skvělé IDE a kromě těch dvou zmiňovaných v této práci nabízí ještě nespočet dalších. Pro návrh struktury databáze bude ale použit nástroj od jiné společnosti. Konktrétně to bude Mysql Workbench od společnosti Oracle. Tento nástroj nabízí celou řadu funkcí od modelování datového modelu, přes konstrukci SQL dotazů, až po správu běžící databáze a řízení uživatelského přístupu.

Pro Tuto práci je, ale dostačující možnost tvorby datového modelu. Postup práce při jeho tvorbě zde spočívá ve vytváření entit. Těm lze přidávat atributy s případným nastavením datového typu, omezením rozsahu či specifikováním povinnosti. Nechybí zde samozřejmě ani nástroj pro tvorbu vazeb mezi entitami. Ruční tvorba modelu však není jediným možným řešením, jak model vytvořit. Mysql Workbench totiž umožňuje model vytvořit pomocí SQL skriptu. Stačí tedy vyexportovat strukturu již nějaké existující databáze a přes skript je možno ji naimportovat do vývojového prostředí naimportovat. Program také dovede tuto funkcionalitu provádět naopak a z již hotového modelu vytvořit SQL skript, pomocí kterého lze vytvořit funkční databáze. [11]

3.4 Datový model

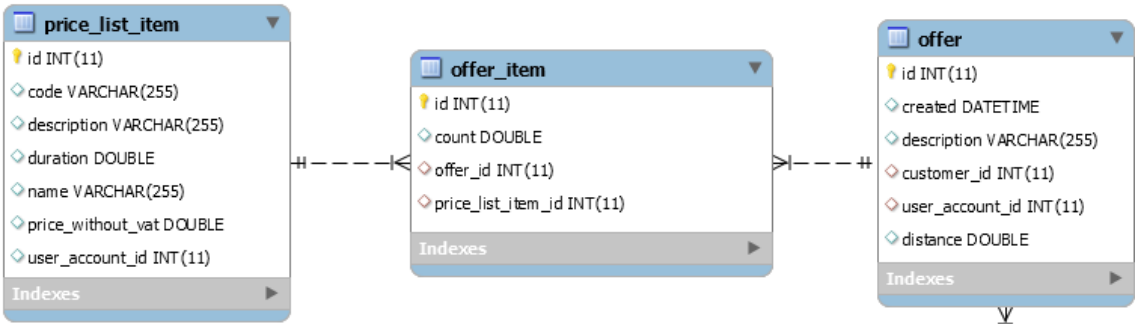
Samotná fáze návrhu systému je velice důležitá část a stejně jako sběr požadavků by se neměla podceňovat. Jako u výběru používaných technologií je i u datového modelu potřeba udělat správná rozhodnutí a navrhnout ho co nejlépe. Mohlo by totiž se v pozdějších fázích vývoje narazit na problémy, které by se buď musely řešit složitou softwarovou cestou, nebo by se nedaly řešit vůbec a projekt by se musel aspoň z části přepracovat. Použitý Framework toto riziko velmi eliminuje. Poskytuje totiž technologii zvanou Hibernate, která ovládá objektově relační mapování, což nám ulehčí spoustu práce při vytváření datového modelu a umožní nám jeho tvorbu až v rámci vývoje. Nicméně datový model je dobré navrhnout ještě před vývojem, z důvodu zamezení výskytu nepředvídatelných chyb a problémů.

Datový model tedy zobrazuje vlastní datovou strukturu aplikace. Jsou zde podrobně zachyceny všechny datové entity, jejich atributy a samozřejmě vztahy mezi entitami. V podstatě nám ukazuje, s jakými daty aplikace pracuje. Po zachycení těchto znalostí do diagramu, lze získat takzvaný ER diagram. Ten ale není jediný diagram, kterým můžeme datový model zachytit. Stejně tak dobře funguje i model popisující databázovou strukturu. ER diagram jde totiž jednoduše převést do databázového modelu. Stačí vyměnit entity za tabulky, atributy za sloupce tabulek a vztahy mezi entitami za relace. Tento přístup však dále v této práci použit nebude a budou zde používány termíny z ER diagramu.

Při konstrukci datového modelu je potřeba provést analýzu zadání a zjistit jaké budou v modelu entity, jaké budou entity mít atributy a jaké vztahy mezi entitami vzniknou. Jak je vidět na již vytvořeném modelu je zde hned několik entit s atributy, které jsou navzájem propojeny pomocí vztahů. A právě tyto entity zde budou následně představeny v dalších kapitolách.

3.4.1 Nabídka a její položky

Nejpodstatnější část celé aplikace je asi tvorba samotné nabídky. Z analýzy zadání bylo zjištěno, že by proces pro tvorbu nabídky měl využívat nejméně dvě hlavní datové entity, které jsou vidět na obrázku 4.



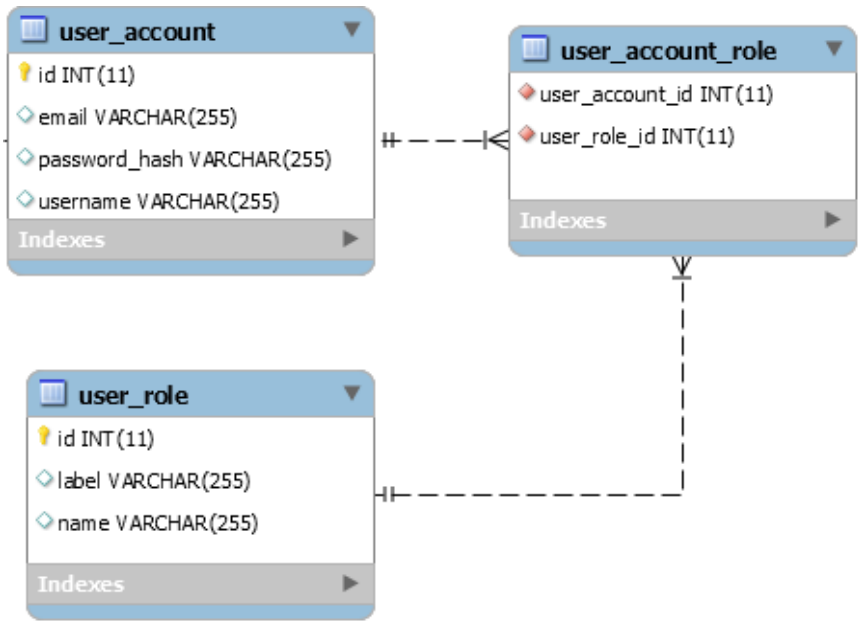
Obrázek 4: Entity pro nabídku a její položky

První z nich je entita pro samotnou nabídku. Mezi jejími atributy jsou položky, které by měla obsahovat i nabídka z reálného světa. Musí zde být atribut odpovídající zákazníkovi, pro kterého je nabídka realizována. Dále je zde uvedena položka `USER_ACOUNT` neboli uživatelský účet autora nabídky. Z požadavků jasně vyplívá, že nabídka musí obsahovat popis, proto je i pro něj zde vytvořen atribut. Jelikož cena nabídky se neskládá jen z položek nabídky, ale je zde zahrnuta cena za dopravu, musí být u nabídky uvedena i vzdálenost k zákazníkovi, ze které se cena za dopravu již dá spočítat. Posledními dvěma tributy, které nejsou implicitně uvedeny v zadání, jsou `CREATED` a `ID`. Každá entita musí mýt svůj unikátní identifikátor proto je zde atribut `ID`. Atribut `CREATED` je zde zase proto, aby se vždy vědělo, kdy byla daná entita vytvořena. Tento atribut může být nápomocný třeba při ladění systému nebo při hledání chyb.

Jak je napsáno v zadání a v již dokončené analýze, nabídka se skládá zejména z položek ceníku. Položka ceníku je tedy druhá datová entita uvedena v ER diagramu. Při sběru požadavků bylo řečeno, že atributy položky zde mají více méně pevně danou strukturu danou importem dat. Každá položka v ceníku tedy obsahuje název, popis, cenu za materiál a předpokládanou dobu práce. Jelikož jsou ceníky personalizované a každý uživatel by měl mít ceník vlastní, je u položky vždy uveden uživatelský učet, ke kterému je ceník navázán.

Tím, jak jsou položky ceníku vkládány do nabídky, vzniká zde mezi nimi vazba. Tato spojení však není tvořeno klasickou vazbou 1:N, ale je zde zapotřebí použít vazbu N:M. Jedna nabídka totiž obsahuje N položek ceníku a naopak položka ceníku může být použita u více nabídek. Proto zde musí být vytvořena další vazebná entita položka nabídky, která rozdělí vazbu N:M na dvě vazby 1:N. Položka nabídky také slouží k uchování počtu kusů, která by jinak muselo být nahrazeno duplikováním entity položky nabídky.

3.4.2 Uživatelské účty



Obrázek 5: Entity uživatelského účtu

Při budování datového modelu se postupuje podobně jako při sběru požadavků. Nejdříve jsou modelovány základní funkcionality a postupně se přistupuje i k těm méně důležitým. Proto po návrhu struktury nabídek a ceníků přišla na řadu část zabývající se uživatelskými účty.

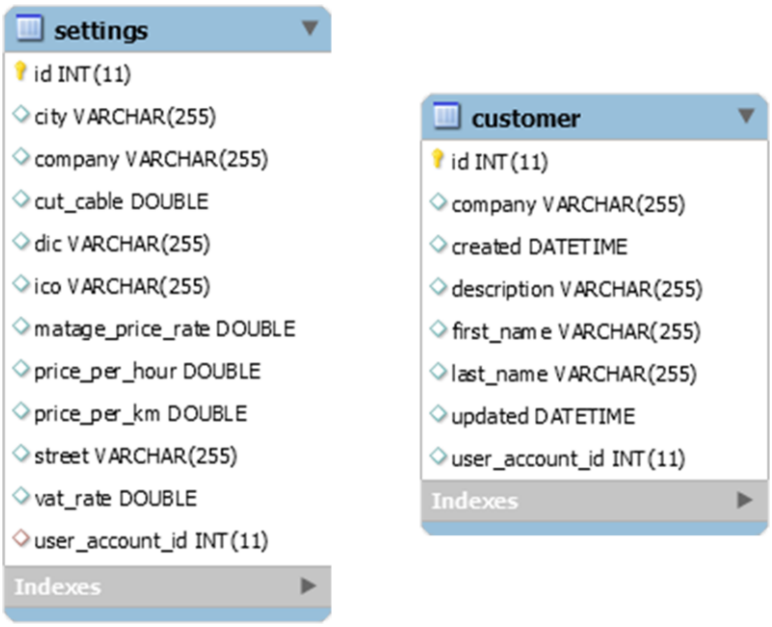
Obrázek 5, ukazuje že, pro uživatelské účty byly navrženy také nejdříve dvě entity. Konkrétně to jsou entity pro uživatelský účet a pro roli, která může být uživateli přiřazena. Atributů, které entita uživatelského účtu obsahuje, není hodně. V podstatě jde jen o přihlašovací údaje do aplikace. Je zde přihlašovací jméno, hash vytvořený z hesla a emailová adresa přes kterou je možno uživatele v případě nutnosti kontaktovat. Entita pro uživatelské role obsahuje jen dva atributy. Jedním z nich je čitelný název zobrazovaný v aplikaci. Druhým je pak strojový název používaný v programu aplikace. Mezi těmito dvěma entitami je, tak jako v předchozím případě, vztah N:M, který je opět nutno vyřešit pomocí vazební entity. V tomto případě však do vazební entity nebude přidávaný žádný atribut.

3.4.3 Ostatní datové entity

Poslední dvě entity, které doposud nebyly představeny, jsou zobrazeny na obrázku 6. Jak již bylo zmíněno při tvorbě entit nabídky, existuje zde entita obsahující informace o zákazníkovi. Jelikož se nabídka tvoří vždy pro nějakého zákazníka. Je také zde vazba 1:N, která tento vztah

popisuje. Samotná entita zákazníka obsahuje opět několik atributů. Mezi nimi lze najít, název společnosti, popis, jméno a příjmení, a nakonec i uživatelský účet, ke kterému je zákazník připojen.

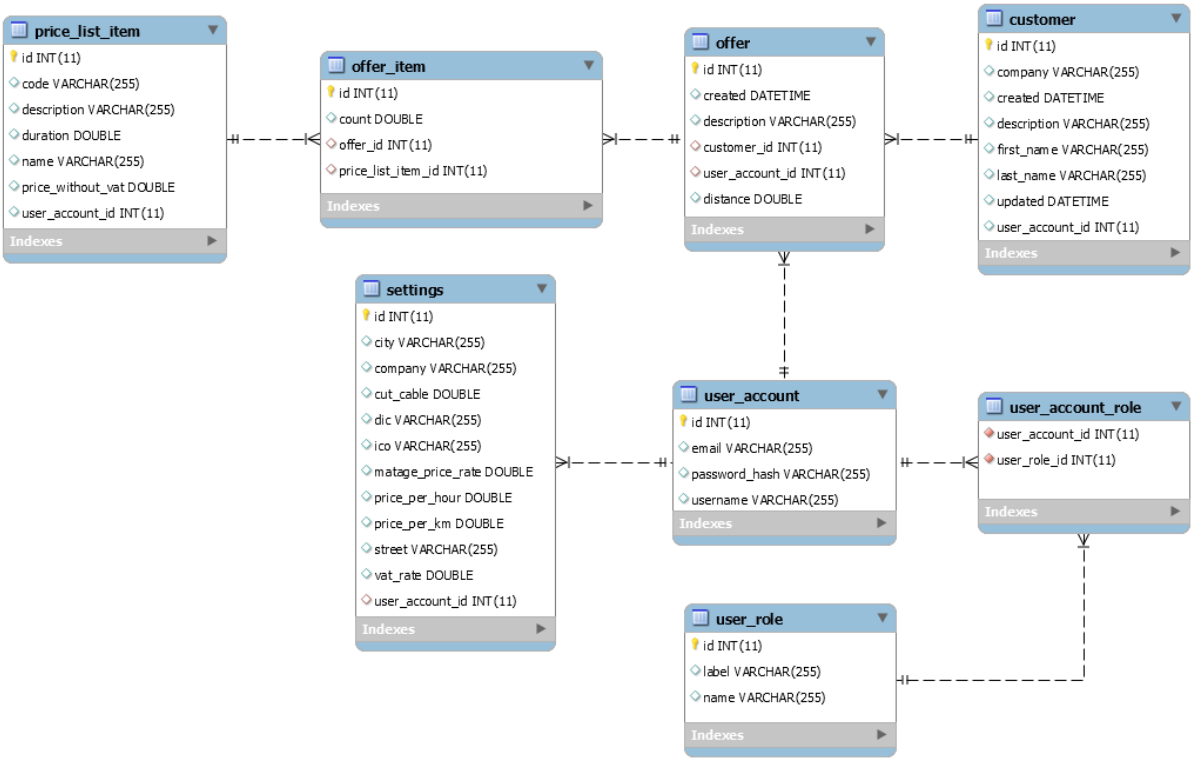
Poslední entitou, která je ER diagramu a ještě nebyla představena, je entita `SETTINGS`. Tato entita má za úkol uchovávat všechny uživatelovi informace. Příkladem těchto informací jsou pole, které dohromady tvoří kompletní adresu. Dále entita obsahuje pole pro výpočet ceny objednávky, mezi kterými nechybí cena za hodinu práce a hodnota DPH. Opět je tato entita napojena vždy na jeden uživatelský učet, proto i zde je pole pro `ID` daného uživatelského účtu.



Obrázek 6: Entity pro uživatelské nastavení a zákazníka

3.4.4 Celkový model

Výsledný model se tedy skládá ze všech zmiňovaných částí. Mezi nimi jsou však ještě některé vztahy, které zde nebyly implicitně uvedeny. V centru diagramu by nejspíše mohl být blok uživatelských účtu, na které jsou napojeny vlastně všechny ostatní části. Vznikne zde tedy vazba mezi uživatelským účtem a nabídkou. Dále je zde vazba mezi uživatelem a jeho nastavením. Entita zákazník je však napojena vazbou na nabídku, aby bylo jasné, pro jakého zákazníka je nabídka tvořena. Celkový ceník se všemi vazbami je pak vidět na následujícím obrázku 7.



Obrázek 7: Celkový ER diagram

3.5 Návrh UI

Návrh uživatelského rozhraní je v podstatě poslední krok před samotnou implementací aplikace. Obvykle se do jeho návrhu nevkládá tolik úsilí oproti třeba zmiňované tvorbě datového modelu. Jak ale říká jedna poučka, vzhled prodává. Proto ani návrh uživatelského rozhraní není radno zanedbat. Zvláště pokud jde o aplikaci, která chce ve všech ohledech překonat konkurenci. Vyvíjená aplikace také neobsahuje ohromné množství funkcionalit, které by mohly grafický návrh takříkajíc posunout do pozadí. Další z věcí, která přikládá váhu grafickému návrhu, je použitý framework. Ten, jak již bylo řečeno při výběru technologií, buduje frontendovou část aplikace pomocí JavaScriptu, což dává návrháři uživatelského rozhraní veliké možnosti a dovoluje mu to využívat moderní webové komponenty.

Pokud jde rozsahem o malou aplikaci a vývojářský tým už je v tomto oboru zkušený. Může se návrh uživatelského rozhraní přeskočit a tvořit ho až při samotné implementaci. Nicméně tento postup v této práci použit nebyl a uživatelské rozhraní bylo navrženo již před implementací. Návrh tohoto rozhraní se obvykle skládá z řady wireframů, které ukazují, jak by mohla výsledná aplikace vypadat. Více méně jde o strukturální návrh aplikace. Je zde ukázáno, rozvrhnutí jednotlivých funkčních bloků, formulářových prvků, grafických elementů a mnoho dalšího.

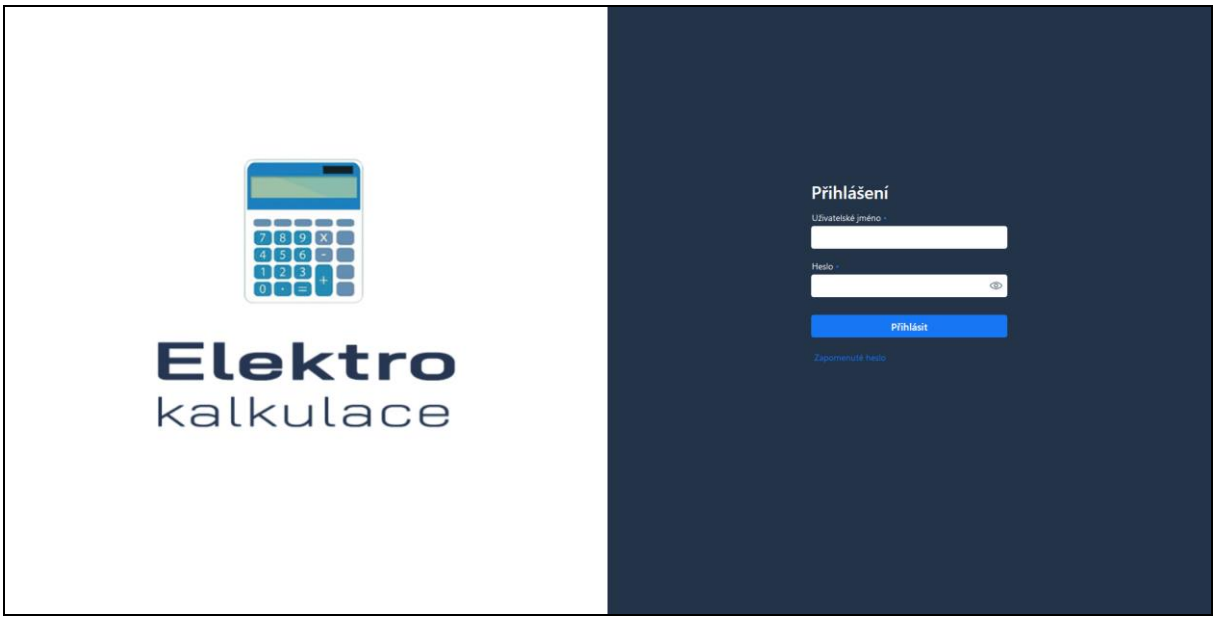
Takto vypadající wireframy, jsou velmi cenné při návrhu a nakonec i při implementaci aplikace. V době psaní této práce však již byla aplikace vytvořena a jsou k dispozici ukázky ze samotné aplikace, které jsou rozhodně více vypovídající než pouhé wireframy. Proto zde budou návrhy uživatelského rozhraní a jednotlivé funkcionality představovány právě na nich.

3.5.1 Přihlašovací obrazovka

Přihlašovací obrazovka by měla být první z věcí, co uživatel uvidí, když poprvé navštíví tuto novou aplikaci. Je to vlastně taková vstupní brána do světa tvorby nabídek. Toto přirovnání bylo použito schválně, protože by tato obrazovka měla opravdu sloužit jako brána, která by měla propustit jen přihlášené uživatele a před těmi ostatními by se měla zavřít.

Je zde přihlašovací formulář, který slouží pro zadání uživatelského jména a hesla. Po zadání těchto údajů může uživatel stisknout tlačítko přihasit, čímž spustí proces ověření uživatelovi identity. Při správné kombinaci přihlašovacích údajů je tedy uživatel vpuštěn do aplikace, kde má přístup jen k povoleným funkcím, které jsou určeny hodnotami jeho uživatelského oprávnění. Probíhá zde tedy proces takzvaná autentizace a autorizace. Pokud uživatel tímto ověřovacím procesem neprojde, ukáže se mu na obrazovce informace o zadání špatných přihlašovacích údajů. Tato informace je vyobrazena v chybovém banneru, aby ji uživatel nepřehlédl.

Jakožto úvodní obrazovka musí být tako stránka vyvedena na velmi vysoké grafické úrovni, aby rovnou svým vzhledem uživatele neodradila od dalšího používání. Úvodní stránka je zde rozdělena na dvě části. Jelikož je jediným funkčním obsahem přihlašovací formulář, bylo do levé části obrazovky umístěno velké logo aplikace. V pravé části obrazovky je už samotný přihlašovací formulář. Ten je na rozdíl od celé aplikace vyveden v tmavém tématu. Důvodem je tvorba kontrastu mezi levou a pravou částí aplikace, který zamezí pocitu prázdnosti celé stránky. Pro představu je návrh úvodní strany zobrazen na následujícím obrázku 8.



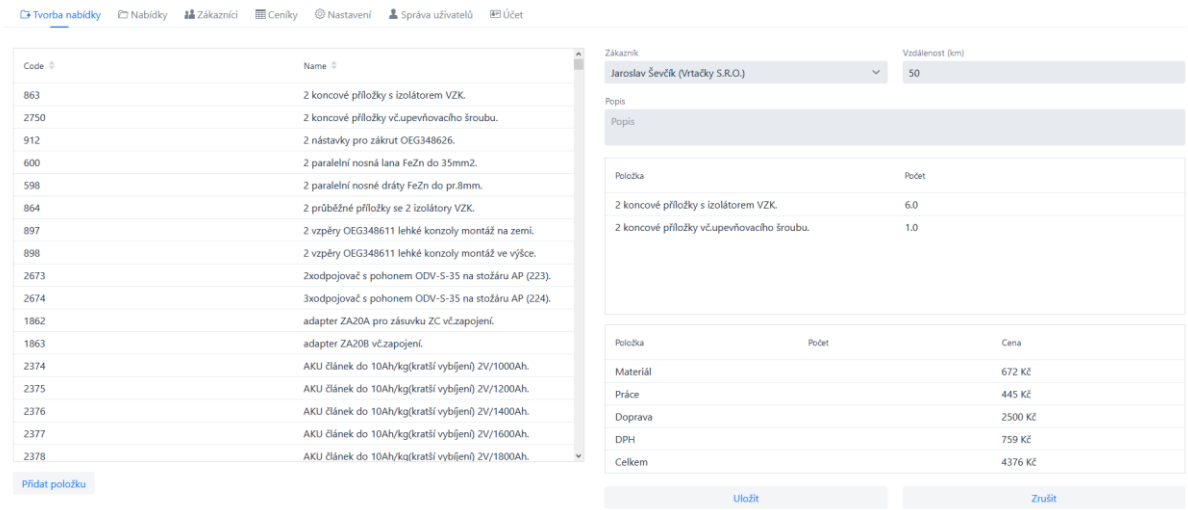
Obrázek 8: Přihlašovací obrazovka

3.5.2 Tvorba nabídky

Po úspěšném zadání svých uživatelských údajů a následném přihlášení se uživatel dostane do samotné aplikace. Ta je, jak již bylo řečeno, vyobrazena pomocí světlého tématu. Na rozdíl od úvodní stránky, kde je přihlašovací formulář vytvořen v tématu tmavém. Jak je vidět na obrázku 9 k navigaci po aplikaci slouží horní menu s názvy stránek a odpovídajícími ikonkami. Není to ale klasické menu, které je většinou u klasických statických webů, kde po změně stránky následuje znovunačtení aplikace. Zde menu funguje na principu přepínání záložek. Ty jsou sice načtené všechny, ale zobrazovaná je pouze jediná, odpovídající výběru v menu. Takovéto prodlevy nebyly odstraněny jen z menu, ale i ze všech míst v aplikaci. Ta je díky tomu více plynulá, načítací čas byl zredukován na minimum a byl tak zlepšen uživatelský prožitek z používání aplikace.

První záložkou, která je uživateli zobrazena, je záložka pro tvorbu a editaci nabídky. Jako většina záložek v aplikaci je rozdělena na dvě hlavní části. Jak již bylo popsáno dříve, samotná nabídka je tvořena položkami z uživatelova ceníku. Proto levou část obrazovky, zabírá obsáhlý seznam položek uživatelova ceníku, ze kterých si může uživatel vybrat následně je přidat do tvořené nabídky. Aby uživatel lépe rozeznal jednotlivé položky ceníku a lépe se mu v něm orientovalo, seznam obsahuje kromě názvu položky také její kód. Samotné přidání položky do nabídky jde udělat dvěma způsoby. Prvním z nich je označení položky a následné kliknutí na tlačítko „Přidat položku“ pod seznamem. Druhým a asi jednodušším způsobem je dvojklik na

položku v ceníku. Po přidání položky do nabídky se aktualizuje druhá strana obrazovky, na které se zobrazuje právě rozpracovaná nabídka. Seznam přidaných položek nabídky se zobrazuje uprostřed pravého panelu. Kromě informační funkce tohoto výpisu je zde i možnost změny počtu kusů vybrané položky. Aby tvorba nabídky mohla být úplná, jsou zde i další pole pro doplnění ostatních informací o nabídce. Prvním z nich je pole pro zákazníka, které je vytvořeno na způsob výběru ze seznamu, ve kterém se dá i vyhledávat. Aby mohla být spočtena hodnota ceny za dopravu, je nutné zde mít i pole pro zadávání vzdálenosti k zákazníkovi. Posledním polem, které bylo zmíněno už při sběru požadavků, je pole pro popis nabídky. Jedná se konkrétně o komponent `textarea` do které lze vyplnit více než 200 znaků. Aby tvorba nabídky byla kompletní, je zde i poslední blok na stránce. Tento blok je sice jen informační, nicméně se jedná zřejmě o nejdůležitější blok na stránce. Obsahuje totiž informace o ceně nabídky. Cena je zde nejdříve rozepsána do všech jejich položek. Ty jsou následně sečteny a celková cena je uvedena na konci seznamu. Posledním krokem při tvorbě či změně nabídky je její uložení. To lze provést pomocí tlačítka „Uložit“ v pravém dolním rohu. Pokud není nabídka uložena před ukončením aplikace, jsou všechna její data nenávratně ztracena.

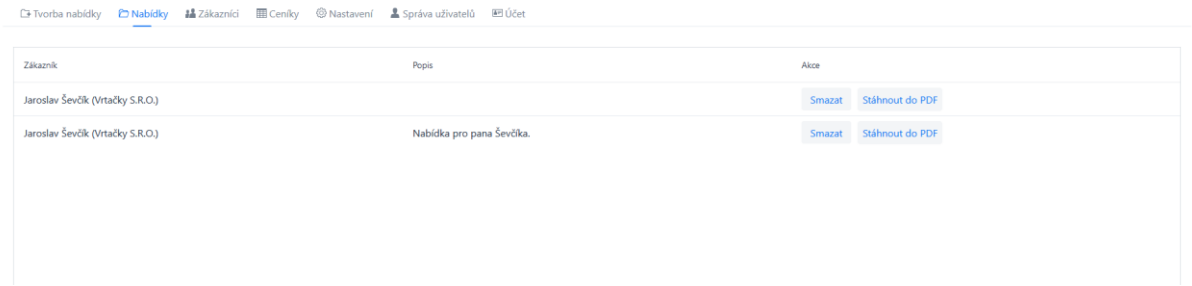


Obrázek 9: Tvorba nabídky

3.5.3 Správa nabídek

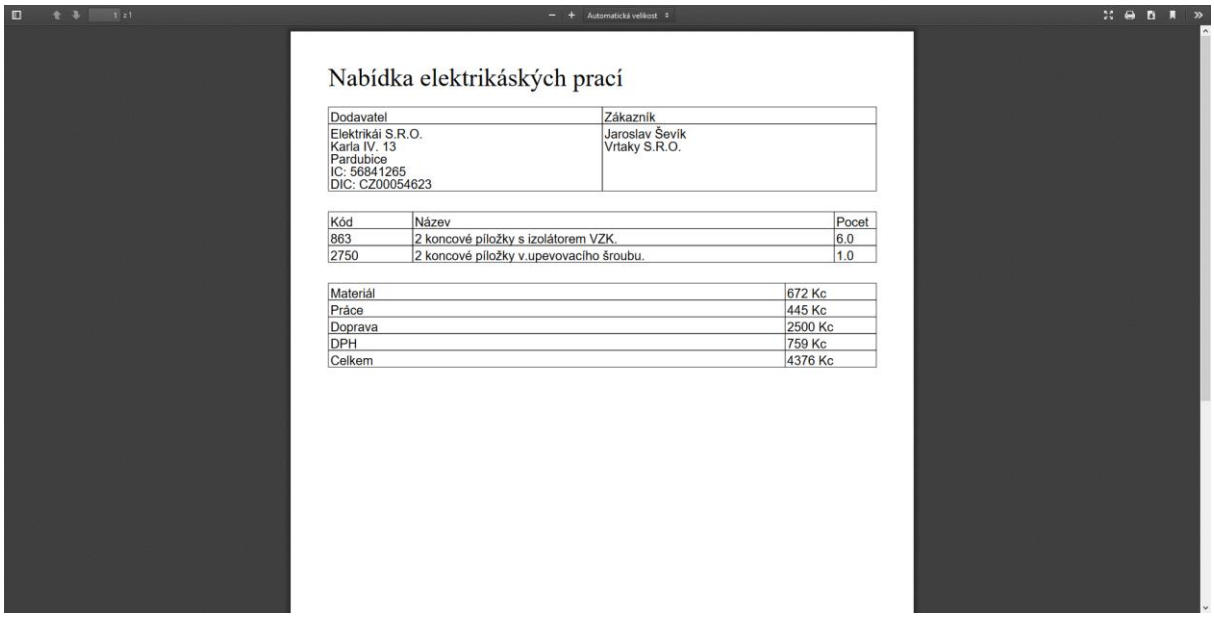
Po vytvoření a uložení nabídky je potřeba uživateli nabídnout, alespoň základní možnost již vytvořené nabídky spravovat. A právě k tomu účelu je v aplikaci druhá záložka nazvaná jednoduše „Nabídky“. Tato záložka je oproti záložce s tvorbou nabídky velmi jednoduchá.

Obsahuje totiž jen seznam vytvořených nabídek. Tyto nabídky jsou však vyfiltrovány a celkový seznam je omezen pouze na nabídky vytvořené přihlášeným uživatelem. Což byla jedna ze základních podmínek. Že uživatel uvidí jen svoje nabídky a naopak, že nabídky jím vytvořené neuvidí nikdo jiný.



Obrázek 10: Seznam nabídek

Tato obrazovka nabízí ale více funkcionalit než jen obyčejný seznam nabídek. Umožňuje například po kliknutí na položku seznamu, vybranou nabídku poslat zpět na první záložku a nabídku tak poskytnout k editaci. Položky seznamu dále, kromě zákazníka a popisu nabídky, obsahují také dvě tlačítka. Tím prvním je tlačítko „Smazat“. To nabízí možnost nabídku po potvrzení ve vyskakovacím okně smazat. Druhým z nich je tlačítko s názvem „Stáhnout do PDF“. Po stisknutí tohoto tlačítka se v novém panelu otevře PDF dokument s kompletními informacemi o vybrané nabídce. Tento dokument je možno stáhnout nebo vytisknout a poskytnout tak nabídku zákazníkovi. Jeho výslednou podobu ukazuje obrázek 11.



Nabídka elektrikáských prací

Dodavatel	Zákazník
Elektrikái S.R.O. Karla IV. 13 Pardubice IC: 56841265 DIC: CZ00054623	Jaroslav Ševík Vrtaky S.R.O.

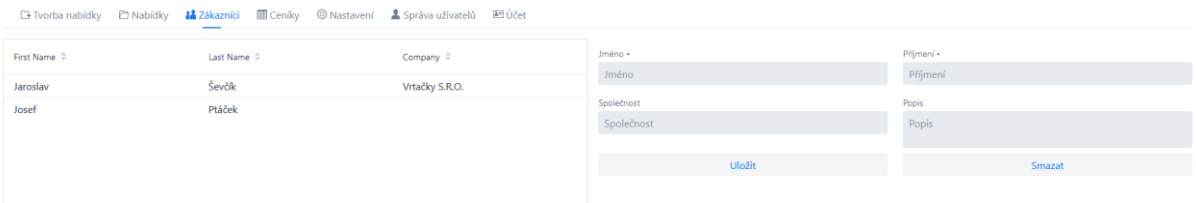
Kód	Název	Pocet
863	2 koncové příložky s izolátorem VZK.	6.0
2750	2 koncové příložky v.upevovacího šroubu.	1.0

Materiál	672 Kc
Práce	445 Kc
Doprava	2500 Kc
DPH	759 Kc
Celkem	4376 Kc

Obrázek 11: PDF export nabídky

3.5.4 Správa zákazníků

Jak bylo naznačeno při představování záložky pro tvorbu nabídek, aplikace obsahuje uložené entity zákazníků, ze kterých si uživatel vybírá právě při vytváření nabídky. Tyto entity zákazníků je tedy potřeba někde vytvářet, editovat a kompletně spravovat. A k tomu byla vytvořena právě záložka s názvem „Zákazníci“. Ta je podobně jako záložka „Tvorba nabídky“ horizontálně rozdělena na dvě části. Levý blok obsahuje seznam uložených zákazníků. Stejně jako seznam nabídek jsou položky omezeny jen na zákazníky přihlášeného uživatele. Položky seznamu obsahují, kromě popisu, všechny atributy, které zákazník má. Tyto atributy jsou pak v seznamu zobrazeny jako jednotlivé sloupce, podle kterých lze dokonce položky řadit, což naznačují i šipky u názvů sloupců, které jsou vidět na následujícím obrázku.



Obrázek 12: Správa zákazníků

Po výběru zákazníka ze seznamu se promítnou hodnoty jeho atributů do formulářových polí v pravé straně obrazovky. Tyto pole slouží právě k úpravě již vytvořených zákazníků, ale to jen v případě, že je zákazník opravdu vybrán. Pokud je tomu naopak a pole jsou prázdná, je možno je využít k vytvoření zákazníka nového. Pod formulářem je jako obvykle tlačítko pro uložení dané entity. Je zde ale navíc tlačítko „Smazat“, které by mělo vybraného zákazníka odstranit. Jak ukazuje obrázek 13, je zde ale možnost, že toto tlačítko nebude povoleno a smazání zákazníka nebude umožněno. Tento stav nastane v případě, že je pro zákazníka v systému vytvořena alespoň jedna nabídka. Uživatel je na tento stav předem upozorněn pomocí chybové hlášky umístěné pod formulářem.

Obrázek 13: Chybová hláška při mazání zákazníka

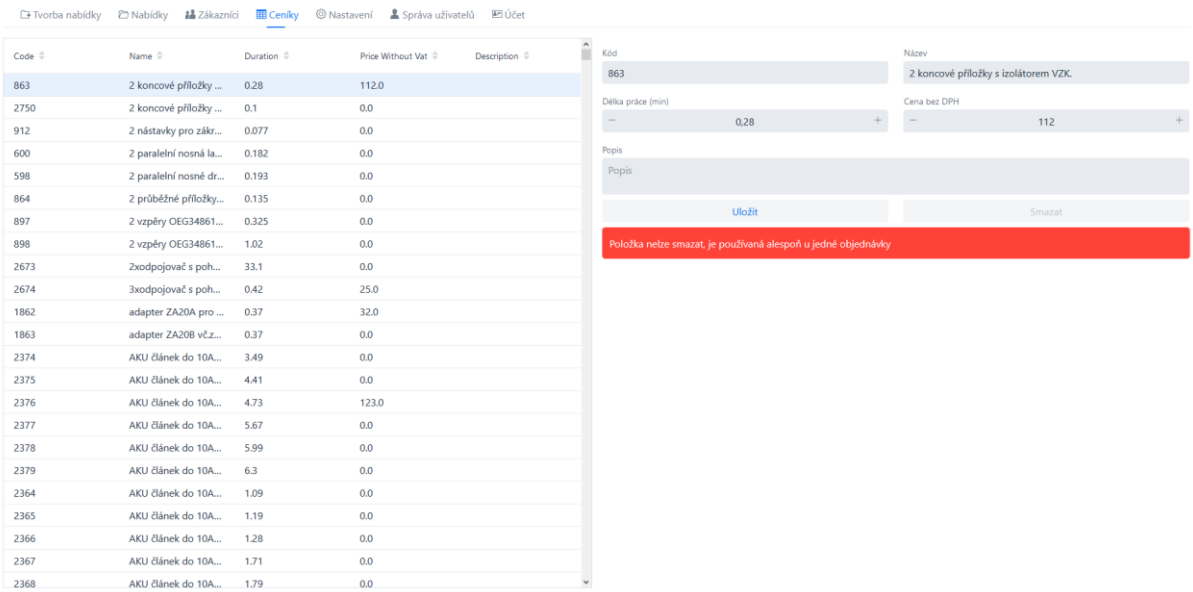
3.5.5 Správa ceníku

V první záložce aplikace je seznam položek ceníku, ze kterého je možné vybírat jednotlivé položky a vytvořit z nich tak nabídku. Jak již bylo zmíněno při sběru požadavků, seznam těchto položek ceníku byl do databáze aplikace naimportován při vývoji. Nicméně bylo zde taky řečeno, že ceník je kompletně upravitelný a každý uživatel si ho může nakonfigurovat podle sebe. Tím pádem má každý uživatel jedinečné položky ceníku, které mezi sebou uživatelé nesdílí. Tím pádem je potřeba mít v aplikaci další záložku, která by poskytovala uživateli možnost ceník spravovat. Proto byla vytvořena karta nesoucí jednoduchý název „Ceník“.

Tato v pořadí čtvrtá záložka, zobrazena i na obrázku 14, se strukturou velice podobá předchozí záložce. Také je horizontálně rozdělena na dvě části. Přičemž levá část obsahuje seznam uložených entit, zatím co v pravé části je umístěn formulář pro tvorbu či editaci vybrané entity. Opět je zde i kontrola závislosti nabídky na vybrané položce, při pokusu položku smazat. Jediný

rozdíl mezi touto a předchozí záložkou je v tom, že zde se jedná o entitu položky ceníku, kdežto v předchozím případě to byla entita zákazníka.

Pokud si ceník prohlíží právě vytvořený uživatel, při prvním pohledu zjistí, že jeho ceník není prázdný. Při vytvoření nového účtu se totiž rozkopíruje výchozí ceník a přiřadí se novému uživateli. Ten tak má svůj vlastní ceník s položkami, které může libovolně upravovat, aniž by měnil leník někoho jiného. Skladbu výchozího ceníku má na starost takzvaný výchozí uživatel. Ten bude ale představen až dále v této práci.



Obrázek 14: Správa ceníku

3.5.6 Uživatelské nastavení

Poslední entita, která zásadně zasahuje do tvorby nabídky je uživatelské nastavení aktuálně přihlášeného uživatele. Tímto nastavením lze u nabídky výrazně ovlivnit, jak budou vypadat její jednotlivé cenové položky, což má v důsledku vliv i na cenu výslednou. Tyto cenové výpočty budou sice představovány až v následující kapitole, nicméně jednotlivé položky nastavení, které tento proces ovlivňují, zde představeny být můžou. Jedná se tedy o pole pro nastavení ceny za hodinu, ceny za kilometr, výše DPH a nakonec procenta slevy.

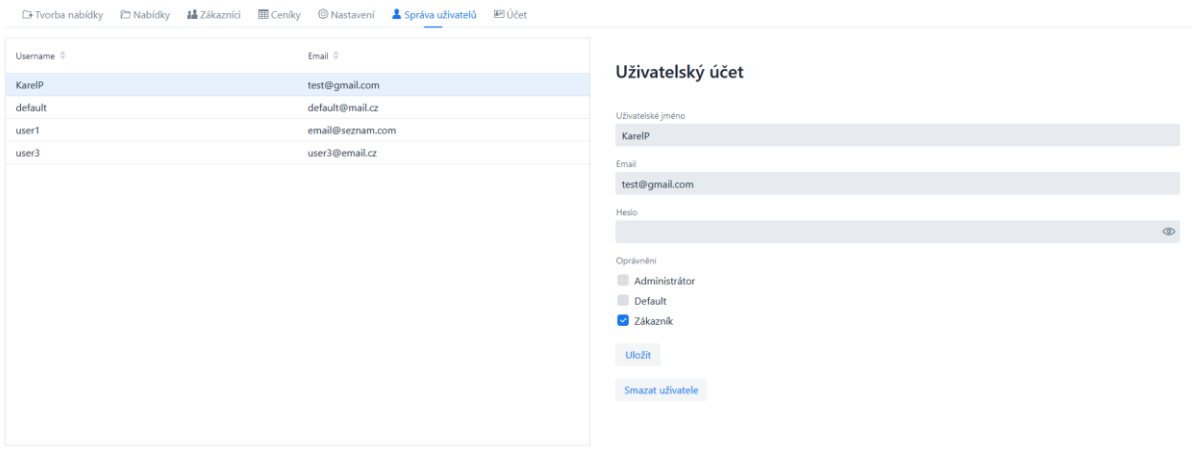
Kromě své cenové politiky zde uživatel vyplňuje i další informace o sobě. Konkrétně jde o název zastupující společnosti, její adresu a nakonec i její IČ a DIČ. Je důležité, aby uživatel tyto informace vyplnil, propisují se totiž do celkového souhrnu nabídky, který je exportován do PDF, což je vlastně takový výstupní kanál aplikace.

Obrázek 15: Uživatelské nastavení

3.5.7 Správa uživatelů

Předchozí záložky se všechny více méně věnovali tvorbě nabídky nebo s ní souvisejících entit. Tato záložka je ale odlišná a má do ní přístup jen uživatel s oprávněním „Administrátor“. A právě role a celkově uživatelské účty jsou to, co se na této záložce spravuje. Obrazovka, jejíž ukázka je na obrázku 16, je jako obvykle rozdělena na dvě části. Levá část obsahuje seznam uživatelů, kteří mají vytvořený uživatelský účet. Je zde vidět jejich uživatelské jméno a email pro případný kontakt. To, že zde má uživatel vytvořený učet, ještě nezaručuje, že se do aplikace přihlásí. Tento přístup dostane pouze v případě, že má u svého účtu nastavenou roli „Zákazník“. A právě nastavení rolí společně s editací uživatele je to, co je umístěno v pravé části obrazovky. Kromě role zákazníka zde již byla zmiňovaná role administrátora, která umožňuje uživateli přístup právě do této sekce se správou uživatelů. Je zde ale ještě jedna speciální role a tou je role „Default“. Defaultní neboli výchozí uživatel, je uživatel, který spravuje výchozí ceník. A právě tento ceník je rozkopírován a přiřazen nově vytvořenému uživateli. Proto je tento uživatel spravující výchozí ceník velice důležitý a v systému by měl být vždy právě jednou.

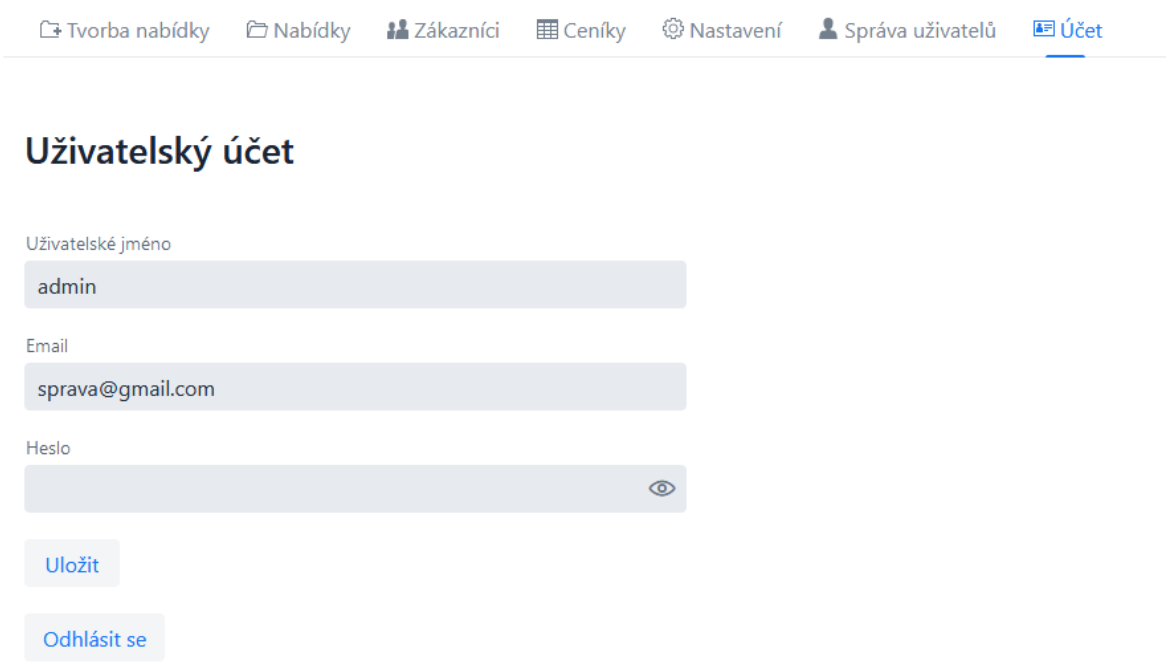
Kromě editace již existujících uživatelů, slouží tento formulář také k přidávání nových uživatelských přístupů do systému. Tak jako u formulářů na předchozích záložkách zde stačí uložit aktuálně upravovaného uživatele. Tím se formulář vyprázdní a je možné ho použít právě k registraci uživatele nového. Tento postup je zatím jediným způsobem, jak přidávat uživatele do systému. Nicméně do budoucna je plánu rozšířit systém o možnost registrace samotného uživatele a přidání schvalovacího mechanizmu pro nové účty. Pokud by správce z nějakého důvodu chtěl celý účet odstranit, je zde i mechanismus zaručující smazání uživatele. Stačí zde kliknout u vybraného uživatele na tlačítko „Smazat uživatele“ a aplikace se už postará o smazání veškerých jeho údajů a dat, které s tímto účtem spojeny.



Obrázek 16: Správa uživatelských účtů

3.5.8 Můj účet

Tato poslední záložka se také jako ta předchozí nezaobírá nabídkou a s tím spojených věcí. Nicméně oproti záložce předchozí není její přístup omezen pouze pro administrátory a dostane se sem každý přihlášený uživatel. Jedná se totiž o záložku pro správu uživatelova účtu. Jsou zde tedy pole pro jeho editaci, které odpovídají polím na předchozí záložce, kde uživatelské účty spravoval administrátor. Konkrétně se jedná o uživatelské jméno, email a heslo. Pole heslo zde není z důvodu bezpečnosti vyplněno a jeho hodnota se změní jen při zadání nové hodnoty. Jak je vidět na obrázku 17, posledním elementem na stránce je kromě tlačítka uložit, ještě tlačítko odhlásit. To má na starost celkové odhlášení uživatele ze systému a zobrazení úvodní přihlašovací obrazovky.



Obrázek 17: Uživatelský účet

3.6 Implementace

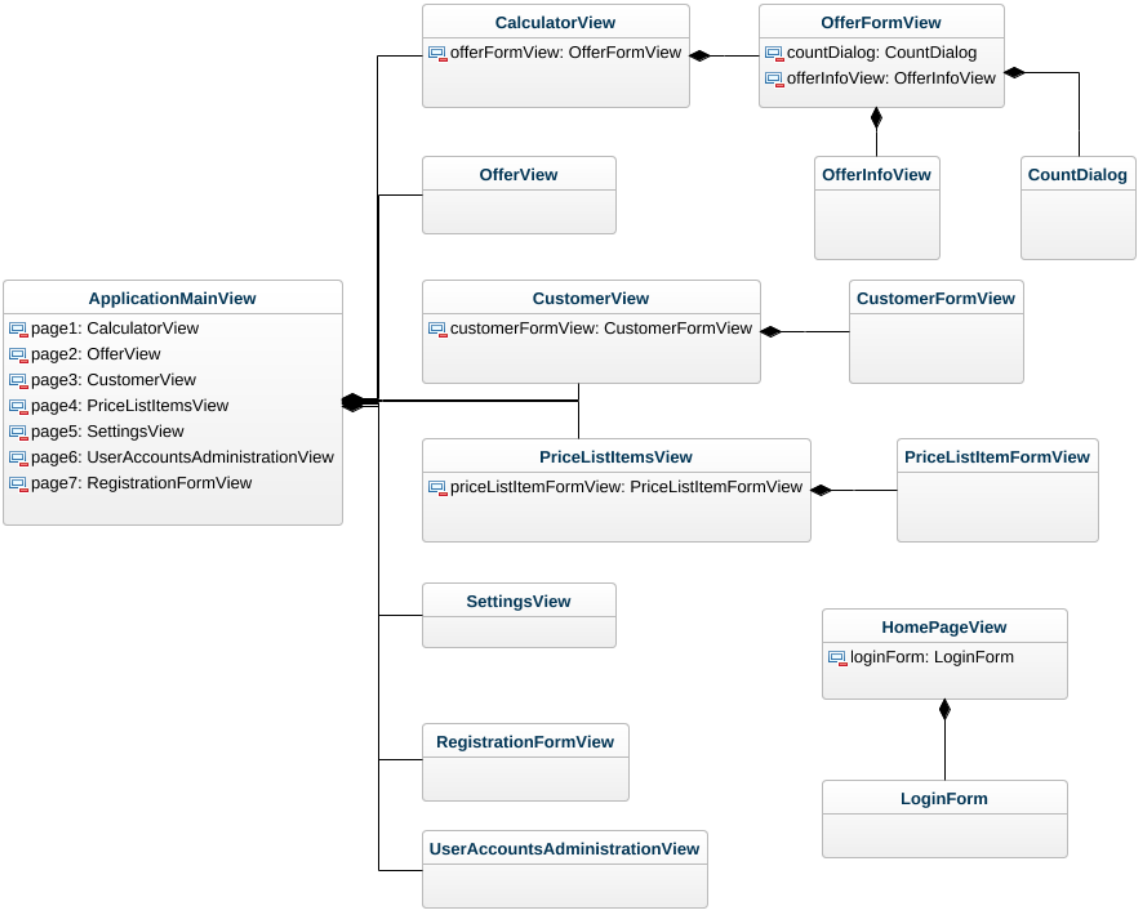
Samotná implementace je kromě závěrečného testování poslední fáze projektu. Využijí se zde všechny vývojové nástroje, ve kterých se bude tvořit aplikace, jejíž struktura a funkční stránka věci byla navržena v předchozích kapitolách. Spojí se zde tak všechny poznatky z návrhu a analýzy a zjistí se zde, zda by projekt úspěšný či nikoli. V případě že se analýze a návrhu věnovalo dostatek času a tyto fáze byly pečlivě zpracovány, tak by vývoj měl být čistě teoreticky jednoduchý a měla by to být jen otázka času, než bude výsledná aplikace hotová. V reálném případě tomu tak ale nikdy není, a i přes sebelepší návrh se najdou problémy, jejichž řešení stojí drahocenný čas vývoje. Otázkou vždy je, zda jsou tyto problémy tak vážné, že by znamenaly prodloužení či dokonce konec projektu.

První věcí při implementaci projektu je založení vlastního projektu. Tuto činnost velmi usnadňuje zvolené IDE. IntelliJ IDEA Ultimate totiž nabízí grafické rozhraní pro stažení a konfiguraci Spring boot šablony, takže programátor si akorát vybere cílovou složku, metodu kompilace a potřebné balíčky či knihovny. IDE se o vše ostatní postará a programátor tak rovnou dostane nakonfigurovaný projekt. Nemusí se tak o nic jiného starat a může začít s tvorbou samotné aplikace.

3.6.1 Diagram tříd

Pro samotnou ukázku výsledné implementace není kromě ukázky asi lepšího nástroje než diagramu tříd. Ten ukáže jak jednotlivé třídy, tak jejich a parametry a metody. Kromě tříd jsou zde uvedeny i vztahy, které mezi sebou jednotlivé třídy mají. Je zde tak vidět struktura celé aplikace. Už při implementaci, kdy byly třídy tvořeny, byly rozdělovány do balíčků. Tyto balíčky oddělují třídy a zařazují je do kategorií podle jejich určení. Mezi hlavní tři kategorie patří balíčky `model`, `view` a `service`. V balíčku `model` jsou všechny datové třídy, které by v podstatě měly odpovídat entitám z datového modelu a tím pádem není potřeba si je zde představovat. Balíček `service` obsahuje všechny třídy starající se o funkcionální stránku aplikace. Některé z těchto funkcionalit budou představeny a podrobně rozebrány v následující kapitole společně s doplňující ukázkou kódu. Nicméně ani pro ně není potřeba ukazovat diagram tříd, jelikož jsou všechny označeny notací `@Service` a do kódu, kde jsou potřeba, jsou vkládány automaticky. Posledním hlavním balíčkem je balíček `view`, ve kterém jsou třídy zajišťující grafickou stránku aplikace. Rozdělení tříd tak připomíná model MVC, ve kterém je hlavním pozitivem oddělení grafické stránky aplikace od té funkční. Což rozdělení balíčků této aplikace splňuje. [25]

Jak již bylo řečeno, nemá ukázka diagramu tříd pro balíčky `model` a `service` moc velký význam. Proto jediný balíček, pro který bude diagram tříd ukázán je balíček `view`. V tomto balíčku se tedy nacházejí třídy tvořící grafickou vrstvu aplikace. Ta je sestavena z jednotlivých komponent, které se postupně vkládají do sebe a jejich vazby mezi sebou budou dobře zobrazeny v diagramu. Jelikož všechny tyto třídy jsou potomky třídy `Component` mají tak v základu stejné parametry a metody, které v diagramu na obrázku 18 zobrazeny nebudou.



Obrázek 18: diagram tříd

Na obrázku diagramu tříd je vidět, že třída `ApplicationMainView` obsahuje nejvíce referencí na ostatní třídy. Zřejmě to bude asi hlavní třída aplikace. Zde bude ale pozornost nejdříve věnovaná třídě `HomePageView`. Tato třída představuje jen úvodní stránku aplikace, na kterou se dostane uživatel jako první. Z důvodu zabezpečení je tato stránka úplně oddělená od celého vnitřku aplikace. Obsahuje jen referenci na přihlašovací formulář. Ten je tvořen třídou `LoginForm`, což je komponenta vaadinu, která je zde jen použita, podobně jako jiné formulářové a webové komponenty na jiných stránkách.

Pokud se tedy uživatel dostane přes přihlášení a dostane se do samotné aplikace, uvidí zde stránku, která je tvořena nejobsáhlejší komponentou v popisovaném diagramu. Jedná se samozřejmě o třídu `ApplicationMainView` o které tady již byla řeč. Tato třída tedy zaštiťuje a zaobaluje celou aplikaci. Kromě komponent tvořících horní menu, obsahuje reference na všechny vytvořené stránky aplikace, které budou popisované dále.

První stránka je asi z pohledu diagramu tříd nejsložitější. Obsahuje komponentu tabulky, která je na stránce zobrazována vlevo a zobrazuje se v ní seznam položek ceníku. V diagramu je dále vidět reference na třídu `OfferFormView`, což je třída, která má na starost celý formulář pro vytvoření nabídky. Jsou zde formulářové komponenty tvořící pole pro editaci informací o nabídce. Dále jsou uvnitř této třídy reference na komponenty `OfferInfoView` a `CountDialog`. Třída `OfferInfoView` by měla mít na starost zobrazení spodního bloku, ve kterém jsou vypsané položky ceny, a nakonec cena celková. `CountDialog` je naopak třída, jejíž náplň zpočátku není vidět. Zobrazí se až ve chvíli, kdy uživatel vyvolá dialog pro změnu počtu kusů u položky objednávky.

Dalšími komplikovanějšími stránkami jsou stránky číslo tři a čtyři. Stránka číslo tři je tvořena třídou `CustomerView`. Měla by to být stránka pro správu zákazníků. Proto tato třída obsahuje komponentu tabulky a obsahuje seznam zákazníků. Dále je zde reference na třídu `CustomerFormView` jenž obsahuje formulářové komponenty pro editaci zákazníka. S třídou `PriceListItemView`. Ta opět obsahuje komponentu tabulky, tentokrát ale pro seznam položek ceníku. Reference na třídu, která obsahuje formulář pro editaci vybrané entity je zde také. Nicméně tentokrát se jedná o třídu `PriceListItemFormView`.

Tímto byly představeny třídy stránek, které obsahují nějakou složitější vnitřní strukturu. Zbyly zde ale třídy, které obsahují v uvozovkách jen formulářové komponenty, a není zde nějaká složitější logika. Konkrétně jde o třídy `OfferView`, `SettingsView`, `RegistrationFormView`, a `UserAccountsAdministrationView`. První z nich obsahuje seznam nabídek. Druhá slouží pro editaci nastavení. V další je uživatelské nastavení. A správa uživatelů je obsahem třídy poslední.

3.6.2 Popis stěžejních funkcionalit

Diagram tříd je jistě důležitou částí implementační dokumentace. Lze se znej dozvědět, jak vypadá celková struktura programu aplikace. Nicméně při představení složitějších funkcionalit již tento diagram nestačí a je nutné přejít k další metodě. Při popisu funkcionality nic nenahradí ukázku kódu, která je ze všech metod nejvíce vypovídající. Proto zde v této kapitole budou popsány fragmenty kódu, které zachycují hlavní funkcionality aplikace.

Už mnohokrát zde byly popisovány komponenty frameworku Vaadin. Ještě zde ale nebyla ukázána ani jejich tvorba natož použití. Proto zde do ukázek kódu byl zahrnut příklad vytvoření jednoduchého číselného pole, jeho nastavení a umístění do stránky. Jak je vidět na následujícím

obrázku, je konstrukce tohoto pole velice jednoduchá. Stačí vytvořit instanci třídy příslušné komponenty, nastavit ji různé parametry a pomocí metody `add`, kterou obsahuje každá obalová komponenta, ji umístit do nadřazeného prvku.

```
NumberField pricePerHour = new NumberField();
pricePerHour.setLabel("Cena za hodinu (Kč)");
pricePerHour.setPlaceholder("Cena za hodinu");
pricePerHour.setHasControls(true);
pricePerHour.setMin(0);
pricePerHour.setStep(0.1d);

add(pricePerHour);
```

Obrázek 19: Ukázka vytvoření číselného pole

Jedna část je tvorba samotného fieldu, dále je ale potřeba vyplněné údaje z pole z pole nějak dostat. A přesně k tomu slouží třída `Binder`, jejíž použití je na obrázku 20. Ta nám spojí celý formulář s datovou entitou a promapuje je pomocí shody názvu formulářového pole s atributem datové entity. Pokud se názvy u nějakých polí nezhodují, je možné mapování provést ručně. Nicméně je pohodlnější pole pojmenovávat tak aby zde shoda byla a práci s propojením nechat na třídě „Binder“, která to udělá automaticky. Jak je vidět na příkladu, stačí třídě říct jaký typ datové entity má použít a následně ještě připojit nadřazenou komponentu v níž jsou umístěny formulářová pole. Potom už stačí instanci třídy připojit objekt datové entity do které má data ukládat a vše už by mělo správně fungovat.

```
Binder<Settings> binder = new Binder<>(Settings.class);

binder.bindInstanceFields( objectWithMemberFields: this);

Settings actualSettings = settingsService.getSettings();
binder.setBean((actualSettings == null) ? new Settings() : actualSettings);

save.addClickListener(event -> save());
```

Obrázek 20: Ukázka použití třídy Binder

Další komponentou, která je v aplikaci hojně využívána, je komponenta tabulky. Její tvorba je také poměrně jednoduchá, jak ukazují obrázek 21. Pokud je ale potřeba do tabulky vložit nějaké komponenty, jako jsou třeba tlačítka nebo odkazy, je to docela problém. Jeho řešení je však

ukázáno na následujícím příkladu. Je zde instance třídy `Grid` do které je přidáván nový sloupec s potřebnými komponenty. Na to je využita metoda `addComponentColumn`, jejíž parametr obsahuje lambda výraz, ve kterém je prováděna většina kódu. Tento výraz bude proveden pro každý řádek v tabulce a měl by vracet komponentu, která bude vložena do nového sloupce. V tomto případě jde o `Div`, který obsahuje tlačítko pro smazání a odkaz pro stažení PDF souboru. Za lambda výrazem následuje ještě metoda nastavující jméno sloupce.

```
grid.addComponentColumn(offer -> {
    Button deleteButton = new Button();
    deleteButton.setText("Smazat");
    deleteButton.addClickListener(buttonClickEvent -> offerDeleteDialog.openOfferDeleteDialog(offer));

    Anchor anchor = new Anchor();
    anchor.setHref("/offer/" + offer.getId() + "/pdf-export");
    anchor.setText("Stáhnout do PDF");
    anchor.setTarget("_blank");
    anchor.setClassName("download-pdf-button");

    Div div = new Div();
    div.add(deleteButton);
    div.add(anchor);


    return div;
}).setHeader("Akce");
```

Obrázek 21: Ukázka vložení sloupce s komponenty do tabulky

Dále zde už ale komponenty z frameworku waadin popisovány nebudou. Následující ukázka se spíše zaměří na samotnou funkcionalitu aplikace než na tvoření uživatelského rozhraní. Konkrétně jde o ukázku funkcionality tvoření PDF exportu nabídky. Je zde použita knihovna `iTextPDF`, která v Javě umožňuje tvorbu PDF dokumentů. Nejdříve je potřeba vytvořit instanci samotného dokumentu a pomocí třídy `PDFWriter` ji spojit s výstupem. Tím by měl být výstupní stream bytového pole. Poté už lze přidávat jednotlivé položky do dokumentu pomocí metody `add`. Na konci je ještě nutno dokument zavřít, aby došlo k dealokaci prostředků, to ale již není v následující ukázce kódu vidět, neboť se zde nachází ještě mnoho prvků, které byly do dokumentu vkládány.

```
Document document = new Document();
PdfWriter.getInstance(document, out);

document.open();
Font font = FontFactory.getFont(FontFactory.TIMES,  size: 24, BaseColor.BLACK);
Chunk chunk = new Chunk( content: "Nabídka elektrikářských prací", font);

Paragraph title = new Paragraph();
title.setSpacingBefore(500);
title.setSpacingAfter(20);
title.add(chunk);

document.add(title);
```

Obrázek 22: Ukázka exportu nabídky do PDF dokumentu

4 NASAZENÍ A TESTOVÁNÍ ALIKACE

Implementace samotné aplikace ještě neznamená, že je projekt dokončený. Jsou tu ještě další věci, které musí být udělány, než se vývojáři mohou přesunout na další projekt. Od běžícího programu ve vývojovém prostředí je to totiž ještě hodně daleko do funkční aplikace nasazené na produkčním serveru. Prvním krokem, který je potřeba udělat je sestavení aplikace, kterou následně bude možno nahrát na server. Aplikace sama o sobě není schopna běžet a potřebuje být připojená na nějaké datové uložiště. Konkrétně jde o databázi, kterou je potřeba vytvořit a nahrát do ní nějaké výchozí data. Ani po nahrání aplikace na server a její rozeběhnutí však práce nekončí. Než je spuštěn produkční provoz a aplikace je zpřístupněna koncovým uživatelům musí se provést testování. Aby se zaručilo, že se do produkce nedává aplikace obsahující nedodělky nebo chyby.

4.1 Sestavení aplikace a nasazení na server

K tomu aby mohla být aplikace nasazena na server, je potřeba ji sestavit do výsledného řešení tedy do souboru, který půjde na server nahrát. Jelikož se jedná o server, na kterém běží Tomcat, tak by výsledný soubor měl být typu `war`. Pokud byl u projektu použit balíčkovací nástroj maven, je sestavení poměrně jednoduché. U tohoto projektu byla navíc použita šablona pro konfiguraci souboru pom.xml, kterou Vaadin nabízí, což práci velice ulehčí a vývojáři se s tímto nastavením nemusí zabývat. Stačí použít příkaz pro zabalení, který maven po správné konfiguraci nabízí. Následně je tedy projekt sestaven a vytvoří se soubor s příponou `war`, který už lze jednoduše na server nahrát a aplikaci zde spustit.

4.2 Import dat

Aplikace však ještě nebude fungovat správně. Minimálně je potřeba ji ještě připojit na databázi, která obsahuje všechny potřebné data. K tomu je zapotřebí vytvořená databáze, jejíž nastavení a přístupové údaje se shodují s údaji uvedenými v souboru `application.properties`. V případě tohoto projektu je zde však ještě určeno, že se musí jednat o databázi typu Mysql. Důvodem je připravený import dat, který je připojený k projektu. Tento import je tvořen SQL souborem s importem jak struktury databáze, tak i samotných dat. Zejména se pak jedná o výchozí ceník a předpřipravené uživatelské účty. Uživatelské účty jdou do databáze vkládány proto, aby se zákazník mohl do aplikace vůbec dostat. Bez přístupu k vytvořenému administrátorskému účtu by do aplikace nešly přidávat noví uživatelé a aplikace by tak byla

nepoužitelná. Výchozí přihlašovací údaje jsou uvedeny v přiložené tabulce. Tyto údaje je ale možno hned při prvním přihlášení změnit v nastavení uživatelského účtu.

Uživatelské jméno	**Heslo**	**Oprávnění**
admin	admin	Administrátor
default	default	Default

Tabulka 3: Výchozí přihlašovací údaje

4.3 Testování aplikace

Pokud aplikace už běží na serveru a má připojenou produkční databázi tak je práce více méně hotová. Posledním krokem je závěrečné testování, kterým je zhodnocena připravenost aplikace přejít do produkčního režimu. Testování by se samozřejmě mělo provádět během celého vývoje a mělo zahrnovat testy jednotlivých vrstev aplikace, počínaje unit testy, přes testy integrační, až nakonec i testy uživatelského rozhraní. Nicméně u aplikace malého rozsahu, tak jako je právě tato, je možno přistoupit až k testování hotové aplikace a předchozí fáze testování zanedbat. Bude zde tedy jen představena ukázka testování uživatelského rozhraní. Pro tento účel bude použit nástroj `Selenium`, konkrétně doplněk do internetového prohlížeče, který vývojáři umožní psaní testů přímo v něm a nabízí mu tak mnoho nástrojů jak tento proces urychlit. [26]

Na přiložené ukázce je vidět obrázek jednotlivých kroků v již vytvořeného testu. Někteří ze znalejších čtenářů jistě poznali, že se jedná o test uživatelského přihlášení a odhlášení z aplikace. První kroky tohoto scénáře jsou takové inicializační. Nejdříve se otevře aplikace na zadané adrese a následně se nastaví její velikost. Nastaveno je zde rozlišení FullHD, na kterém se bude aplikace asi nejvíce využívat. Další kroky už jsou jednotlivé interakce s aplikací. Vybrání pole pro zadání uživatelského jména, následováno jeho vyplněním. Dále jsou zde podobné kroky pro vyplnění hesla a odeslaní formuláře po stisku tlačítka. Dalšími kroky se automat přepne na záložku uživatelského účtu a zase se pomocí tlačítka odhlásí. Posledním krokem je porovnání zda se opravdu zobrazila úvodní obrazovka a zda se zde opravdu nachází přihlašovací formulář. Pokud by test na konci nenašel přihlašovací formulář nebo by v jakémkoli svém kroku nenašel příslušný element, test by selhal a aplikace by tak nebyla připravena k produkci pro zákazníka. Takovýchto testů bylo samozřejmě vytvořeno několik, aby byla otestována většina funkcionalit aplikace.

	Command	Target	Value
1	*open*	/	
2	*set window size*	1920x1080	
3	*click*	css=#vaadinLoginUsername > input	
4	*type*	css=#vaadinLoginUsername > input	admin
5	*click*	css=#vaadinLoginPassword > input	
6	*type*	css=#vaadinLoginPassword > input	admin
7	*click*	css=vaadin-button	
8	*click*	xpath=//vaadin-tab[contains(.,'Účet')]	
9	*click*	xpath=//vaadin-button[contains(.,'Odhlásit se')]	
10	*assert element present*	css=vaadin-login-form-wrapper	

Obrázek 23: Selenium test přihlášení a odhlášení uživatele

ZÁVĚR

Návrh a implementace webového systému je velice komplexní proces vyžadující značné úsilí a notnou dávku času. To se projevilo i při tvorbě této práce, která se zabývá právě vývojem takovéhoto systému pro tvorbu rozpočtů elektrikářských prací.

Před vlastní implementací systému pro tvorbu rozpočtu elektrikářských prací byly představeny vybrané softwarové řešení pro tvorbu rozpočtů a cenových kalkulací. Byly zde popsány jejich výhody, oblasti využití společně s cílovou skupinou uživatelů, a nakonec i moderní trendy v jejich použití. Důležitou součástí teoretické části je také rešerše, ve které byly porovnány již existující aplikace z tohoto oboru.

Dále se práce zabývá už jen praktickou částí. Prvním z několika kroků při vývoji systému, bylo stanovení požadavků. To probíhalo na principu analýzy zadání, ze kterého se získala sada jak funkčních tak nefunkčních požadavků. Druhým krokem návrhu projektu byl výběr použitých technologií, které značně ovlivnily vývoj aplikace. Dalšími fázemi návrhu byly návrhy datového modelu a uživatelského rozhraní. Datový model určoval strukturu uložených dat v databázi, zatímco návrh uživatelského rozhraní ukazoval, jak bude aplikace vypadat ze strany uživatele.

Po fázích návrhu následoval krok samotné implementace. V této kapitole nebyl popisován její přesný postup. Spíše zde byly představeny základní komponenty aplikace a byla zde uvedena ukázka použití vybraného frameworku. Nechybí zde ani vysvětlení stěžejních funkcionalit systému, kterými může být například export nabídky do PDF.

Závěrečná část práce byla věnována nasazení výsledného softwarového systému na server. Byly zde tak popsány veškeré kroky, které je potřeba provést pro nasazení aplikace na produkční server. Než byla však aplikace dokončena, zbývalo ji ještě otestovat pomocí testů uživatelského rozhraní, které by měly detekovat možné problémy před samotným zpřístupněním systému zákazníkovi.

Výsledkem této práce je komplexní webová aplikace, schopná konkurovat podobným systémům v tomto oboru. Stanovené požadavky na aplikaci byly bez jediné výjimky naplněny a samotné cíle této práce tak byly splněny v plném rozsahu.

POUŽITÁ LITERATURA

[1] 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.

[2] GROFF, James R. a Paul N. WEINBERG. *SQL: kompletní průvodce*. Brno: CP Books, 2005. Programování (CP Books). ISBN isbn80-251-0369-2.

[3] CARNELL, John. Spring microservices in action: a multiplatform approach to building chatbots. Shelter, Island, NY: Manning Publications Co., 2017. ISBN 16-172-9398-9

[4] CARNELL, John. Beginning spring boot 2: applications and microservices with the spring framework. New York, NY: Springer Science Business Media, 2017. ISBN 978-148-4229-309

[5] Rozpočtování pomocí počítače | ASB Portal. Portál ASB [online]. Jaga Media, 2008 [cit. 2020-08-24]. Dostupné z: https://www.asb-portal.cz/architektura/rozpoctovani-pomoci-pocitace

[6] KONCES: UŽIVATELSKÁ PŘÍRUČKA. In: *KONCES, spol. s r. o.* [online]. Brno, 2011 [cit. 2020-08-24]. Dostupné z: http://www.konces.cz/download.php?23

[7] KONCES: rozpočty - objednávka - nový uživatel. In: KONCES, spol. s r. o. [online]. Brno, 2011 [cit. 2020-08-24]. Dostupné z: http://www.konces.cz/cz/objednavka/obj-konces/kon-novy/

[8] SELPO - OCEP: rozpočty a kalkulace montáží elektro a voda topení plyn. In: SELPO - OCEP [online]. Broumy, c1991-2020 [cit. 2020-08-24]. Dostupné z: https://www.selpo.cz/selpo/index.htm

[9] OCEP: Ceník programu včetně ceníku montážních prací 21M, 22M. In: SELPO - OCEP [online]. Broumy, c1991-2020 [cit. 2020-08-24]. Dostupné z: https://www.selpo.cz/selpo/ocep_cenik.htm

[10] PLECHÁČ, Jan. Software pro návrh rozpočtu elektrikářských prací. Pardubice, 2010. Bakalářská práce. Univerzita Pardubice, Fakulta elektrotechniky a informatiky. Vedoucí práce Fikejz Jan.

[11] *MySQL Workbench* [online]. Oracle Corporation, c2020 [cit. 2020-08-24]. Dostupné z: https://www.mysql.com/products/workbench

[12] Vodopádový model (Waterfall model). In: ManagementMania: Sociální síť pro business [online]. c2011-2016 [cit. 2020-08-24]. Dostupné z: https://managementmania.com/cs/vodopadovy-model-waterfall-model

[13] Spring Boot - Introduction. In: Tutorialspoint [online]. c2020 [cit. 2020-08-24]. Dostupné z: https://www.tutorialspoint.com/spring_boot/spring_boot_introduction.htm

[14] Desktop Applications vs Web Apps. In: Avesta Group [online]. [cit. 2020-08-24]. Dostupné z: https://www.avestagroup.net/DetailsEN.aspx?PostID=1006&CataType=5&CataID=1006

[15] Java Introduction. In: W3Schools: Online Web Tutorials [online]. c1999-2020 [cit. 2020-08-24]. Dostupné z: https://www.w3schools.com/java/java_intro.asp

[16] Java: Resources for Students, Hobbyists and More. In: Java [online]. c2020 [cit. 2020-08-24]. Dostupné z: https://go.java/

[17] Why Spring? In: Spring [online]. VMware, c2020 [cit. 2020-08-24]. Dostupné z: https://spring.io/why-spring

[18] Spring Boot. In: Spring [online]. VMware, c2020 [cit. 2020-08-24]. Dostupné z: https://spring.io/projects/spring-boot

[19] JavaScript Tutorial. In: W3Schools: Online Web Tutorials [online]. c1999-2020 [cit. 2020-08-24]. Dostupné z: https://www.w3schools.com/js/

[20] Vaadin: Full stack framework for building web apps in Java. In: Vaadin [online]. c2020 [cit. 2020-08-24]. Dostupné z: https://vaadin.com/

[21] GWT: Overview. In: GWT: Productivity for developers, performance for users [online]. [cit. 2020-08-24]. Dostupné z: http://www.gwtproject.org/overview.html

[22] IntelliJ IDEA: The Java IDE for Professional Developers by JetBrains. In: JetBrains: Developer Tools for Professionals and Teams [online]. c2000-2020 [cit. 2020-08-24]. Dostupné z: https://www.jetbrains.com/idea/

[23] Why MySQL? In: MySQL [online]. c2020 [cit. 2020-08-24]. Dostupné z: https://www.mysql.com/why-mysql/

[24] DataGrip: Features and Screenshots. In: JetBrains: Developer Tools for Professionals and Teams [online]. c2000-2020 [cit. 2020-08-24]. Dostupné z: https://www.jetbrains.com/datagrip/features/

[25] BERNARD, Borek. Úvod do architektury MVC. In: Zdroják: o tvorbě webových stránek a aplikací [online]. 2009, 7.5.2009 [cit. 2020-08-24]. Dostupné z: https://www.zdrojak.cz/clanky/uvod-do-architektury-mvc/

[26] The Selenium Browser Automation Project: Documentation for Selenium. In: The Selenium Browser Automation Project [online]. c2013-2020 [cit. 2020-08-24]. Dostupné z: https://www.selenium.dev/documentation/en/