FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Webová aplikace, informační systému pro posilovnu Fitness Power
Student:	Patrik Kubec
Vedoucí:	Ing. Stanislav Kuznetsov
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2019/20

Pokyny pro vypracování

Cílem této práce je návrh, implementace a vývoj informačního systému pro pracovníky posilovny Fitness Power.
1) Proveďte analýzu požadavků na informační systém a use case potřeb pro posilovnu Fitness Power.
2) Seznamte se s webovými technologiemi pro vývoj webových aplikací, zejména s technologií Java Spring a webovým frameworkem Vaadin 10+.
3) Na základě analýzy požadavků navrhněte vhodnou architekturu, navrhněte vhodné databázové řešení pro ukládání dat, případně zvolte další vhodné webové technologie pro potřeby aplikace.
4) Implementujte prototyp aplikace. Řešení řádně otestujte a zdokumentujte.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 23. ledna 2019

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Webová aplikace, informační systém pro posilovnu Fitness Power

Patrik Kubec

Katedra softwarového inženýrství
Vedoucí práce: Ing. Stanislav Kuznetsov

14. května 2019

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo"), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 14. května 2019 ……

Odkaz na tuto práci

Kubec, Patrik. *Webová aplikace, informační systém pro posilovnu Fitness Power*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2019.

Abstrakt

Tato bakalářská práce se zabývá vytvořením funkčního prototypu webové aplikace pro fitness studio. Tento prototyp bude v budoucnu v rámci diplomové práce doplněn a plně přizpůsoben fitness studiu. Aplikace bude využívána zejména zaměstnanci firmy a bude sloužit jako nástroj pro evidenci zboží, správu pracovníků, přehled financí a mnoho dalšího. Smyslem aplikace je zlepšit kvalitu práce na pracovišti. Aplikace slouží jako náhrada za papírovou evidenci a tím urychluje a zpříjemňuje práci ve firmě. Díky snadnému a přehlednému ovládání je usnadněno evidování příchodu a odchodu zákazníků ze studia. Pro vytvoření práce jsou použity technologie Spring Boot, Spring Data, Vaadin Framework, Vaadin Flow, Maven, H2 database.

Klíčová slova webová aplikace, informační systém, fitness studio, funkční prototyp, Spring Framework, evidence zboží, správa zaměstnanců

Abstract

This bachelor thesis deals with creation of functional prototype of web application for fitness studio. This prototype will be in future added and fully adapted for fitness studio in case of thesis. Apllication will be used especially by employees of company and will be used like a tool for evidence of goods, staff management, finance overview and many other things. The purpose of the application is to improve quality of work in the workplace. Application serves as a substitute for paper records and because of that it makes work faster and more enjoyable. Thanks to easy and clearly control of application it is easier recording of movement customers in fitness studio. Technologies Spring Boot, Spring Data, Vaadin Framework, Vaadin Flow, Maven, H2 database are used to create the work.

Keywords web application, information system, fitness studio, functional prototype, Spring Framework, records of goods, records of employees

Obsah

Seznam obrázků

Úvod

Většina firem má v dnešní době svůj informační systém, nebo webovou aplikaci pro správu, řízení a přehled dění na pracovišti. Takovýto typ webové aplikace má za úkol evidovat zboží na firemním skladu, spravovat své zaměstnance, rozdělovat zaměstnance do rolí podle jejich pravomocí, vést údaje o financích a mnoho dalšího. Uchovávání těchto údajů v papírové podobě je v současnosti zbytečné. Vyhledávání nebo vytváření statistik je v takovém případě příliš složité a zdlouhavé.

Existují programy, které dokáží řešit tyto problémy, avšak nejsou uzpůsobeny přímo pro potřeby firmy, pro niž je tato aplikace vytvořena. Menší fitness studio, pro které je určena tato aplikace má většinou jednoho zaměstnance, který je aktivně za pultem a zároveň se věnuje klientům. Aplikace se bude zaměřovat zejména na zjednodušení a zrychlení práce zaměstnanců firmy. Aplikace by měla být intuitivní a jednoduchá na ovládání. Pracovník tak může rychle reagovat na nového zákazníka a přitom mít stále přehled o dění na pracovišti. Přehlednost mu mimo jiné zajistí aktuální statistka o počtu klientů, kteří v dané době využívají služeb fitness studia. Statistika ve formě přehledného grafů by měla být viditelná téměř ve všech částech aplikace.

Podněty k vytvoření této práce vznikly na základě průběžného pozorování chodu studia a na základě rozhovorů se zaměstnanci. Hlavním motivem pro tvorbu a konkrétní podobu aplikace pak byly vysledované nedostatky ve fungování podniku. Specifické požadavky, jednoduchý a čistý vzhled a snadné ovládání aplikace jsou často věci, které se těžko slučují dohromady.

Část I

Teoretická část

Kapitola 1

Cíl práce

Cílem této bakalářské práce je vytvořit návrh a implementovat funkční prototyp webové aplikace, která pomůže pracovníkům firmy Fitness Power, sídlící v Praze 4, Pujmanové 1220/6, Pankrác 140 00, zlepšit kvalitu a rychlost práce na pracovišti. Řešení bude realizováno hlavně pomocí webových frameworků Java Spring a Vaadin.

Cílem teoretické části práce je seznámení se s technologiemi pro vývoj webových aplikací, zejména s technologií Java Spring a webovým frameworkem Vaadin verze 10 a vyšším. Tyto technologie jsou často používány pro vývoj aplikací tohoto typu.

V části analytické bude provedená analýza požadavků na informační systém a specifických případů užití. Tato analýza má za úkol zjistit, jaké konkrétní požadavky mají pracovníci na aplikaci a posoudit jejich realizovatelnost. Díky tomu bude aplikace vypracovaná přesně na míru fitness studia.

Praktická část se bude zabývat návrhem a implementací prototypu. To zahrnuje i výběr vhodného databázového řešení pro ukládání dat, případně zvolit další vhodné webové technologie, které by mohly pomoci při vývoji aplikace, to vše na základě analýzy požadavků. Součástí praktické části je i otestování hlavních částí aplikace a její dokumentace.

Kapitola 2

Rešerše

2.1 Webová aplikace

2.1.1 Výhody webové aplikace

Vytvoření webové aplikace přizpůsobené požadavkům firmy má jednoznačně své výhody. Přístup k datům je možný odkudkoliv a více uživateli najednou. Přehlednost záznamů je mnohem větší než u papírové evidence. Takováto webová aplikace umístěná na *cloud* je nezávislá na konkrétním počítači. To souvisí s nároky kladené na lokální počítač. Díky umístění na *cloud* jsou nároky na hardware počítače minimální. Je možné vést aktuální a přehlednou statistiku o návštěvnosti zákazníků. Technologie se vyvíjí závratným tempem a tak i požadavky firmy a zákazníků. V reakci na tyto skutečnosti je možné v budoucnu aplikaci dále vyvíjet a rozšiřovat.

2.1.2 Nevýhody webové aplikace

Skoro každé řešení má své výhody, ale zároveň i nevýhody. V případě webové aplikace se jedna z výhod (přístupnost z internetu) může změnit i v nevýhodu. Taková aplikace je právě závislá na internetu a pokud připojení není dostatečně kvalitní, může být práce s takovou aplikací nepříjemná. V dnešní uspěchané době nikdo nechce trávit svůj drahocenný čas čekáním na odpověď serveru. Další nevýhodou je závislost na poskytovateli služeb. Pokud je program nainstalovaný lokálně, dá se využívat nadále i když už skončila jeho podpora.

2.2 Srovnání různých řešení a technologií

Tato kapitola obsahuje popis různých řešení a technologií, které připadaly v úvahu použít pro vytvoření aplikace. Jsou zde rozebrány výhody, nevýhody a hlavně důvody, proč byly zvoleny právě výše zmíněné technologie.

Jako jedno z mnohých řešení se nabízel framework Angular, který je od firmy Google a je dnes nejpoužívanějším webovým frameworkem. Nedávno vydaný Framework ReactJS je konkurencí Angular.js, proto se také objevil na seznamu možných kandidátů. Nakonec byl však vybrán framework Vaadin, který se v posledních letech dostává do povědomí vývojářů, a to díky rychlému vývoji tohoto frameworku.

2.2.1 Angular

Angular je webový framework vyvíjený firmou Google. Předchůdce Angularu je AngularJs. Architektura Angularu je podobně jako architektura Vaadin postavena na sadě komponent, jejichž základem je programová třída. Na rozdíl však od frameworku Vaadin, pro práci s Angularem je zapotřebí naučit se mnoho technologií, které Angular využívá jako například HTML a CSS. Angular aplikace se píší v jazyce TypeScript, což je nadstavba JavaScriptu. Potřebné nástroje, které jsou zapotřebí pro vývoj v Angularu jsou nástroje spjaté s Node.Js.[1]

2.2.2 React

Webový framework React je knihovna od společnosti Facebook. Hlavní rozdíl od ostatních frameworků, jako například Vaadin nebo Angular spočívá v jeho zaměření. React se soutředí pouze na jednu konkrétní vrstvu z klasické MVC architekrury a tou je *view*. Vrstva prezentující data uživateli. Ostatní frameworky jsou tzv. úplné a dají se použít i na tvorbu *backendu*.

Základem Reactu, tak jako u většiny jiných frameworků, jsou komopnenty. V případě Reactu se jedná o zapouzdřené HTML elementy s vlastní funkcionalitou. Složením více těchto komponent dohromady a rozmístěnín po stránce, se tvoří tzv. UI aplikace. Stejně jako u jiných webových frameworků i zde mají komponenty své *properties* a samy si spravují svůj vnitřní stav. Framework založený na sestavování komponent dobře spolupracuje s dalšími knihovnami podobného zaměření.

Jako programovací jazyk se používá JavaScript. Ve většině případů je to čistá verze JavaScriptu. Při tvorbě UI je dále potřeba znát HTML a CSS.[2]

2.2.2.1 Využití Reactu

Jak již bylo zmíněno, React je velmi specifickým frameworkem se zaměřením pouze na UI, proto se využívá ve spolupráci s dalšími knihovnami s rozdílnou architekturou a přístupem. React se poté stane pouze nástroj na vykreslení prezentační vrstvy. Případy využití jsou:

- tvorba single-page aplikací (SPA)
- vykreslení webových stránek na straně serveru

- vykreslování statických webových stránek. Nejčastěji ve spojení s knihovnou Gatsby
- vytvoření nativních mobilních aplikací pomocí JavaScriptu

2.2.3 Java Spring

Java Spring je aplikační framework využívající návrhového vzoru IoC (inversion of control) a technologie DI (dependency injection). IoC přesouvá zodpovědnost za vytváření a provázání objektů z aplikace na framework. Základní funkce Java Spring frameworku mohou být použity jakoukoliv java aplikací. V dnešní době je Spring alternativa k EJB (Enterprise JavaBeans) modelu. Spring implementuje interface repository a paging. Například CudRepository, který nabízí sofistikované funkcionality okolo CRUD pro menežovatelné entitní třídy. Spring zahrnuje velké množství modulů (MVC-framework, data access framework, transaction management, spring security), které nabízí širokou škálu služeb. Některé již byly vyjmenovány, ale za zmínku stojí také těstování. Spring zahrnuje testovací třídu podporující psaní unit testů a integračních testů. Java Spring se tedy pro svoji obecnou stadardizaci hodí pro psaní backendu aplikace.

2.3 Srovnání databázových systémů

Pro ukládání dat na vzdálený server je nutné zvolit, jakou relační databázi použít. Pro výběr relační databáze je k dispozici hned několik kandidátů. S omezením na open source software a a přihlédnutím ke skutečnosti, že se v tomto případě jedná o malou firmu, která do budoucna neplánuje další rozvoj, by bylo nejvhodnější užití databáze H2 nebo Derby. Na rozdíl od komerčních databází nejsou tolik škálovatelné. Nejznámějšími databázemi jsou:

- PostgreSQL
- MySQL
- SQLite
- Oracle
- Derby
- H2

2.3.1 PostgresSQL

PostgresSQL je objektově-relační databázový systém (ORDBMS) a je pod licencí MIT. Jedná se tedy o free a open source software. Primárně je PostgreSQL vyvíjen pro unixové systémy, nicméně existují i balíčky pro windows.

V dnešní době je PostgreSQL nejpoužívanějším a nejoblíbenějším free databázovým systémem. Z funkcí podporuje například Triggery, Indexy nejen nad výrazy, ale i nad částí tabulky a MVCC (Multi-Version Concurrency Control). MVCC každému uživateli zpřístupňuje snapshot databáze. Pokud se provádí změna, ostatní uživatelé ji vidí až poté, co je potvrzená transakce.

2.3.2 MySQL

Mezi další free databázový systém patří MySQL. Existuje i placená verze, která umí víc funkcionalit. Dříve bylo MySQL optimalizováno především pro rychlost i za cenu některých zjednodušení. Donedávna nepodporovalo například triggery, pohledy, nebo uložené procedury. Dnes spadá MySQL pod firmu Oracle. Jelikož má Oracle svůj vlastní databázový systém se stejným názvem Oracle, je velice nepravděpodobný příchod větších aktualizací.

2.3.3 Oracle

Z již zmíněných databází je právě Oracle tou nejlepší možností co se optimalizací týče. Oracle licence je však finančně náročná. Oracle je také nejstarší, a proto má v něčem lehce odlišné standardy jak zachází s uživateli, daty či schématy. Také striktně nedodržuje SQL normy.

2.3.4 H2, Derby

Jiným druhem databáze je H2 databáze. Celá je napsaná v javě a může být vložena do jakékoliv java aplikace, nebo může běžet jako client-server. H2 je open source software pod licencí „Eclipse Public License“. Hlavní API je SQL a JDBC, nicméně tato databáze podporuje také PostgreSQL ODBC driver a může se chovat jako PostreSQL server. H2 se hodí především pro lokální použití a pro systémy, kde nepřistupuje k datům mnoho uživatelů najednou. Databáze Apache Derby je velmi podobná H2 databázi a nabízí stejné funkcionality i použití.

2.4 Vaadin

Kapitola Vaadin je obsahově bohatější, jelikož byla vybrána právě technologie Vaadin jakožto stěžejní frontendový framework pro aplikaci. Podklady pro informace uvedené v této kapitole jsou čerpány z diplomové práce a z původní Vaadin dokumentace.[3][4]

2.4.1 Programátorský pohled

Vaadin si klade za cíl splnit požadavky uživatelů a programátorů. Mezi tyto požadavky spadá nástroj, jehož pomocí půjde snadno a rychle vytvářet webové

aplikace, které jsou stále žádanější. Zároveň je vyvíjena snaha mít na platformě nezávislý jazyk, který by byl snadno udržitelný a dostatečně výkonný.

2.4.2 Uživatelský pohled

Uživatelský pohled se zaměřuje na jiné věci, především aby byly výsledné aplikace čisté, přehledné a jednoduché na používání. Tyto problémy však neřeší zvolený programovací jazyk, ale samotný vývojář, který musí porozumět potřebám běžného uživatele a vyvinout adekvátní a uživatelsky příjemné prostředí.

2.4.3 Vaadin jako knihovna

Vaadin framework je knihovna, která obsahuje sadu nástrojů a otestovaných komponent. Tyto komponenty jsou připraveny rovnou k použití, čímž je usnadněn vývoj velkých webových aplikací. Samotný Vaadin kód se píše v jazyce java. Jak již bylo zmíněno, Vaadin je vyvíjen zejména pro jednoduchou tvorbu a údržbu webových uživatelských systémů. Webové aplikace jsou rozděleny na serverovou a klientskou část. Síla tohoto frameworku spočívá především na serverové části.

Velká a nepopíratelná výhoda je možnost psát celou webovou aplikaci kompletně v jazyku Java a nezabývat se HTML, CSS, JavaScriptem nebo JSONem. Programátor tedy nepotřebuje ani minimální znalosti těchto jazyků, aby byl schopný vyvinout webovou aplikaci ve frameworku Vaadin. To však neznamená, že by Vaadin nepodporoval tyto technologie, právě naopak. Plného potenciálu tohoto frameworku se dosáhne až tehdy, když se programátor plně naučí pracovat s těmito technologiemi a dokáže je používat napříč celým projektem.

Vaadin se snaží ubírat cestou, která vyžaduje minimální znalosti k tomu, aby byl vývojář schopný pracovat s tímto frameworkem. Všechny komponenty jsou napsány tak, aby podporovali širokou škálu funkcionalit. Může se však stát, že programátor potřebuje po komponentě vlastní funkcionalitu. V takovémto případě je možné tuto funkcionalitu dodatečně přidat (doprogramovat). Všechny komponenty by měly mít jednoduchý *interface* uzpůsobený přesně pro tyto účely.

2.4.4 Klientská část

Tato část aplikace spouští pouze JavaScript v internetovém prohlížeči bez použití jakýchkoliv *pluginů*. Aby bylo *view* vykresleno, je použit framework GWT.[5] Zdrojový kód je zkompilován do JavaScriptu a zobrazen v prohlížeči. Některé části jsou vykresleny jako HTML5. Skutečnost, že aplikace spouští pouze JavaScript vede k celkové kompatibilitě mezi různými internetovými prohlížeči. To také přispívá k zjednodušení vývoje aplikace. Celý výsledek

a jeho prvky jsou díky GWT automaticky přeloženy do několika rozdílných verzí. Každý internetový prohlížeč má různé požadavky a funkce. Programátor tudíž nemusí vyvíjet několik verzí pro jednotlivé prohlížeče a výsledné zobrazení je všude stejné. Vaadin tedy používá framework, který umí vytvářet sofistikované komponenty uživatelského rozhraní. Veškerá logika komponent je prováděna v prohlížeči.

Komponenty vytvořené a přeložené tímto způsobem činí aplikaci snadno rozšířitelnou o *addony* (doplňky) třetích stran.

2.4.5 Vaadin motivy

Vaadin striktně odděluje rozdělení uživatelského rozhraní a jeho vzhled. Tímto krokem je umožněno odděleně vyvíjet tyto dvě části. Vaadin pro svůj vzhled uživatelského rozhraní používá takzvané motivy. Ty jsou uloženy v souborech s názvem *themes* a definují celkový vzhled pomocí CSS a HTML formou šablon. Základní motivy jsou uloženy přímo v jádře frameworku. Tyto motivy se pro svou jednoduchost a eleganci dostaly do povědomí vývojářů velmi rychle. Vaadin však umožňuje jejich široké rozšíření, nebo dokonce možnost přepsat celé motivy vlastními styly.

2.4.6 Vaadin architektura

2.4.7 Popis architektury

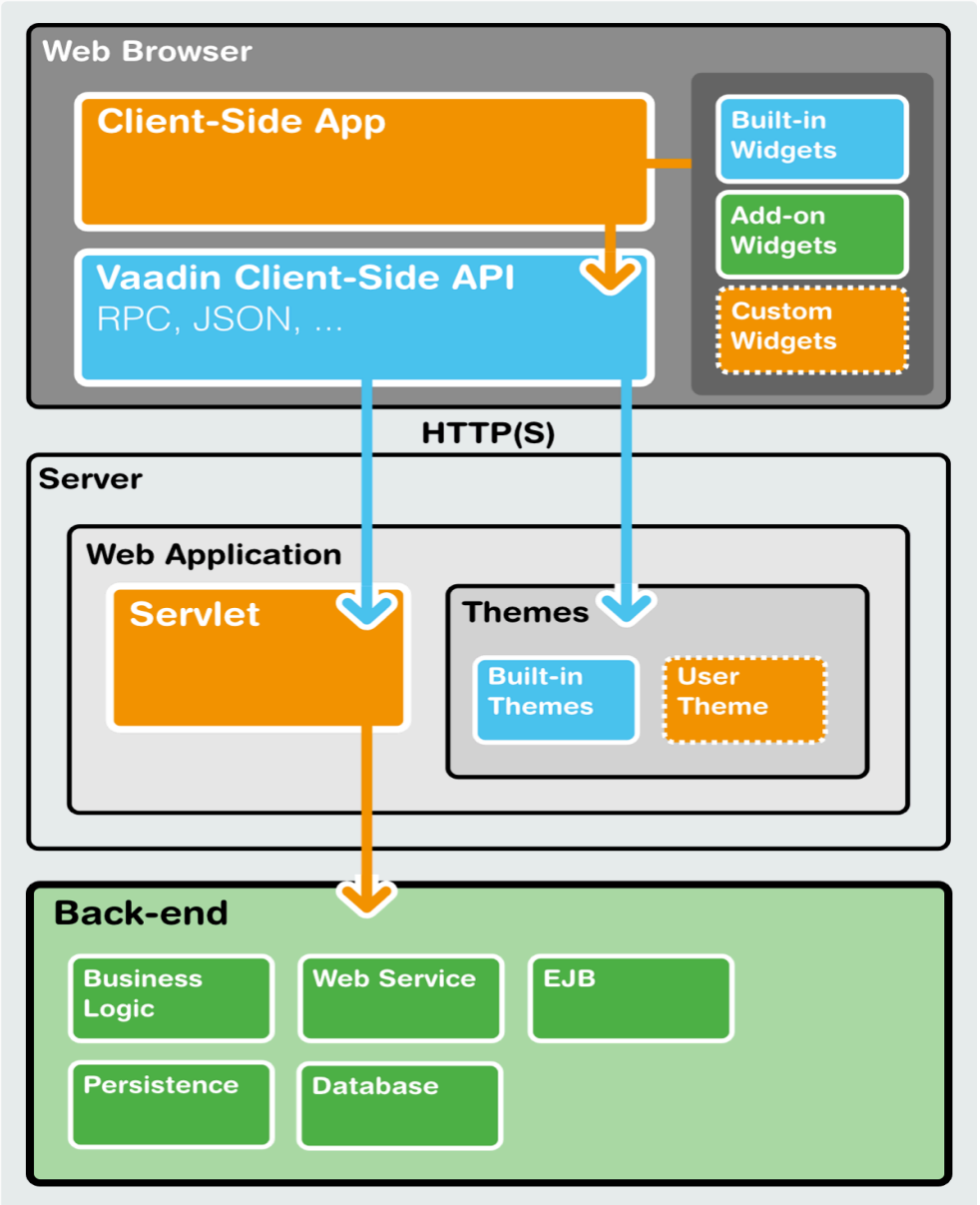
- uživatelské rozhraní – uživatel má k dispozici rozhraní, přes které komunikuje s aplikační vrstvou (logika, data). Tato vrstva je zodpovědná za zobrazení v prohlížeči v závislosti na uživatelském vstupu.

- komponenty – komponenty jsou stavebními kameny, ze kterých se skládá uživatelské rozhraní. *Widget* je prvek zobrazen ve webovém prohlížeci na klientské straně. Každý *widget* je výsledkem vykreslení své komponenty.

- client-side engine – tento engine má na starosti vykreslení všech komponent uživatelského rozhraní do formy widgetů. Vykreslování probíhá na základě oboustranné komunikace mezi klientskou a serverovou částí aplikace. Komunikace je zprostředkována pomocí HTTP a HTTPS *requestů*.

- vaadin servlet – servlet je vrstva mezi HTTP požadavkem od webového klienta a aplikací na jiném serveru. Každý klient má od servletu určen svůj prostor, který je přidělen na základě přijatého požadavku.

- motivy – tato část architektury obsahuje jednotlivé CSS styly jak pro celou aplikaci, tak pro jednotlivé komponenty. HTML šablony pro jednotlivé komponenty jsou zde uloženy také. Dále zde mohou být různé obrázky, nebo grafické soubory.



Obrázek 2.1: Architektura aplikace ve frameworku Vaadin

- back-end – vrstva, která obsahuje veškerou *bussines* logiku aplikace oddělenou od prezentační vrstvy.
- klientské aplikace – tak se nazývají moduly, které běží samostatně ve webovém prohlížeči. Vaadin podporuje vytváření takovýchto modulů, které mohou používat stejné widgety, motivy a aplikační vrstvu jako serverové aplikace.

2.4.8 Vaadin eventy

Vaadin stejně jako jiné webové java frameworky má implementovaný *event-driven model*, jehož úkolem je interagovat s uživatelským rozhraním. Interakce spočívá v tom, že uživatel vyplní formulář a zmáčkne tlačítko. Touto akcí se změní stav v uživatelském rozhraní a *observer* informuje server o změně stavu. Vaadin používá jeden z mnoha návrhových vzorů pro tyto akce. Tento návrhový vzor se nazývá „Observer“ a slouží k přenosu dat z vrstvy uživatelského rozhraní do vrstvy datové. Návrhový vzor „Observer“ je rozdělen na dvě části. První část je takzvaný posluchač, který čeká na událost (stisknutí tlačítka) a druhá část je objekt, který tyto události generuje a posílá.

Každý posluchač se musí zaregistrovat do seznamu posluchačů. Registrace nejčastěji probíhá pomocí metody addListener(), nebo její podobné verze. V případě tlačítka se jedná o metodu addClickListener(). Posluchač musí nabízet rozhraní, které implementuje metodu. Tato metoda je poté volána objektem (zdroj události). Různé komponenty mají implementovány různé rozhraní pro posluchače. Na obrázku je zobrazena ukázka zdrojového kódu.

```
final Button button = new Button("Uložit");
button.addClickListener(new Button.ClickListener() {
    public void buttonClick(ClickEvent event) {
        button.setCaption("Uloženo");
    }
});
```

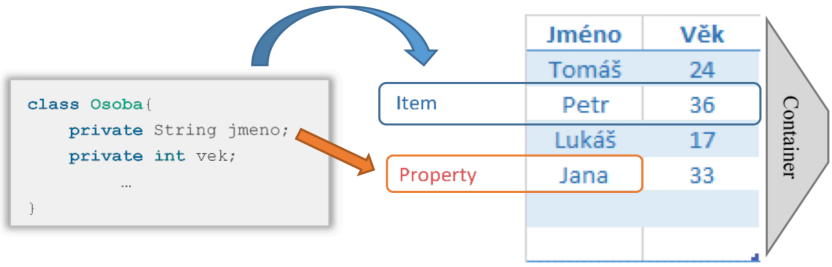
Obrázek 2.2: Obsluha události click

2.4.9 Data binding

Vaadin disponuje celou třídou pro data *binding*. Pro přímý přístup komponent k modelu má Vaadin definovaný několik rozhraní. Třída propojuje data z modelu a konkrétní prvek ve *view*. Prvek pak jednoduše zobrazuje data z modelu, které lze například upravovat.

Datový model je rozdělen na tři části. Nazývají se Item, Container a Property a každá z těchto částí představuje jednu část tabulky. Jedná se o celou tabulku, řádek tabulky a její buňku.

Tento datový model je použit ve velké části Vaadin komponent. Zejména se jedná o komponenty, které nabízí uživateli nějaký druh vstupu. Jako příklad lze uvést *TextField*, *NumberField*, *Grid* a spoustu dalších komponent. Propojení těchto prvků s datovým modelem přináší programátorovi snadnější vývoj a přehlednou práci s komponenty.



Obrázek 2.3: Zachycení data-binding mezi view a modelem

Obrázek 2.3 zachycuje návrhový vzor data-binding. V tomto případě je datový model třída „Osoba“. Ve třídě jsou dvě privátní proměnné „jmeno“ a „vek“. Řádek tabulky tvoří Item, který má své Property (jmeno, vek). Všechno dohromady tvoří objekt. Komponenta zobrazuje obsah kontejneru, protože je kontejner nastaven jako zdroj informací pro danou komponentu. Jakákoliv akce měnící obsah kontejneru se okamžitě zobrazí ve *view*, protože se komponenta automaticky překreslí. Toto funguje i opačným směrem, jestliže uživatel změní data v uživatelském rozhraní, změna se projeví také v modelu.

Vaadin data-binding obsahuje také celou škálu validací. Při svazování komponenty s modelem se dá na jednotlivé prvky nasadit validátor dat, který je buď *defaultní*, anebo si ho může vývojář upravit podle své potřeby. Validace však není jediná věc, kterou data-binding podporuje. Při převádění dat z komponenty do modelu se občas hodí vstupní data konvertovat do požadovaného datového typu. Například ze *Stringu* do *Integeru*. Vaadin nabízí tuto možnost v podobě vlastních konvertorů. Nadále platí, že lze tyto konvertory předefinovat podle potřeby.

Podpora takzvaného *Lazy loadingu* pomáhá rychlejšímu zobrazování dat. Ze serveru jsou odeslána data pouze z první stránky seznamu/gridu a další jsou načtena až poté, co uživatel přejde na další stránku v seznamu.

2.5 Zvolené řešení

Zvolené řešení daného problému, je vyvinout webovou aplikaci, která bude modulární a bude splňovat požadavky firmy. Konzultace s pracovníky firmy vedly také k volbě webového frameworku Vaadin[6] verze 10 a vyšším. Jako jedno z možných řešení se nabízel framework Angular[7], který spadá pod firmu Google a je dnes nejpoužívanějším webovým frameworkem. Webový framework je základem pro frontendovou část aplikace. Základ pro backendovou část aplikace jsem zvolil Java Spring Boot framework, který má v sobě integrovaný Vaadin Framework a to z toho důvodu, aby byla výsledná aplikace co nejvíce ucelená z pohledu použitých technologií. Další tehnologie na seznamu je H2 databáze. Použití H2 databáze je v případě této aplikace velice vhodné. Nejen že je tato databáze napsána celá v javě, ale právě díky tomu může být velmi snadno vložena do kterékoliv java aplikace.

Část II

Praktická část

Kapitola 3

Analýza

3.1 Obecný pohled

Fitness studio jakožto menší firma musí mít přehled o svých nabízených produktech, pracovnících, financích, zákaznících a současně mít způsob, jak udržovat tyto údaje přehledné a snadno dostupné. Existuje spousta kancelářských balíčků a placeného softwaru, jako jsou Microsoft Excel[8] nebo LibreOffice[9], ve kterých lze tyto údaje uchovávat. Mnohem lepším řešením je však software vytvořený přímo na zakázku, který vyhovuje potřebám firmy. Software dělaný na míru by měl obsahovat pouze takové funkcionality, které požaduje zadavatel. To vede k odstranění problémů jako jsou nepřehlednost aplikace nebo složité ovládání. Což zaměstnancům ušetří mnoho práce a zaměstnavateli ušetří čas se zdlouhavým proškolováním nových zaměstnanců s firemním softwarem.

3.2 Současný stav řešení

Tato kapitola popisuje současný stav a řešení, které společnost používá. Veškeré informace byly získány po osobním setkání a rozhovoru se zaměstnancem firmy.[10]

Firma Fitness Power má část své evidence vedenou papírově a zbylou část v desktopové aplikaci vytvořenou pouze pro platformu Microsoft Windows XP. Aplikace je nainstalovaná na firemním počítači, ale je nepřenositelná, jelikož v současnosti nemá firma instalační balíček. To nese velké riziko. Pokud by došlo k poškození firemního počítače, muselo by se přejít kompletně na papírovou evidenci a to by znamenalo velké komplikace pro firmu.

Z těchto důvodů nelze provést rozšíření stávající aplikace. Další nevýhodou nynějšího řešení je nepřístupnost k datům odkudkoliv. Jedno z možných ale pouze částečných řešení je použití různých on-line nástrojů, jako například Google Sheets[11], který umožňuje přístup více uživatelům najednou. Jediná

možnost, která zbývá je vyvinout novou aplikaci, která bude přizpůsobená požadavkům firmy.

3.3 Specifikace požadavků

Analýza požadavků se opírá o školní zkušenosti, subjektivní pozorování jakožto zákazníka fitness studia a zároveň vychází z jednotlivých sezení a konzultací s pracovníky firmy.

Požadavky na aplikaci jsou výsledkem všech předešlých bodů. Jedním z požadavků je zaznamenávání příchodu a odchodu zákazníků. To bude realizováno pomocí správy jednotlivých šatních skříněk určených pro zákazníky. Každému klientovi bude přidělena jedna konkrétní skříňka. Ke každé skříňce se bude moci přidat produkt z firemní nabídky a bude započteno vstupné. Použitá skříňka bude zobrazena v aplikaci se svým číslem, aby bylo vidět, které skříňky jsou již obsazené. Související požadavek je evidování produktů na skladu a spravování zaměstnanců. Každý produkt má svůj unikátní kód, který se zadá do aplikace. Pomocí tohoto kódu bude realizováno přidávání produktů k zákazníkům (do skříněk v aplikaci). Zaměstnanci budou evidováni v aplikaci pod svým jménem, příjmením a emailem. Mezi dalšími požadavky figuruje mimo jiné i vzhled výsledné aplikace. Vzhled by měl být jednoduchý a intuitivní pro používání. Práce s nepoutavou aplikací může být pro zaměstnance uživatelsky nepříjemná. Současný trend je vytvářet takzvané *user friendly* aplikace, které svým ovládáním napomáhají rychlé práci a snadnému zaškolování nových pracovníků.

Poslední požadavek je, aby byla aplikace přístupná přes webový prohlížeč jakožto tenký klient. Tím se stane aplikace přenositelnou a zároveň bude veškerá logika a zabezpečení na straně serveru.

3.4 Funkční a nefunkční požadavky

Funkční a nefunkční požadavky specifikují, co přesně by měla výsledná aplikace dělat a jaké funkcionality by měla pokrývat.

3.4.1 Funkční požadavky

- vytvoření nového produktu
- smazání stávajícího produktu
- upravení stávajícího produktu
- vytvoření nového zaměstnance
- smazání stávajícího zaměstnance

- upravení stávajícího zaměstnance
- vytvoření nové skříňky
- přidání produktu do skříňky
- odebrání produktu ze skříňky
- zaplacení a odebrání skříňky
- přehled financí
- vložení financí do pokladny
- odebrání financí z pokladny
- přehled zaplacených produktů
- přehled zákazníků v posilovně

3.4.2 Nefunkční požadavky

- použít framework Vaadin 10+
- použít Java Spring Boot framework
- webová aplikace

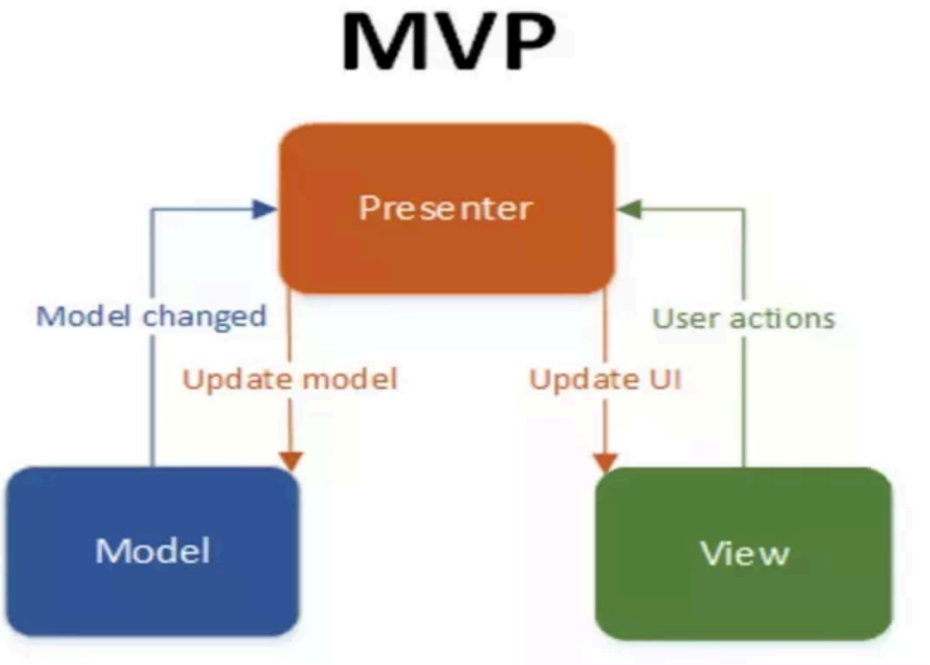
Kapitola 4

Návrh a implementace

Tato část práce obsahuje popis implementace a popis technologií, které byly použity při vytváření práce.

4.1 Architektonický styl

4.1.1 MVP



Obrázek 4.1: MVP architektura

V této sekci jsou čerpány informace o MVP architektuře z článku[12] dále využívané v této práci. Pro tvorbu uživatelského rozhraní byla zvolena architektura MVP. Tato architektura rozděluje aplikaci na tři části. Smysl tohoto rozdělení je oddělit interní data od view (to, co vidí uživatel). Komponenty jsou tyto:

- model – je rozhraní starající se o zpracování vnitřních dat, která mají být uložena do databáze, nebo jinak zpracována v uživatelském rozhraní.
- view (pohled) – je vrstva, která zajišťuje prezentaci dat. Obsahuje všechny komponenty uživatelského rozhraní.
- presenter – obsahuje veškerou logiku aplikace. Pracuje s modelem (načítá data z repositářů) a připravuje data pro zobrazení v pohledu.

MPV architektura se dá chápat jako implementace klasického modelu MVC architektury. Celkové rozdíly mezi MVP a MVC jsou malé. MVP i MVC lze implementovat více způsoby a nedá se určit, který způsob je správný. Vše záleží na konkrétní aplikaci a pohledu programátora. V MVP architekruře je umožněna komunikace mezi *view* a *presenterem*. V některých případech může view přebírat celé řízení aplikace. Poté záleží na standardech konkrétního framewroku, kterými se ovšem nemusí vývojář řídit. Z důvodu častého používání modelu MVP ve Vaadin aplikacích byla MVP architektura použita i v této práci.

4.2 Použité technologie a nástroje

4.2.1 Přehled

- spring boot
- vaadin
- git
- HTML
- CSS
- maven
- AJAX
- javaScript
- JPA
- lombok
- hibernate

4.2.2 Spring boot

Spring boot se zaměřuje na vytváření *stand-alone* aplikací a vychází ze samotného Springu. Sílou Spring bootu je především celková autokonfugurace aplikace pomocí propojených knihoven. Vývojář neztrácí čas složitou konfigurací a může rovnou psát *backendovou* část aplikace. Přímo *embedovaný* Tomcat server, nebo Jetty plugin přináší možnost spustit aplikaci jako JAR. Potřeba *deployonut* aplikaci na server jako WAR tedy odpadá.[13]

4.2.3 Vaadin

Vaadin je webový framework pro tvorbu aplikací. Pomocí zapouzdřených komponent umožňuje rychle vyvíjet uživatelské rozhraní. Vaadin aplikace se píší v jazyku Java jsou zobrazeny ve webovém prohlížeči jako HTML5.

4.2.4 Git

Git je nástroj pro správu a verzování celého projektu. Git je založen na *commitech*, díky kterým je snadné udržovat historii a dokumentaci projektu.[14]

4.2.5 HTML

HTML je značkovací jazyk používaný pro tvorbu a formátování obsahu webových stránek. Webové prohlížeče tento kód následně zobrazí. HTML se dříve používalo pro definování vzhledu stránky, ale dnes slouží pouze pro definování její struktury.

```
<dom-module id="box-view">
    <template>
        <style include="shared-styles">
        </style>

        <vaadin-vertical-layout id="layout" class="content-layout"
        theme="spacing margin">
            <top-bar id="bar" title="Skříňky">
                <vaadin-button slot="btns">btn</vaadin-button>
            </top-bar>
            <vaadin-grid id="order-grid"
            theme="orders no-row-borders">
            </vaadin-grid>
        </vaadin-vertical-layout>

    </template>
```

Zdrojový kód 1: Ukázka HTML definující rozložení elementů

4.2.6 CSS

CSS je zkratka pro kaskádové styly. Dávají vzhled jednotlivým HTML elementům. Jsou dva způsoby jak použít CSS, buď v odděleném souboru *style.css*, nebo jako součást HTML tagu. V projektu jsou tyto styly využívány pro stylizování celého UI.

```
<style>
    h1, h2, h3, h4, h5, h6 {
        margin-bottom: 0;
        margin-right: var(--lumo-space-s);
    }

    .main-layout {
        height: 100%;
        overflow-y: hidden;
        scroll-padding-bottom: 0;
    }

    .content-layout {
        height: 100%;
        overflow-y: scroll;
        scroll-padding-bottom: 0;
    }

    .Available {
        color: #2dd085;
    }

    .Full {
        color: #ffc66e;
    }

    .Broken {
        color: #f54993;
    }

</style>
```

Zdrojový kód 2: Ukázka CSS definující vzhled prvků

4.2.7 Maven

Maven je *buildovací* nástroj od společnosti Apache. Spravuje a automatizuje buildy aplikací. Každý projekt běžící na nástroji Maven je založený na tzv. Project Object Modelu jehož definice je v XML souboru s názvem pom.xml. Soubor obsahuje závislosti projektu, proces sestavení a jeho celkový popis. Maven projekt je spojený s repozitáři, ze kterých jsou automaticky stahovány artefakty a instalovány do projektu.[15]

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi=
"http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http:
  //maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/
  maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.2.0.M1</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>cz.cvut.fit.kubecpat</groupId>
    <artifactId>fitness-app</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>fitness-app</name>
    <description>Demo project for Spring Boot</description>

    <properties>
        <java.version>11</java.version>
        <vaadin.version>13.0.1</vaadin.version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
```

Zdrojový kód 3: Ukázka konfigurace projektu v souboru pom.xml a propojených závislostí

4.2.8 AJAX

AJAX je označení pro souhrn technologií používaných pro vývoj webových aplikací. Konkrétně to jsou technologie JavaScript a XML. Pomocí asynchronního zpracování stránek se mění obsah, aniž by docházelo k znovunačtení celé stránky.[16]

4.2.9 JavaScript

JavaScript je událostmi řízený, objektově orientovaný skriptovací jazyk. Používá se pro tvorbu dynamického obsahu na webových stránkách a vkládá se jako součást HTML kódu. JavaScript se spouští na straně klienta až po stáhnutí celé stránky.[17]

4.2.10 JPA

JPA (Java Persistence API) je ORM (objektově-relační mapování) pro ukládání, přístup a správu java objektů v relační databázi. JPA není nástroj nebo framework, ale sada konceptů, které mohou být implementovány různými nástroji. Nejznámější jsou například Hibernate a EclipseLink. JPA specifikace nabízí vývojáři možnost, aby si sám definoval, jak a jaké objekty mají být uloženy v java aplikaci.[18] V přiloženém kódu lze vidět definici rozhraní rozšiřující JPA repozitář.

```
@Repository
public interface ProductRepository extends JpaRepository<Product, Long> {

    Page<Product> findBy(Pageable page);
    Page<Product> findByNameIgnoreCase(String name, Pageable page);
    int countByName(String name);
    Product findByCode(String code);

}
```

Zdrojový kód 4: Ukázka JPA repozitáře persistující objekt Produkt

4.2.11 Lombok

Lombok je knihovna generující zdrojový kód v podobě java anotací. Cílem knihovny je odstranit rutinní práci programátora. Metody jako *get() set()*, nebo *equalsAndHashCode()* se s použitím Lomboku nemusí ručně psát, ale stačí použít anotaci. Použití Lomboku je zobrazeno ve zdrojovém kódu níže.[19]

```
@EqualsAndHashCode(callSuper = true)
@Data
@NoArgsConstructor @AllArgsConstructor
@Entity
public class Product extends AbstractEntity {
    private String name;
    private Double quantity;
    private String category;
    private String code;
    private Double price;
}
```

Zdrojový kód 5: Ukázka použití Lombok knihovny

4.2.12 Hibernate

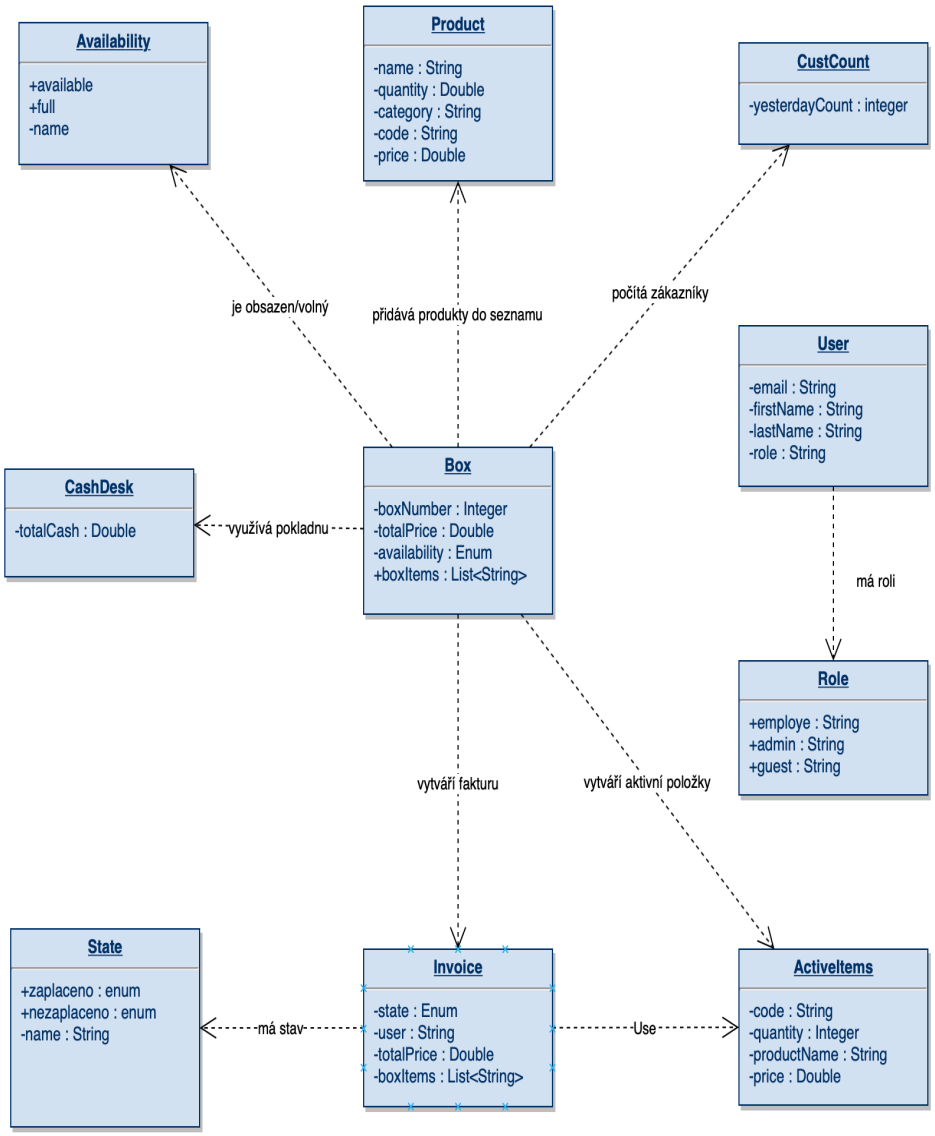
Hibernate je ORM knihovna (framework) pro javu vyvinutá jako alternativa k entitním *beanám* pro persistenci. Aktuální knihovna Hibernate implementuje nejnovější verzi JPA, navíc samotný Hibernate podporuje sadu nástrojů. Mezi nimi nechybí Hibernate Search, Hibernate Validator, nebo Hibernate OGM, které podporují doménový model persistence pro NoSQL.[18]

```
spring.main.lazy-initialization=true

spring.datasource.url=jdbc:h2:file:~/test2;DB_CLOSE_ON_EXIT=FALSE;
AUTO_RECONNECT=TRUE
spring.datasource.driverClassName=org.h2.Driver
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

spring.jpa.hibernate.ddl-auto = create-drop
spring.jpa.properties.hibernate.show_sql=false

spring.h2.console.enabled=true
spring.h2.console.path=/h2-console
```

Zdrojový kód 6: Ukázka konfigurace hibernate s JPA

4.3 Class diagram

Na obrázku níže je vidět class diagram webové aplikace, který znázorňuje datové struktury a souvislosti mezi jednotlivými objekty.



Obrázek 4.2: Class diagram aplikace

4.4 View

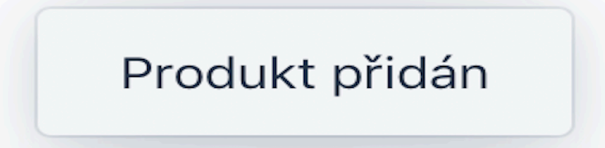
4.4.1 Notifikace

O notifikace v projektu se stará nakonfigurovaná servisní *beana* zobrazující přijatý text v horní části stránky jako notifikační label.

```
@Service
public class MessageBean {
    public void message(String notif) {
        Notification.show(notif, 2000, Notification.Position.TOP_CENTER);
    }
}
```

Zdrojový kód 7: Ukázka servisní message beany

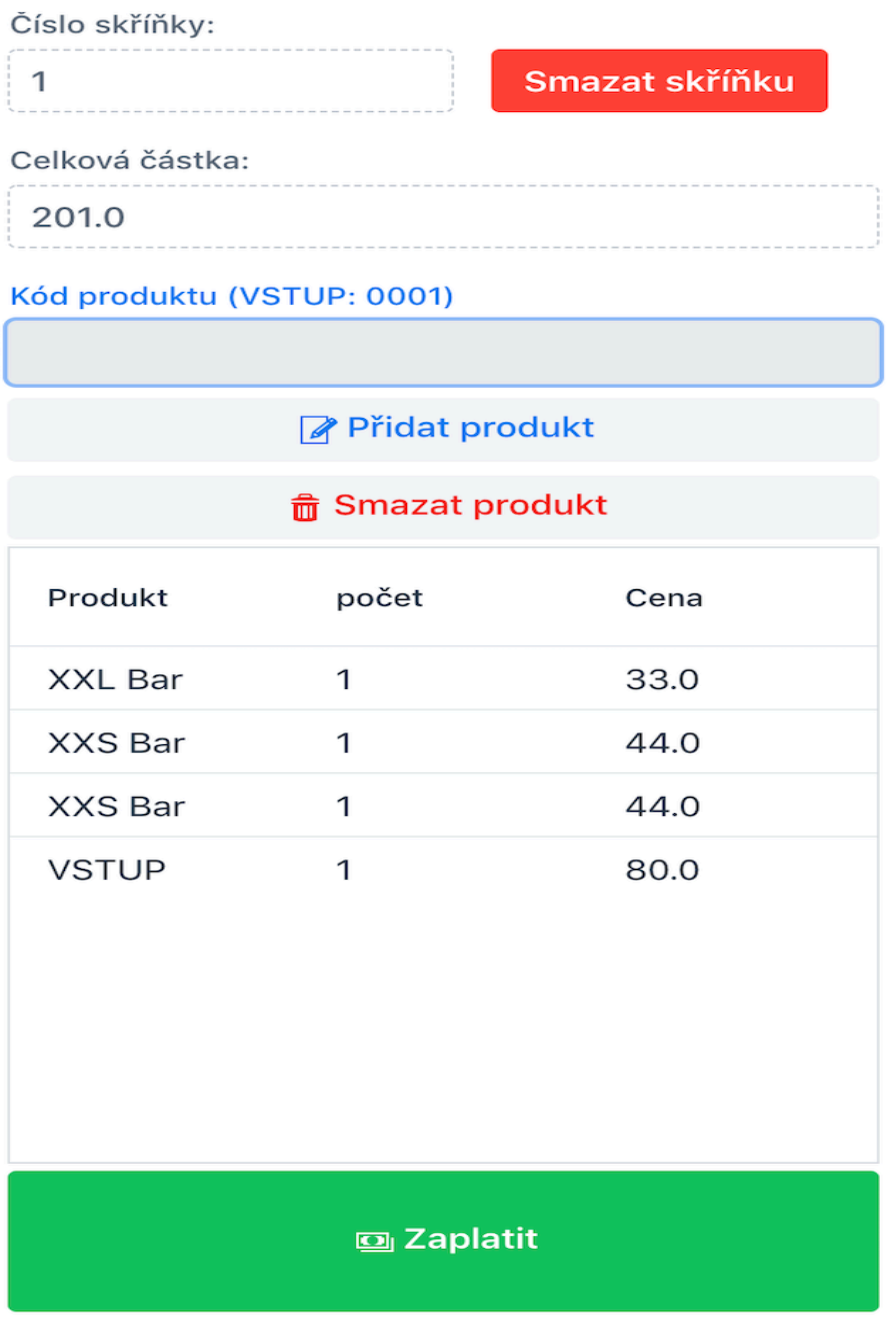
Snaha o čistý vzhled aplikace je jedna z priorit, kladených na tuto práci, čemuž napomáhá Vaadin komponenta „Notification“.



Obrázek 4.3: Notifikace o přidání produktu

4.4.2 Detail vybrané skříňky

Uživatelské rozhraní pro editaci jednotlivých skříněk je zobrazeno na obrázku níže. Umožňuje do skříňky přidat produkt pomocí unikátního kódu a tlačítka „Přidat produkt“. Výběrem produktu ve skříňce a následným stiskem tlačítka „Smazat produkt“ se ze skříňky produkt odebere. Tlačítkem „Zaplatit“ se skříňka uvolní a celková suma se přičte do pokladny, zároveň se vytvoří objekt „Invoice“, který je dále evidován pomocí JPA.



Obrázek 4.4: Detail formuláře pro skříňku

4.4.3 Validace

Validace některých prvků je provedena pomocí Vaadin komponent a binderu. Při vytváření vazeb (binderu) je možnost definovat convertor i s validací dat. Validace se dá provést i pomocí tzv. Spring anotací nabízející Spring framework. V ukázce kódu pod textem jsou zachyceny oba způsoby validace.

```
public void setBinder(BeanValidationBinder<Product> binder) {
        binder.forField(name)
                .asRequired("Nemůže být prázdný")
                .bind("name");
        binder.forField(quantity)
                .withConverter(Double::valueOf, value ->
                (value == null) ? "" : String.valueOf(value),
                "Musí být číslo")
                .asRequired("1-100")
                .bind("quantity");
        binder.forField(price)
                .withConverter(Double::valueOf, value ->
                (value == null) ? "" : String.valueOf(value),
                "Musí být číslo")
                .bind("price");
        binder.bind(category, "category");
        binder.bind(kod, "code");
        binder.bindInstanceFields(this);
    }
```

Zdrojový kód 8: Ukázka validace pomocí Vaadin binderu

```
@Entity
public class User extends AbstractEntity {

    @NotEmpty
    @Email
    @Size(max = 255)
    @Column(unique = true)
    private String email;

    @NotBlank
    @Size(max = 255)
    private String firstName;
```

Zdrojový kód 9: Ukázka validace pomocí Spring anotací

Kapitola 5

Testování

Java Spring obsahuje mimo jiné i knihovnu JUnit 4 umožňující implementaci a spuštění integračních a unit testů. Knihovna obsahuje také funkce k porovnávání objektů a hodnot.

5.0.1 Implementace JUnit 4 testů

Test v JUnit 4 vypadá jako běžná java třída s anotací *@Test*. Od verze 4 lze celé testování řídit pouze anotacemi, testovací třída tedy nemusí obsahovat žádný kód, až na samotný test.

5.0.2 View testy

Pro testování samostatných view komponent je zapotřebí tzv. mockování. Mock je „falešná“ třída, které definujeme chování sami v konkrétním testu. Pro vytvoření mock třídy je třeba použít anotace *@MockBean*.

```
@MockBean
private TopBar topBar;
@MockBean
private MasterDetail masterDetail;
@MockBean
private BasicFormLayout basicFormLayout;
```

Zdrojový kód 10: Ukázka vytvoření mock

```
@RunWith(SpringRunner.class)
@DataJpaTest
public class FitnessAppApplicationTests {}
```

Zdrojový kód 11: Ukázka konfigurace testů

5.0.3 Databázové testy

Beany potřebné pro databázovou komunikaci jsou dostupné pouze tehdy, když je spuštěná aplikace a navázáno spojení s databází. Framework Spring se o to stará pomocí anotací zobrazených v kódu níže. Databázové testy testují především CRUD operace nad všemi repozitáři v aplikaci.

```
@RunWith(SpringRunner.class)
@DataJpaTest
public class FitnessAppApplicationTests {

    @Autowired
    private ProductRepository productRepository;

    @Test
    public void testCreateProduct() {
        Product product = new Product( "Heat", 20D, "Energy",
        "2222",59D);
        productRepository.saveAndFlush(product);
        Product myProduct       productRepository.findByCode
        (product.getCode());
        assertEquals(myProduct.getId(), product.getId());
    }
    @Test
    public void deleteProduct() {
        Product product = new Product( "Heat", 20D, "Energy",
        "2222", 59D);
        productRepository.saveAndFlush(product);
        productRepository.deleteById(product.getId());
        Assert.assertFalse("delete product error",
        productRepository.existsById(product.getId()));
    }
    @Test
    public void testEditProduct(){
        Product product = new Product( "Heat", 20D, "Energy",
        "2222", 59D);
        productRepository.saveAndFlush(product);
        product.setName("name");
        productRepository.saveAndFlush(product);
        assertEquals("Edit product error", product.getName(),
        "name");
    }
```

Zdrojový kód 12: Ukázka testování CRUD

Závěr

Cílem práce bylo navrhnout a implementovat prototyp webové aplikace, která eviduje skladové položky, spravuje zaměstnance a je přizpůsobená specifickým potřebám konkrétního fitness studia (Fitness Power, Praha 4, Pujmanové 1220/6, Pankrác 140 00). Aplikace měla být přehledná a jednoduchá na ovládání, aby urychlila přijímání nových zákazníků, a zaměstnancům tak usnadnila fungování na pracovišti. Mezi funkce aplikace patří zaznamenávání příchodu a odchodu jednotlivých zákazníků a vedení statistiky o jejich aktuálním počtu. Statistika ve formě přehledného grafu je viditelná ve všech požadovaných částech aplikace. Přehled zákazníků je realizován prostřednictvím skříněk. Každému zákazníkovi je přiděleno číslo skříňky, pod kterým je v systému veden. Dalším funkčním požadavkem bylo přiřazování zakoupeného produktu ke skřínce daného zákazníka. S tím souvisí požadavek evidovat (přidávat, mazat a upravovat) produkty na firemním skladu. Každý produkt má svůj unikátní čárový kód. Po zaplacení produktu se informace o platbě (seznam zakoupených produktů, celková cena a datum zaplacení) persistentně uloží do aplikace. Výsledná aplikace splňuje všechny stanovené požadavky, které byly kladeny v analýze. Funkční prototyp je poutavý a má nadčasový vzhled, který je docíleno využitím webového frameworku Vaadin.

Webovou aplikaci lze dále doplnit o rozpis zaměstnanců, automatické posílání emailů, o Spring security nebo o měření a vyhodnocování statistik návštěvnosti.

Bibliografie

1. GOOGLE. *Úvod do Angular* [[online]]. Dostupné také z: https://www.itnetwork.cz/javascript/angular/zaklady/uvod-do-angular-frameworku. [cit. 2019-05-10].
2. FACEBOOK. *Úvod do React* [[online]]. Dostupné také z: https://www.itnetwork.cz/javascript/react/uvod-do-react. [cit. 2019-05-10].
3. KUBOVÝ, Tomáš. *Vaadin - přechod na novou technologii existujícího portfolia produktů* [[online]]. Dostupné také z: https://vaadin.com/download/book-of-vaadin/vaadin-7/pdf/book-of-vaadin.pdf. [cit. 2019-05-09].
4. GRÖNROOS, MARKO. *VBook of Vaadin: Vaadin 7 Edition* [[online]]. Dostupné také z: https://otik.uk.zcu.cz/bitstream/11025/12548/1/DP_Kubovy.pdf. [cit. 2019-05-09].
5. GOOGLE. *Google Web Toolkit (GWT)* [[online]]. Dostupné také z: https://opensource.google.com/projects/gwt. [cit. 2019-05-09].
6. VAADIN. *Components and tools for building web apps in Java* [[online]]. Dostupné také z: https://vaadin.com/docs/v10/flow/Overview.html. [cit. 2019-04-020].
7. GOOGLE. *AngularJS* [[online]]. Dostupné také z: https://angular.io/docs. [cit. 2019-04-020].
8. MICROSOFT. *Microsoft Excel* [[online]]. Dostupné také z: https://products.office.com/cs-cz/excel. [cit. 2019-04-019].
9. FOUNDATION, The Document. *LibreOffice* [[online]]. Dostupné také z: https://cs.libreoffice.org. [cit. 2019-04-019].
10. LEBL, David [osobní rozhovor]. 2019. 03. 20.
11. GOOGLE. *Google Sheets* [[online]]. Dostupné také z: https://www.google.com/intl/cs_cz/sheets/about/. [cit. 2019-04-020].

12. RAMSDALE, Chris. *Building MVP apps: MVP Part I* [[online]]. 2010. Dostupné také z: `http://www.gwtproject.org/articles/mvp-architecture.html`. [cit. 2019-04-019].

13. SPRINGSOURCE. *Spring Boot* [[online]]. Dostupné také z: `https://spring.io/projects/spring-boot#overview`. [cit. 2019-05-10].

14. GIT. *Git docs* [[online]]. Dostupné také z: `https://git-scm.com/book/en/v1/Getting-Started-Git-Basics`. [cit. 2019-05-10].

15. APACHE. *Maven Archetype* [[online]]. Dostupné také z: `https://maven.apache.org/archetype/index.html`. [cit. 2019-05-10].

16. MDNWEBDOCS-BOT. *Getting Started* [[online]]. Dostupné také z: `https://developer.mozilla.org/en-US/docs/Web/Guide/AJAX/Getting_Started`. [cit. 2019-05-10].

17. JANOVSKÝ, Dušan. *Úvod do JavaScriptu* [[online]]. Dostupné také z: `https://www.jakpsatweb.cz/javascript/javascript-uvod.html`. [cit. 2019-05-10].

18. TYSON, Matthew. *What is JPA? Introduction to the Java Persistence API* [[online]]. Dostupné také z: `https://www.javaworld.com/article/3379043/what-is-jpa-introduction-to-the-java-persistence-api.html`. [cit. 2019-05-10].

19. MOROSYSTEMS. *LOMBOK – KNIHOVNA PRO URYCHLENÍ VÝVOJE* [[online]]. Dostupné také z: `https://vsadnajavu.cz/2009-11/odborne/java/lombok-knihovna-pro-urychleni-vyvoje/`. [cit. 2019-05-10].

Seznam výpisu kódu

Příloha A

Seznam použitých zkratek

AJAX asynchronous JavaScript and XML

API application interface

CRUD create read update delete

CSS cascading style sheet

DI dependency injection

EJB enterprise java bean

GWT google web toolkit

HTML hypertext markup language

HTTP hypertext transfer protocol

HTTPS hypertext transfer protocol secure

IoC inversion of control

JAR java archive

JDBC java database connectivity

JPA java persistence api

JS JavaScript

JSON JavaScript object notation

MVC model view controller

MVCC multi-version concurrency control

MVP model view presenter

SPA single page application

SQL structured query language

ODBC open database connectivity

OGM object grid mapper

ORDBMS bject-relational database anagement ystem

ORM object-relational mapping

UI user interface

XML extensible markup language

WAR web application archive

Příloha B

Uživatelská příručka

B.1 Spuštění aplikace

Jelikož je aplikace založená na Spring frameworku, je pro spuštění aplikace zapotřebí jen velice málo kroků.

- nainstalovat JDK 11
- nainstalovat nejnovější verzi Maven
- vložit do PATH složku s Maven/bin

Po splnění těchto kroků stačí otevřít terminál a ve složce s aplikací spustit příkaz *mvn spring-boot:run*. Alternativně lze spustit aplikaci pomocí IDE jako IntelliJ IDEA. Po spuštění je aplikace dostupná v prohlížečí na adrese http://localhost:8080/.

Příloha C

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `application` ..... adresář s aplikací
  - `src` ..... zdrojové kódy aplikace
  - `uploads` ..... adresář s náhledy aplikace
  - `bower_components` ..... adresář s bower komponenty aplikace
  - `pom.xml` ..... konfigurační soubor mavenu
  - `README.md` ..... popis aplikace v git adresáři
  - `mvnw` ..... maven wrapper
  - `mvnw.cmd` ..... maven wrapper
- `text` ..... adresář s textem práce
  - `BP_Kubec_Patrik_2019.pdf` ..... text práce ve formátu PDF
  - `src` ..... zrdrojové kódy práce ve formátu LaTeX