UNIVERZITA PARDUBICE

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

BAKALÁŘSKÁ PRÁCE

2017 Oleksandra Kosenko

Univerzita Pardubice

Fakulta elektrotechniky a informatiky

Desktopová Java aplikace na webu

Oleksandra Kosenko

Bakalářská práce

2017

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2016/2017**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Oleksandra Kosenko**
Osobní číslo:	**I13155**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Desktopová Java aplikace na webu**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování:

Cílem práce je vytvořit zvolenou desktopovou aplikaci v jazyce Java a vystavit ji na webové stránce.

V teoretické části budou v historické posloupnosti popsány a zhodnoceny možnosti vystavení desktopové aplikace napsané v jazyce Java SE na webovské stránce.

V praktické části bude vytvořena GUI desktopová aplikace v Javě, která bude realizovat jednoduchý hotelový účetní systém, a bude vybraným způsobem implementována do webové stránky a v ní spuštěna.

Rozsah grafických prací:

Rozsah pracovní zprávy: **cca 35 stran**

Forma zpracování bakalářské práce: **tištěná**

Seznam odborné literatury:

SCHILDT, Herbert. Java 8: výukový kurs. Přeložil Jakub GONER. Brno: Computer Press, 2016. ISBN 978-80-251-4665-1.

KALIN, Martin. Java web services: up and running. Sebastopol, Calif.: O'Reilly, c2009. ISBN 059652112X.

PARSONS, David. Dynamic web application development using XML and Java. London: Cengage Learning EMES, 2008. ISBN 9781844805419.

Vedoucí bakalářské práce: **Ing. Zdeněk Šilar, Ph.D.**
Katedra informačních technologií

Datum zadání bakalářské práce: **31. října 2016**

Termín odevzdání bakalářské práce: **12. května 2017**

Univerzita Pardubice Česká republika

L.S.

Ing. Zdeněk Němec, Ph.D.
děkan

Ing. Zdeněk Šilar, Ph.D.
pověřený vedením katedry

V Pardubicích dne 31. března 2017

Prohlášení autora

Prohlašuji, že jsem tuto práci 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.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 29.04.2017

Oleksandra Kosenko

PODĚKOVÁNÍ

Ráda bych poděkovala vedoucímu své bakalářské práce Ing. Zdeňku Šilaru Ph.D. za odborné vedení, jeho pomoc, podporu a praktické rady při zpracování této práce.

Chci poděkovat Univerzitě Pardubice za to, že jsem byla přijata jako studentka české univerzity. Univerzita Pardubice změnila od základu mou představu o kvalitě studia a ukázala mi, jak musí probíhat správné vzdělávání, správné vztahy mezi studentem a učitelem, a navždy mi změnila život.

Děkuji také své rodině a přátelům za jejích podporu, které si vysoce vážím.

ANOTACE

Bakalářská práce se zabývá analýzou a problematikou existujících způsobů spuštění desktopové aplikace v jazyce Java na webové stránce. Teoretická část práce obsahuje popis Java aplikace, frameworku a technologií určené pro spuštění GUI aplikace na webu. Součásti této práce je také vývoj aplikace ve frameworku Vaadin jako demonstrace praktické ukázky jednoho ze způsobů řešení postaveného problému.

KLÍČOVÁ SLOVA

Java, GUI, desktopová aplikace, Java Applet, framework, Vaadin.

TITLE

Java desktop application on web.

ANNOTATION

This bachelor thesis deals with analysis and problems of existing ways of the desktop application launch process In the Java language on a web page. A theoretical part of this thesis contains description of the Java application, frameworks and technology designed to run a GUI application on the web. A part of this work consists of development of the application in the Vaadin framework as a demonstration of one of the ways of solving the problem.

KEYWORDS

Java, GUI, desktop application, Java Applet, framework, Vaadin.

OBSAH

SEZNAM ILUSTRACÍ A TABULEK

SEZNAM ZKRATEK A ZNAČEK

ACID	Atomicity, Consistency, Isolation, Durability
API	Application Programming Interface
AWT	Abstract Window Toolkit
CSS	Cascading Style Sheets
DNS	Domain Name System
GUI	Graphical User Interface
GWT	Google Web Toolkit
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IDE	Integrated Development Environment
IE	Internet Explorer
Java EE	Enterprise Edition
JFC	Java Foundations Classes
Java SE	Java Standard Edition
JDBC	Java Database Connectivity
JDK	Java Development Kit
JEE	Java Enterprise Edition
JNLP	Java Network Launcher Protocol
JRE	Java Runtime Environment
JSF	JavaServer Faces
JSON	JavaScript Object Notation
JSP	Java Server Pages
JVM	Java Virtual Machine

JWS	Java Web Start
LBS	Legacy, Browser Support
MIME	Multimedia Inetrnet Mail Extension
MSN	Microsoft Network
MVC	Model View Controller
NaCI	Native Client
NPAPI	Netscape Plugin Application Programming Interface
OCI	Oracle Call Interface
PC	Personal Computer
PHP	Hypertext Preprocessor
PL/SQL	Procedural Language/Structured Query Language
POJO	Plain Old Java Objects
POM	Project object model
PPAPI	Pepper API
RIA	Rich Internet Applications
SDK	Software development kit
SMTP	Simple Mail Transfer Protocol
SPDY	pronounced "SPeeDY"
SQL	Structured Query Language
UI	User Interface
WISYWIG	What you see is what you get
WWW	World Wide Web
XHTML	Extensible HyperText Markup Language
XML	eXtensible Markup Language

ÚVOD

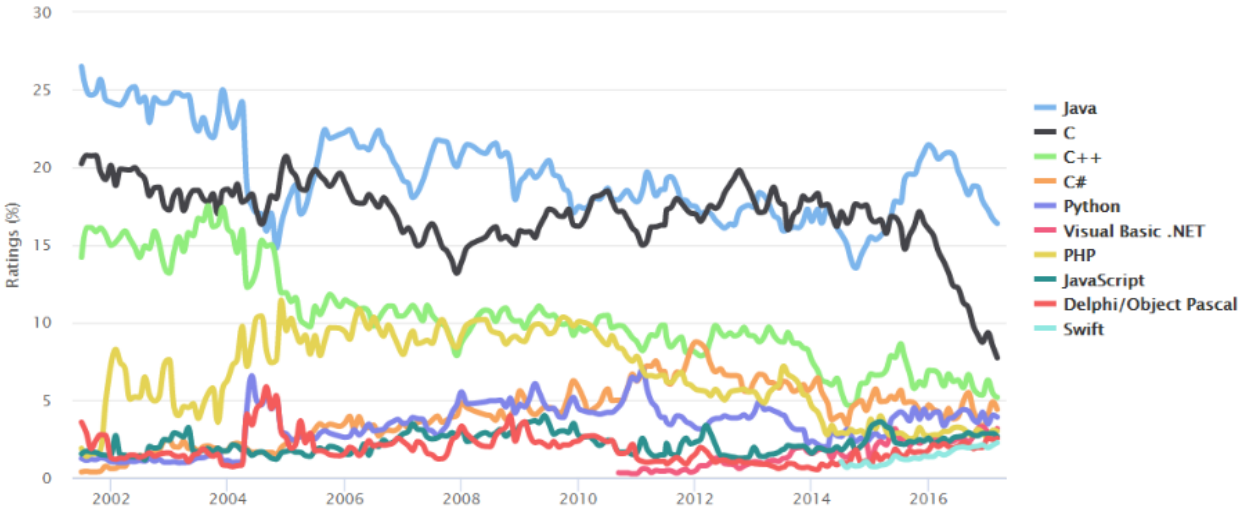
Internet od svého vzniku prošel značným vývojem. S každým dnem roste počet uživatelů internetu a, jako následek, i počet webových stránek a aplikaci. Potenciálního uživatele může upoutat užitečná funkcionalita, příjemný vzhled, dostupnost a uživatelská jednoduchost. Samozřejmě se vždy vzniká otázka, jak daný projekt zrealizovat a kterou technologií a programovací jazyk použít. Stavebními kameny tvorby webové aplikace jsou značkovací jazyk HTML/XHTML (Extensible HyperText Markup Language) a CSS (Cascading Style Sheets), které potřebujeme pro vizualizaci stránek. Mezi jiné velice používané technologie můžeme uvést JavaScript pro realizace komunikace ze strany klienta a PHP ze strany serveru.

Vývoj webových aplikací je ale o trošku složitější než například vývoj stejné desktopové aplikace. Hodně vývojářů se setkává s problémy při tvorbě takových aplikací. Takovými problémy se dost často stávají malé zkušenosti, nedostatečné znalosti v této oblasti, existence velkého množství jiných než při tvorbě desktopové aplikace pravidel.

Existuje však řada technologií, které umožnují tvorbu www aplikací používáním programovacích jazyků, tak dobře známé vývojářům desktopových aplikací, jako např. Java. Podle hodnoceni TIOBE[1] Java je nejpopulárnějším programovacím jazykem současnosti (viz obrázek 1), a z toho důvodu je i neustále udržována a vylepšována, aby si udržela svou pozici na trhu informačních technologií.

Mezi nejznámější technologie tvorby desktopové aplikace na platformě Java patří Java Applet a Java Web Start. Tyto technologie budou podrobněji popsané v kapitole 2**.** Samozřejmě, že i frameworků, postavených na Javě, existuje velké množství. Podle statistiky nejpopulárnější jsou Spring MVC, JSF, Vaadin, dále také můžeme k tomu seznamu přidat méně používané Google Web Toolkit, Gralis, Play 2 atd. Některé z nich úplně nebo částečně řeší problém spuštění desktopové aplikace na webu [1].

[1] TIOBE index je hodnoceni založené na analýze počtu výskytu dotazů do různých internetových vyhledávačů (Google, Google Blogs, MSN, Yahoo!, Baidu, Wikipedia, YouTube). Výsledky jsou zpracovávané jednou měsíčně.



Obrázek 1 - TIOBE Programming Community Index [2]

Hlavním cílem bakalářské práce je analýza existujících způsobu spuštění desktopové aplikace v Javě na webové stránce a následně vytvořeni a spuštění aplikace ve webovém prohlížeče. Čtenář by měl získat informace o nejznámějších technologií a o několika nejpopulárnějších webových frameworcích v jazyce Java, pomoci kterých, lze celkově nebo částečně realizovat postavený cil. Důkladně bude popsán framework Vaadin který byl vybrán pro vývoj aplikace, a v poslední kapitole bude uvedená praktická ukázka jeho použití.

[2] Zdroj: https://www.tiobe.com/tiobe-index/

1 TYPY JAVA APLIKACI

V této kapitole jsou krátce popsané jednotlivé druhy Java aplikací, jejích výhody a nevýhody. Podrobněji bude rozebrán framework Vaadin, princip práce, jeho architektura, silné a slabé stránky atd.

1.1 Konzolové aplikace

Základem v programování je konzolová aplikace. Je to v podstatě počítačový program, který nevyužívá grafické rozhraní, ale pracuje pouze v testovém rozhraní a k ovládání se používá jenom počítačová klávesnice. Standardním výstupem konzolové aplikace je java.lang. system.out, vstupem – java.lang.system.in. Hlavní třída aplikace musí mít statickou metodu main(), které po spuštění aplikace bude předáno řízení aplikací. Konzolové aplikace jsou nejvhodnější pro automatizované úlohy, které nevyžadují grafické prostředí a interakci s uživatelem. Takovou aplikaci lze spustit pomoci vývojového prostředí nebo příkazové řádky.

```
11  public class JavaConsoleApp {
12
13      /**
14       * @param args the command line arguments
15       */
16      public static void main(String[] args) {
17
18          Scanner in = new Scanner(System.in);
19          System.out.print("Jmeno: ");
20          String name = in.nextLine();
21          System.out.print("Prijmeni: ");
22          String surname = in.nextLine();
23          System.out.print("Rok narozeni: ");
24          int year = in.nextInt();
25          int age = round(2017 - year);
26
27          System.out.println("Informace: \nJmeno: " + name + " Prijmeni: " + surname + " Vek: " + age);
28      }
29  }
30
```

```
Output - JavaConsoleApp (run)
run:
Jmeno: Oleksandra
Prijmeni: Kosenko
Rok narozeni: 1992
Informace:
Jmeno: Oleksandra Prijmeni: Kosenko Vek: 25
BUILD SUCCESSFUL (total time: 20 seconds)
```

Obrázek 2 - Konzolová Java aplikace

1.2 GUI desktopové aplikace

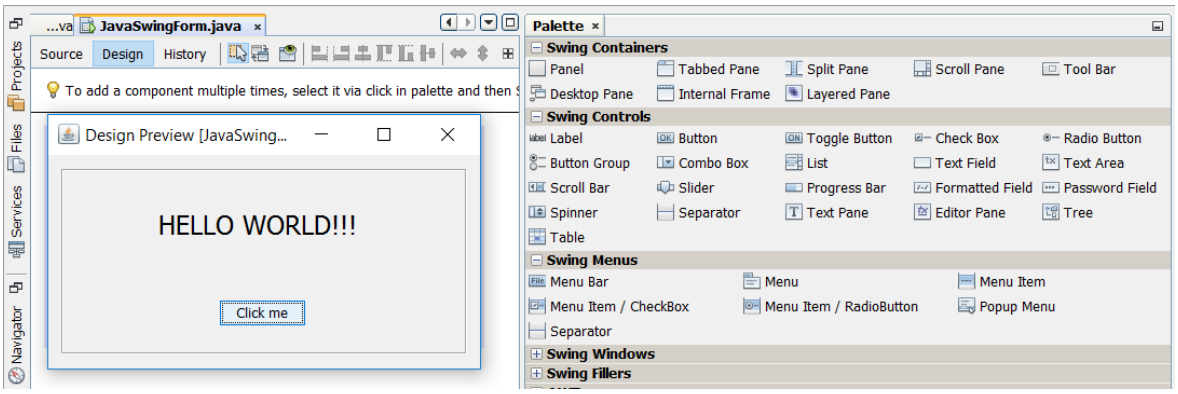
Desktopová aplikace je program s grafickým uživatelským rozhraním, který je instalován na počítač, startuje z pevného disku a využívá operační paměť a procesor pro běh.

Hlavní výhodou takové aplikace je to, že program se instaluje na PC, to znamená že je vždy k dispozici bez ohledu na připojení k síti. Mezi nevýhody patři potřeba neustálé bezpečnostní aktualizace, kontroly na viry, stahování a instalace nové verze, potřebu samostatného ukládání a zálohováni dat, práce se souborovým systémem a nízká bezpečnost.

1.2.1 Swing

Swing je knihovna, která může být známa i pod názvem Java Foundations Classes (JFC), byla představena po starší knihovně AWT, ale je novější a obsahuje větší kolekci grafických tříd než AWT. Knihovny se navzájem doplňují, takže mnoho tříd Swingu jsou podtřídami tříd z AWT. Swing eliminuje slabiny Abstract Window Toolkit knihovny, ale je více náročná na paměť. Má komponenty napsané přímo v Javě, což znamená, že vykreslování komponent je nezávislé na operačním systému, a tak na všech platformách Swing aplikace vypadají stejně. Samozřejmě lze to změnit pomocí tzv. LookAndFeel – metod třídy javax.swing.UIManager.

Existují dva způsoby, jakými lze vytvářet Swing formuláře. Buď to pomocí grafického návrháře nebo psaním kódu formuláře. V druhém případě tvorba větších aplikací je velmi náročná kvůli počtu komponent, jejích pozicování atd., takže se doporučuje využívat grafický návrhář [2].



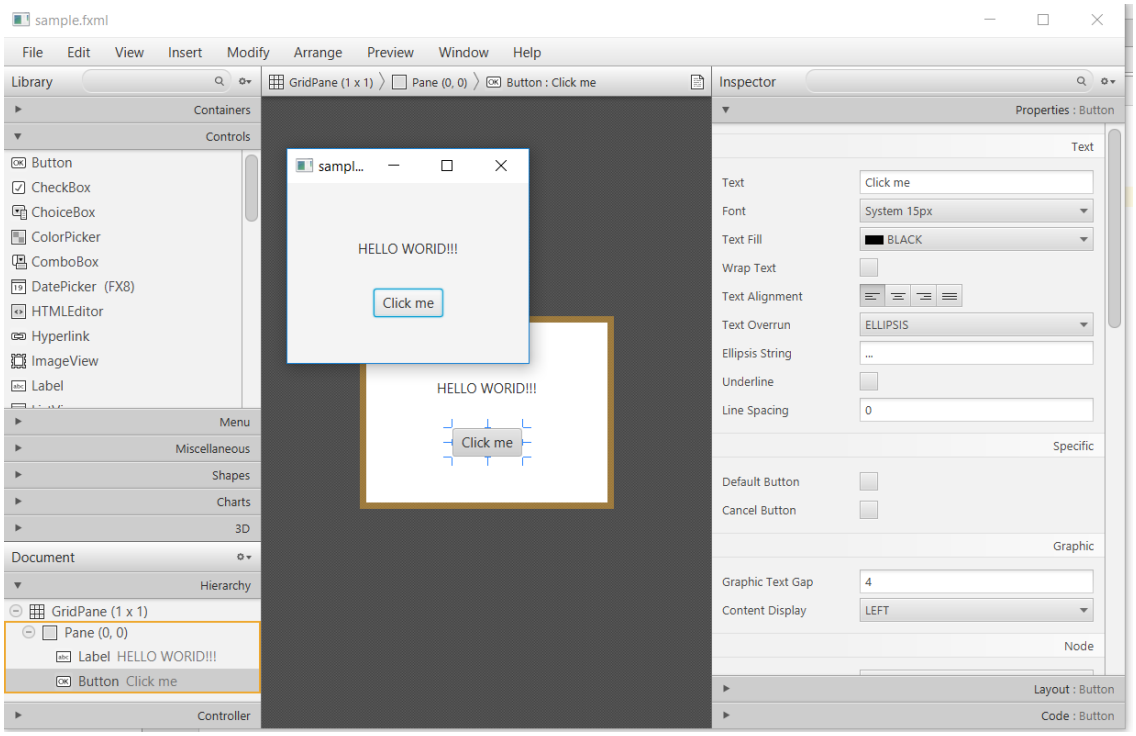
Obrázek 3 – Swing aplikace

1.2.2 JavaFX

JavaFX představuje moderní náhradu za Swing, od Java 8 má je součástí JRE/JDK. Obsahuje lepší oddělení logiky a výstupu než Swing, přináší podporu obrázků, videa, zvuku, grafů, CSS stylů, práce s dotykovým zařízením a mnoha dalších technologií. Zároveň je kladen důraz na jednoduchost tvorby. JavaFX se hodí jak pro desktopové aplikace, tak i pro webové a pro mobilní aplikace.

JavaFX je na platformě závislý (stejně jako JRE). Je k dispozici pro Windows, Linux, MacOs a Solaris. Novější aplikace většinou už používají JavaFX, v praxi se ale ještě setkáváme s mnoha aplikacemi, které jsou napsané ve Swingu.

Velkou výhodou je to, že máme k dispozici grafický designer – JavaFX Scene builder, takže FXML nemusíme psát ručně. Je to samostatná aplikace, jako plugin do Netbeans, Eclipse a Intellij Idea, je to v podstatě WISYWIG návrhář aplikace. Pomocí Scene builderu můžeme snadno navrhovat rozložení prvků, layoutu, nastavit obsluhy atd. Neumí ale generovat zdrojový text Javy. Podporuje CSS, lokalizační soubory, tvorbu kontrolérů atd. Aplikaci naprogramovanou v JavaFX také lze spustit na webu, ve skutečností je to stejné, jako spuštění Java Appletu (viz kapitola 2.1) [3].



Obrázek 4 – JavaFX aplikace, Scene Builder

1.3 Webová aplikace

Webové aplikace si získávají v dnešní době stále větší popularitu díky novým technologiím, které dokážou tyto aplikace udělat velice interaktivními a urychlují reakce na činnost uživatele, který pak nemusí čekat.

Velkou výhodou www aplikací je to, že se nemusí instalovat a aktualizovat (aktualizace probíhá na serveru). Pro práci potřebujeme jenom webový prohlížeč a připojení na internet, což znamená že aplikaci nezáleží na operačním systému nebo hardwaru. Aplikace pak funguje

prakticky všude, dokonce i na mobilu nebo tabletu. Data jsou uchovávána a zálohována na serveru. A samozřejmě zaleží i na počtu uživatelů. Těch, kteří by používali tu samou aplikaci, kdyby ona byla na desktopu místo webu, by bylo mnohem méně.

Mezi nevýhody webových aplikací patří jejích náročnější vývoj, potřeba připojení na internet, pomalejší tok dat a práce z aplikace, což muže být způsobeno špatnou kvalitou připojení. Nevýhodou jsou také bezpečnostní rizika dat, například v případě nekvalitního poskytovatele.

1.3.1 Architektura klient-server

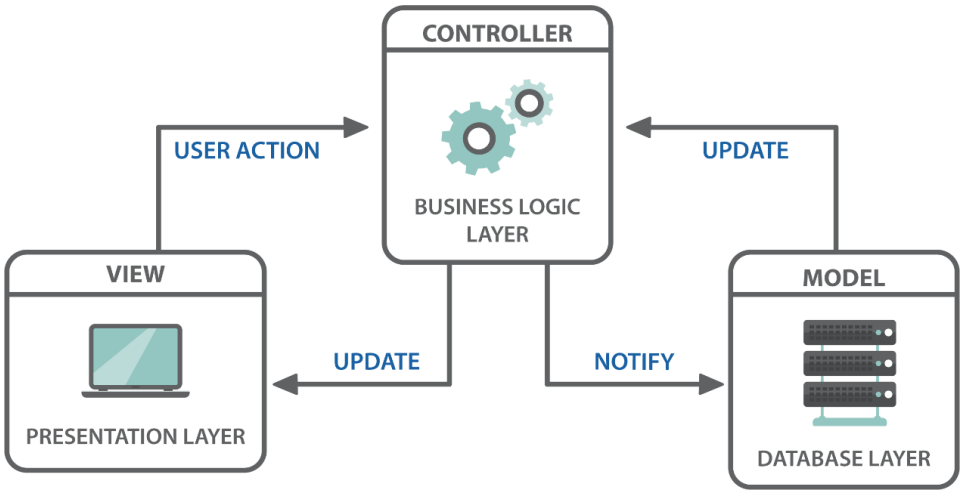
Webová aplikace je to v podstatě program, který je postaven na architektuře „klient-server“. Je to síťová architektura, která odděluje klienta a server, kteří spolu komunikují přes počítačovou síť. Model klient-server je jedním z hlavních myšlenek síťové technologie. Tento model používá většina obchodních nebo firemních aplikací, dále ho pak používají např. i tyto internetové protokoly HTTP, SMTP, Telnet, DNS, atd.

Každá instance klienta, v případě webové aplikace je to prohlížeč (IE, Opera, Google Chrome atd.), který může posílat žádost o data jednomu nebo více připojeným serverům. Na druhé straně – webové aplikace, které jsou umístěny na serveru, mohou akceptovat tyto žádosti, zpracovat je a vrátit klientovi požadovanou informaci [4].

Mezi výhody takové architektury patří rychlejší a jednodušší aktualizování údajů a taky větší bezpečnost. Servery lépe kontrolují přístup a zdroje, to zaručuje, že přistupovat, měnit a mazat data mohou pouze oprávnění klienti. Další výhodou je uložení dat na jednom místě – na serverech, které jsou mnohem bezpečnější než většina klientů. Lze nahradit, opravit, modernizovat nebo přemístit server, aniž by to klienti poznali, nebo tím byli nějak ovlivněni.

1.3.2 MVC architektura

Architektura MVC (Model View Controller) dělí aplikaci na 3 logické časti tak, aby je šlo upravovat samostatně a vliv na ostatní části byl co nejmenší. Vrstva „Model“ má na starosti práce s daty v databázi, v souborovém systému nebo webových formulářů. Tato vrstva se stará o správnou interpretaci uložených dat do aplikace, a naopak o manipulaci uložených dat příkazy aplikace. „View“ je prezentační vrstva, která se stará o správné zobrazení výsledků uživatelů. „Controller“ představuje sebou řídící vrstvu aplikace, která spravuje všechny požadavky od uživatele, provádí příslušné akce a na výstup předává pohled naplněný daty z modelu [4].



Obrázek 5 – MVC architektura

1.3.3 Java Server Pages

Velice známou technologií pro vývoj webové stránky s dynamickým obsahem jsou JSP (Java Server Pages). Tato technologie pro zobrazování obsahu umožnuje vývoj aplikací založených na webu při zachování na platformě a hardwaru hostitelského systému. JSP se jednoduše vytváří než Servlety, takže údržba dynamických stránek se stává snadnější a přehlednější. Pomocí JSP jde dynamický obsah stránky napsat v jazyce Java a statický – pomocí HTML. Hlavními výhodami jsou jednoduchost, existence velkého množství knihoven a snadnost jejich používání [5], [6].

1.4 Frameworky

V současné době s pojmem framework lze se setkat velmi často. Framework tvoří rámec (frame) pro řešení určité problémové oblasti, jako např. připojení a práce s databází, poskytování www služeb, tvorba uživatelského rozhraní apod. Pro řešení určitého problému framework poskytuje rozhrání, abstraktní i konkrétní třídy, které jsou podrobně popsány pomocí API. Podobným způsobem jsou definované i knihovny, které také obsahují třídy a rozhraní, ale poskytují pouze přesně definované operace. Framework na rozdíl od knihovny tvoří kostru celé budoucí aplikace a poskytuje rozhraní pro propojování jiných komponent, a tak programátor má dostatek prostoru pro implementaci vlastního rozšíření.

Webové frameworky jsou určené k podpoře vývoje dynamických webových aplikací či služeb. Je to v podstatě nadstavba nad základní vrstvou pro obsluhu HTTP protokolu. Takové frameworky obsahují např. nástroje pro generování webového obsahu a prostředky pro práci s databází. I když jsou většina z nich zaměřená na vytváření dynamických www stránek, je možné je využít i pro statické stránky.

1.4.1 JavaServer Faces

JavaServer Faces je komponentově orientovaný UI framework. Je součástí Java EE, takže je standardem této platformy a umožnuje vytvářet aplikace bez nutnosti přidávat do projektu další knihovny. JSF byl vytvořen pro jednoduchost návrhu a implementace uživatelského rozhraní webových aplikací. Tento framework je založen na MVC návrhovém vzoru, což znamená, že je zde kladen důraz na více vrstevnost aplikace, prezentační vrstva je oddělena od aplikační (business) vrstvy. JSF nabízí vestavěnou sadu UI komponent a HTML_BASIC Render Kit, který se stará o grafickou stránku aplikace.

Velkou výhodou JSF je efektivnější dělba práce, totiž vzniká několik rolí vývojářů aplikace, které jsou technologicky oddělitelné. Tak na projektu může pracovat několik specializovaných vývojářů, kteří se mají domluvit pouze na rozhraní. Každý z nich může být nasazen na vývoj své konkrétní oblasti, čímž můžeme zajistit kvalitní a rychlý vývoj aplikace. JavaServer Faces definuje 5 rolí, které se při vývoji aplikace uplatní, ale nejdůležitější jsou pouze tři z nich:

- Tvůrce stránek – zodpovědný za tvorbu uživatelského rozhraní a jeho chovaní.
- Vývojář aplikace – má na starosti funkcionalitu aplikace.
- Tvůrce komponent – tvoří znovupoužitelné komponenty, které se používají v JSP stránkách k vykreslování aplikačních objektů.

Další výhodou JSF je zautomatizování některých rutinních úkonů jako např. ukládání a načítání stavu komponent mezi požadavky, validace a konverze uživatelských vstupů, vkládání komponent uživatelského rozhraní na stránku pomocí dostupných knihoven tagů, což jsou knihovny, které zajišťují přistup k JSF třidám z JSP stránky atd.

JSF podporuje také událostní programování, což znamená, že je to v podstatě webová aplikace, která se programuje podobně jako desktopová [7].

1.4.2 Spring

Dalším frameworkem postaveném nad Java EE je Spring. Jde o open-source framework, který podporuje vývoj aplikace na všech vrstvách aplikačního modelu, tudíž, na rozdíl od většiny jiných frameworku, se nesoustřeďuje pouze na jedné vrstvě. Spring, jako open-source projekt, neustále vyvíjí a velice rychle reaguje na nové techniky. Díky otevřenému kódu lze do frameworku přímo zasahovat a upravovat jej podle vlastní potřeby a také se dá i šířit vlastní distribuce frameworku.

Je to modulární odlehčený kontejner, který umožnuje jak tvorbu webových, tak i desktopových aplikací. Cílem tohoto rámce je zefektivnění, snížení ceny a zjednodušení návrhu a vývoje JEE aplikací. A to v několika základních bodech:

- Odstranění těsných vazeb mezi jednotlivými POJO („Plain Old Java Objects“ – označování klasických Java tříd) objekty za pomocí návrhového vzoru Dependency Ingection,
- Správa konfigurace business komponent,
- Podpora pro přístup k datům, např. formou JDBC (Java Database Connectivity),
- Možnost volby implementace business vrstvy pro aplikační architekturu.

Zásadní filosofie tohoto aplikačního rámce spočívá ve vysoké modularitě. V případě potřeby využití pro určité účely vlastního řešení nebo alternativy třetích stran, není problém použít jen některé moduly. Navíc Spring velmi často nabízí také dobrou podporu i dalších frameworků, technologií atd. [8].

1.4.3 Vaadin

„Our mission is to make building amazing web applications easy.“[3]

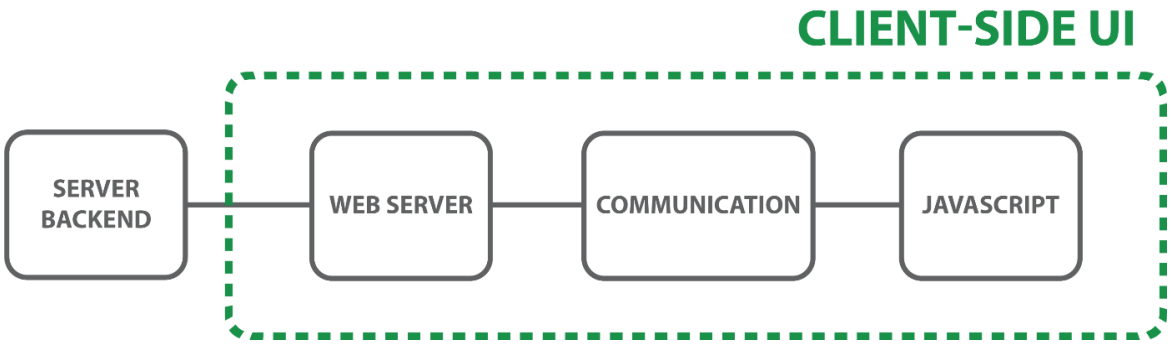
Obrázek 6 – Vaadin logo

Framework určený pro realizace rozsáhlých, korporátních webových aplikací. „Vaadin“ v překladu z finského jazyka znamená jelen (obrázek 6). Vaadin je jedním z moderních frameworků pro jednoduchou a efektivní tvorbu User Interface (UI) webových aplikací.

Mezi hlavní výhody patří výběr komponent[4], design – a stylu aplikace (nejnovější verze je Valo Theme[5]). Vaadin poskytuje tzv. motivy (Themes) – soubory ovlivňující vizuální vzhled aplikací. Motivy definují vzhled uživatelského rozhraní pomoci CSS a HTML. Vaadin také nabízí možnost jejich rozšíření či přepsání vlastními styly. Výhodou je také využiti GWT, což poskytuje tvorbu pokročilých komponent uživatelského rozhráni. Tyto vytvořené

[3] Zdroj: https://vaadin.com/company
[4] Přehled jednotlivých komponent a ukázky kódu: http://demo.vaadin.com/sampler/#
[5] Valo Theme: http://demo.vaadin.com/valo-theme/

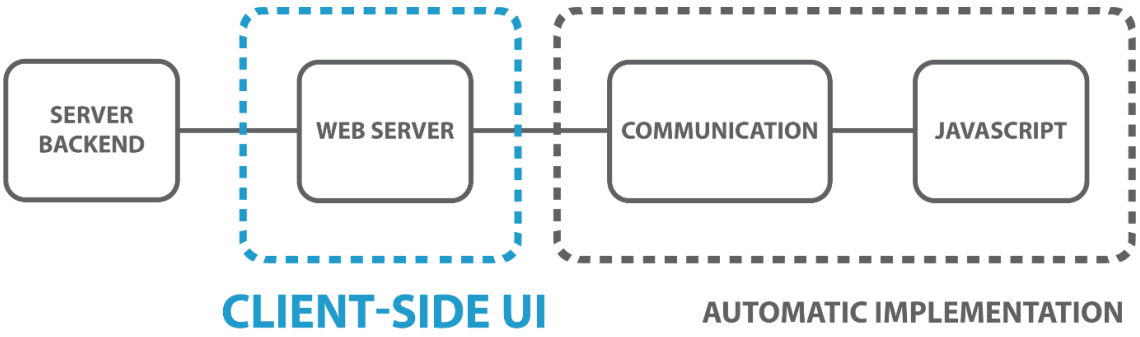
a přeložené komponenty umožňují rozšířit vlastní aplikaci o volně dostupné prvky třetích stran, tzv. add-ons („doplňky“)[6] .



Obrázek 7 – Struktura obecné webové aplikace

Další výhodou Vadinu je interakce klienta a serveru. Obyčejnou webovou aplikaci můžeme rozdělit na 4 časti (viz obrázek 7). Tři bloky – Web server, Komunikace a JavaScript zhruba můžeme sloučit a pojmenovat to uživatelským rozhraním na straně klienta.

Vývojáři Vaadinu se řídi jiným principem, hlavní myšlenkou, kterou je rozdělení jednotlivých bloků na dva odlišné modely, a to na stranu serverovou a klientskou, kde právě strana serverová nabízí sílu vývoje (viz obrázek 8). Všechny časti zodpovědné za komunikaci mezi serverem a klientem a klientské scripty se realizují automaticky. Programátor píše kód pro serverovou část a všechen zbylý kód se generuje na základě tohoto kódu. Díky tomu jde programovat uživatelské rozhraní jako pro klasické desktopové Java aplikace, ovšem podstatně jednodušeji, než nabízí například Swing.



Obrázek 8 – Struktura webové aplikace ve Vaadinu

Porovnejme realizace webové aplikace na libovolném JavaScript frameworku, na GWT a na Vaadinu. V prvním případě potřebujeme mít realizované následující části:

[6] Vaadin add-ons: https://vaadin.com/directory#!browse

- server backend,
- logika na web serveru, který přijímá požadavky a spolupracuje s backend-serverem,
- komunikace (např. JSON, XML, atd.)
- klientská část (odesílání požadavku na server, zpracování požadavku atd.)

Velkou nevýhodou je také použití dvou programovacích jazyků, Java pro stranu serveru a JavaScript – pro klienta.

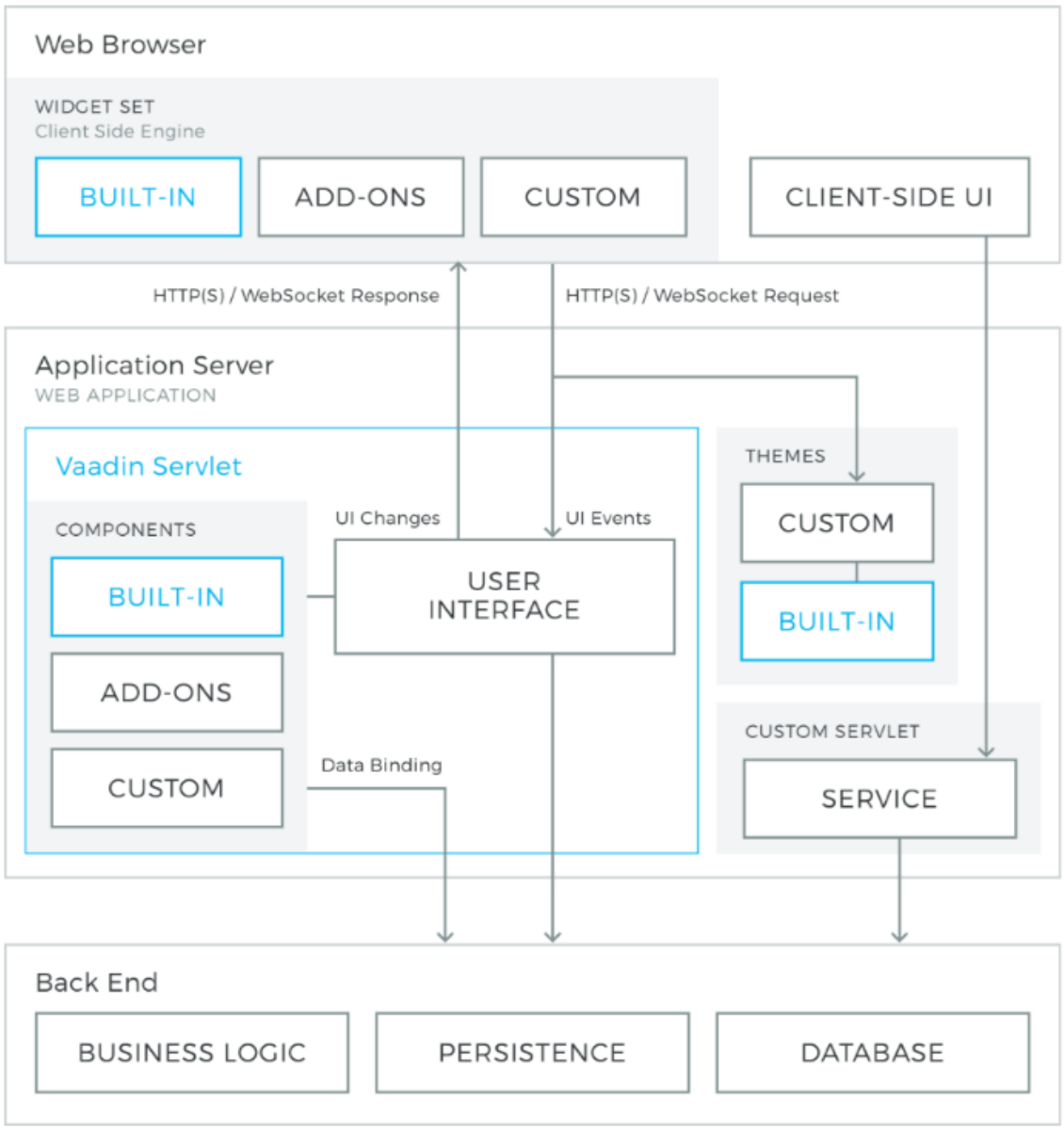
Framework GWT díky vnitřnímu kompilátoru Java-JavaScript usnadňuje tvorbu webové aplikace použitím jenom jednoho programovacího jazyka, totiž Java. Ale stále je potřeba realizace těch částí aplikace jak i v případě JS frameworku.

V případě práce z Vaadinem se realizuje jenom část strany serveru. Vaadin si bere na starosti klientský kód, a tak programátor bude pracovat mnohem rychleji a efektivněji. Vaadin používá pro vývoj technologie Java, takže je možnost použití jiných jazyků, které používají JVM. Tento framework využívá GWT (Google Web Toolkit) pro vykreslování uživatelského rozhraní v prohlížeči. Java zdrojový kód je následně překompilován do JavaScript. To výrazně usnadňuje vývoj a vede ke kompatibilitě ze všemi populárními webovými prohlížeči, jak na osobních počítačích, tak i na mobilních zařízeních a tabletech a na rozdíl od frameworků jako Flash nebo Java Applets, strana klienta spouští pouze JavaScript, takže není potřeba instalaci žádných speciálních pluginů.

Webová aplikace vytvořená pomocí Vaadinu se skládá ze serverové části, kde se nachází logická vrstva aplikace, logická část komponent a jednotlivé komponenty klientského „enginu“. Tento „engine“ běží v prohlížeči jako JavaScript. UI logika aplikace běží jako Java Servlet na Java aplikačním serveru (viz obrázek 9).

Vzhledem k tomu, že HTML, JavaScript a další technologie jsou v podstatě skryté před aplikační logikou, lze na prohlížeč hledět jako na tenkého klienta. Takový klient zobrazuje uživatelské rozhraní a komunikuje se serverovou částí po nižších vrstvách.

UI logika běží na aplikačním serveru v podobě servletů spolu s aplikační logikou. Naproti tomu normální architektura klient-server s klientskou aplikací může zahrnovat řadu aplikačně specifických komunikací mezi klientem a serverem. V podstatě odstranění vrstvy uživatelského rozhraní z aplikační architektury vede k efektivnějšímu přístupu dovolujícímu vytvářet tzv. RIA aplikace (Rich Internet Applications – „Bohaté internetové aplikace“).



Obrázek 9 – Detailní znázornění architektury aplikace vytvořené ve Vaadinu[7]

[7] Zdroj: https://vaadin.com/docs/-/part/framework/architecture/architecture-overview.html#architecture.overview

2 SPUŠTĚNÍ DESKTOPOVÉ APLIKACE VE WEBOVÉ STRANCE

Tato kapitola obsahuje informace o způsobech vystavení desktopové aplikace v jazyce Java na webu. Jsou tu stručně popsané určité technologie, jejich výhody a nevýhody, problematika a způsoby vyřešení. Mezi tyto technologie patří také framework Vaadin a JSF, které byly popsané v kapitolách 1.4.1, 1.4.3.

2.1 Java Applet

Java Applet je miniaplikace napsaná v programovacím jazyce Java, která je umístěná do těla HTML stránky, jako se vkládají obrázky, a pomocí virtuálního stroje JVM (Java Virtual Machine) je pouštěná v prohlížeči, který podporuje Javu. Svého času applety patřily k nejčastějším programům vytvářeným v Javě, sloužily většinou k oživení webových stránek, k upoutání pozornosti, ale i k vytváření seriózních rozsáhlých aplikací založených na WWW stránkách.

Doba appletu už pominula, ale i ve své době se applety moc netěšily velké oblibě. Uživatele odrazovala nutnost mít nainstalovaný JVM a správnou verzi Javy. Applety mají některá omezení jako čtení a zápis do souborového systému v počítači, kde je applet spouštěn, a nemožnost načítání knihoven a systémových proměnných. Tato omezeni mohou být na určitých JDK odlišná. Dále lze taková omezení nastavovat pomocí webového prohlížeče. V dnešní době k těmto omezením patří i nemožnost anebo velká obtížnost spouštění takovéto aplikace ve většině webových prohlížečů.

2.1.1 Struktura appletu

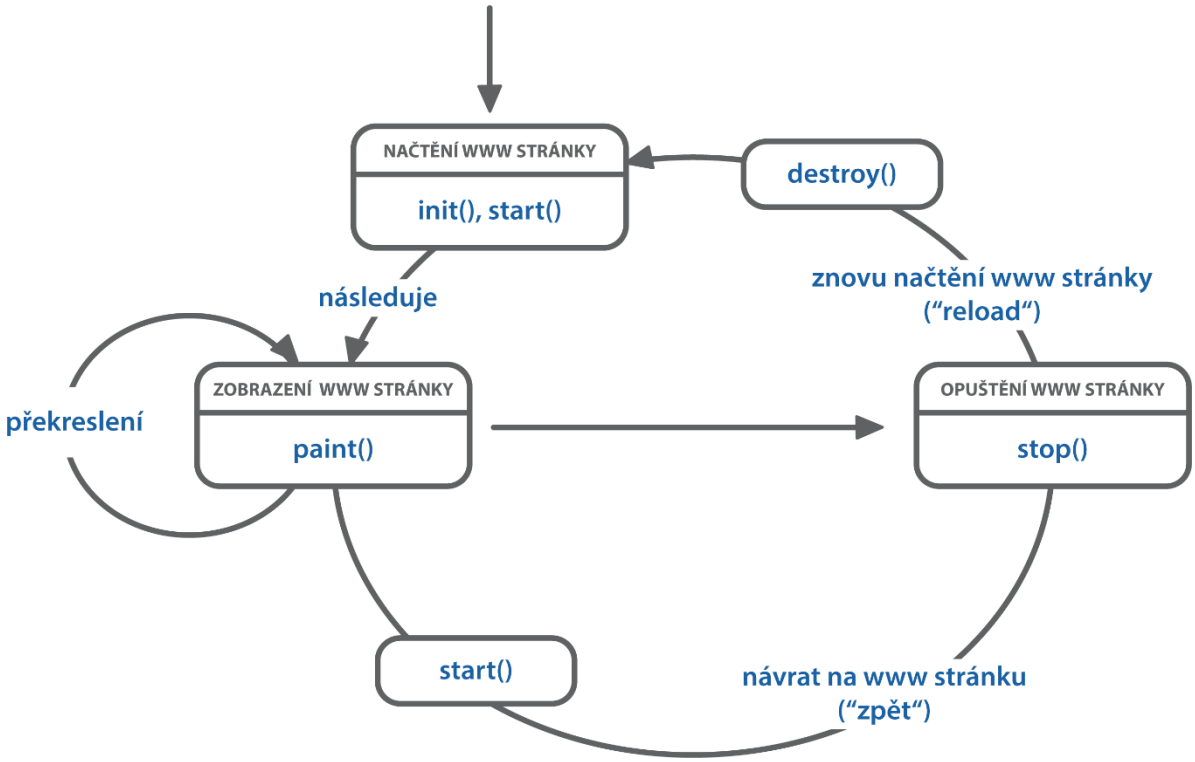
Applet je tvořen podobně jako obyčejná aplikace jednou nebo více třídami. Základem je třída odvozená od knihovny třídy javax.swing.JApplet. Tato třida umožnuje využívat grafické prvky knihovny Swing.Programování appletu je podobné jako vytváření jakékoliv jiné aplikace s grafickým uživatelským rozhraním, kromě toho, že:

- Na rozdíl od desktopové aplikace může, ale nemusí mít metodu main(). V případě, že tato metoda je použita v aplikaci, tak Java applet můžeme spustit jako normální desktopovou aplikaci. Třída java.applet.Applet definuje základní metody, které tvoří rozhraní mezi prohlížečem a appletem. Každý applet musí být potomkem této třídy. Pokud je načten do www stránky s příponou .html, prohlížeč řídí zpracování voláním

určitých metod. Pro činnost aplikace je zapotřebí v potomkovi překrýt alespoň tři z nich, a to jsou init(), start(), stop(). Mezi ostatní základné metody patří destroy(), paint() atd.

- Nesmí pracovat se soubory ani jinak zasahovat do systému na počítači, na kterém běží.

[3]



Obrázek 10 – Životní cyklus Java Appletu

Na základě ukázky vytvoření a spuštění appletu na webu (viz Příloha A) lze říct, že spuštění Java Appletu je obtížné a v některých webových prohlížečích i vůbec není možné. Větší počet nevýhod, než výhod dělá tuto techniku zastaralou a je vhodné použit novější technologie.

2.1.2 Problematika Java Appletu

Oracle oznámila že ukončuje vývoj a podporu pluginu Java SE v prohlížeči. Už od Oracle JDK 9, který od března tohoto roku je k dispozici, plugin je zapsán do kategorie zastaralých technologií a v dalších verzích se plánuje i odstranění z JDK a JRE. Namísto vkládání Java appletu do webových stránek, vývojáři doporučují používat technologie Java Web Start, která nevyžaduje existenci nainstalovaného pluginu v prohlížeči.

Ukončeni podpory tohoto pluginu je způsobeno zrušením podpory rozhraní NPAPI ve většině populárních prohlížečů. Například podpora NPAPI-pluginu v prohlížeči Chrome je ukončená už od verze Chrome 45 (nejnovější verze v době napsání bakalářské práce je Chrome 57).

Firefox z konce roku 2016 už nepodporuje žádný z NPAPI-pluginu, výjimkou je jenom Flash. Microsoft Edge už od začátku nepodporoval této pluginy.

Důvodem zrušeni podpory NPAPI rozhraní je zastaralá architektura její API, která byla vyvinuta před 15 lety. Dnešní prohlížeče jsou bezpečnější, rychlejší a mají mnohem rozšířenější funkcionalitu než jejích předchůdce. Používaní NPAPI vede ke zkomplikování kódu, problémům se zabezpečením a nízkou kvalitou. Navíc, NPAPI už původně nepodporoval práce s mobilními zařízeními. Jako náhradu NPAPI nabízejí práce z NaCI (Native Client), Apps, Native Messaging API a LBS (Legacy, Browser Support). Google zavadí PPAPI (Pepper API), který je nekompatibilní z NPAPI a zatím se nepodporuje v jiných prohlížečích [10], [11].

2.2 Java Web Start

Java web start je technologie, která umožnuje spouštět GUI aplikace v Javě ve vašem prohlížeči jenom kliknutím na odkaz umístěný na webové stránce. Místo toho, aby uživatel stahoval vybraný program z webového serveru, pak ho rozbaloval, instaloval a až poté spouštěl, poskytnou se JWS všechny potřebné informace v jednom souboru a JWS se o vše zmíněné postará sám. Základem je speciální XML soubor pro JNLP (Java Network Launcher Protocol), který se automaticky stáhne z URL adresy na váš počítač. Soubor obsahuje informace o tom, odkud aplikace stáhnout a jak ji spustit. Aplikace musí být vždy připravena do jednoho nebo více souboru JAR, který musí obsahovat také všechna data, která aplikace potřebuje pro svou činnost.

S Java Web Start se spoušti i když není k dispozici na vašem počítači. JWS si automaticky stáhne všechny potřebné soubory, uloží je v počítači, a tak aplikace bude vždy připravená na to, že bude znovu spuštěná, a to vždy v nejaktuálnější verzi, což je velkou výhodou. Nutnost stažení aplikace z internetu před jejím spuštěním může patřit k nevýhodám JWS.

Pro práci s JWS aplikací ji zaprvé musíme „nasadit“ na webový server, zajistit, aby webový server uměl pracovat s JNLP soubory. Server musí být nastaven tak, aby vracel správný MIME typ k souborům JNLP. Například, pro server Apach stačí přidat do souboru mime.types řádek application/x-java-jnlp-file. Jinak je taková konfigurace pro každý server různá. Pokud je vrácen tento MIME typ, webový prohlížeč už ten soubor stáhne a spustí, v opačném případě ho pouze zobrazí nebo stáhne.

Dalším krokem je vytvoření JNLP souboru. Je to XML soubor, který obsahuje elementy a atributy, které sdělují, jak aplikaci JWS spustit.

```
<?xml version="1.0" encoding="UTF-8"?>
<jnlp spec="1.0+" codebase=
"https://docs.oracle.com/javase/tutorialJWS/samples/deployment/webstart_ComponentArch_DynamicTreeDemo"
    href="dynamictree_webstart.jnlp">
    <information>
        <title>Dynamic Tree Demo</title>
        <vendor>Dynamic Team</vendor>
    </information>
    <resources>
        <!-- Application Resources -->
        <j2se version="1.7+"
              href="http://java.sun.com/products/autodl/j2se"/>
        <jar href="DynamicTreeDemo.jar"
            main="true" />

    </resources>
    <application-desc
         name="Dynamic Tree Demo Application"
         main-class=
           "webstartComponentArch.DynamicTreeApplication"
         width="300"
         height="300">
     </application-desc>
     <update check="background"/>
</jnlp>
```

Obrázek 11 - Ukazka JNLP souboru[8]

Dále je zapotřebí umístit odkaz na webový server a na www stránku. Jako odkaz se používá HTML zápis s atributem href, který specifikuje umístění JNLP souboru. Pro běh aplikace s JWS je nutné mít tuto technologii nainstalovanou [12].

[8] Zdroj: https://docs.oracle.com/javase/tutorial/deployment/webstart/deploying.html

3 IMPLEMENTACE HOTELOVÉHO ÚČETNICTVÍ V FRAMEWORKU VAADIN

Praktická část bakalářské práce obsahuje implementace aplikace hotelového účetnictví. Hlavním cílem je ukázka využiti frameworku Vaadin. Aplikace ve Vaadinu se programuje velmi podobně jako obyčejná desktopová aplikace i když je to v podstatě webová aplikace, protože potřebuje pro svou činnost aplikační server a je to klient-server aplikace.

Aplikace je zaměřena na poměrně malý podnik a poskytuje jednoduchý přehled a správu finančních prostředků hotelu. Hlavním cílem aplikace je spravování finančního toku peněz, uschovávání dat o výnosech, do kterých patří platby za ubytování, a nákladech na údržbu hotelu, jako například nákupy textilu, jídla, technická údržba, služby poskytované prádelnou atd. Tuto aplikaci lze dále rozšiřovat na plnohodnotný účetnický systém přidáním další funkcionality, rozšířením databázového modelu atd.

3.1 Požité technologie

V této kapitole budou popsány jednotlivé technologie, které byly použité v praktické časti bakalářské práce. Pro vývoj aplikace byl použit framework Vaadin verze 7.7.6.

3.1.1 Nastroj Maven

Pro správu a sestavování projektu byl použit nastroj Maven, který se nejčastěji využívá pro automatizace buildu, ale má i mnoho dalších funkcí, jako generování různých reportů, umožnění spouštění testů, nasazování aplikace na server. Repositář systému Maven obsahuje katalog artefaktů a systém pro jejich vyhledávání a stahování, pomocí něhož můžeme spravovat závislosti jednotlivých projektů.

Základem projektu je vytvoření objektového modelu nad zdrojovým kódem, se kterým lze provádět různé operace. Model projektu je definován v souboru pom.xml (POM – project object model), který se nachází v kořenovém adresáři každého projektu (případně v každém jednotlivém balíčku). V něm jsou definovány veškeré informace potřebné k provedení kompletního buildu projektu a taky uvedené informace o projektu, jeho závislostech, občas i informace o jednotlivých vývojářích nebo napojení na systémy pro správu verzí. Využití POM souboru, který obsahuje veškerou informaci potřebnou ke kompilaci a sestaveni projektu, poskytuje integritu do všech velkých IDE (Integrated Development Environment). Každý soubor pom.xml představuje jeden projekt (artefakt). Artefakt je identifikován skupinou (groupId), názvem (artifactId) a verzí (version) [13].

3.1.2 Webový server Jetty

Pro zpracování aplikace ve frameworku Vaadin byl použit webový server Jetty verze 9.3.9.

Jetty je open source webový server na platformě Java vyvíjený společností Mortbay Consulting pod licencemi Eclipse a Apache. Jedná se pouze o webový server, nikoli o plnohodnotný aplikační server. Jetty poskytuje funkcionalitu servlet kontejneru. Návrh serveru umožňuje samostatné spouštění aplikace i jeho integraci do jiné aplikace. Kromě toho obsahuje řadu souvisejících technologií jako jsou SPDY, webové sokety atd. Jetty server byl navržen, aby byl jednoduchý, výkonný, době použitelný, a aby se pro něj velmi snadno vytvářely pluginy.

3.1.3 Databáze Oracle 11g

Pro správu s daty bylo použité databázové uložiště od společnosti Oracle, kvůli velkému počtu její výhod, mezi které patři hlavně široka funkcionalita, kombinace vysoké urovni technologií a integrovaných řešení a technologie Flashback umožňující efektivní obnovu ztracených dat. Databáze Oracle je spolehlivá, poskytuje ACID (atomicita, konzistence, izolace, trvanlivost) test, který je důležitým nástrojem k zajištění integrity uložených dat.

Mezi největší nevýhodu databáze Oracle patří provozní náklady. Alternativou databázového systému může být například open source systém PostgreSQL nebo MySQL.

3.1.4 Ovladač JDBC

Pro práce z databáze bylo použito univerzální aplikační rozhrání – JDBC (Java Database Connectivity), zejména JDBC ovladač verze 12.1.0.2, což je driver určený pro samostatné klientské aplikace.

Výhodou takového postupu je to, že vývojář má plnou kontrolu nad SQL dotazy posílanými do databáze, a tím teoreticky také jistotu ve výkonnosti takového řešení. JDBC je natolik silným nástrojem, že pomocí něho lze bezproblémově volat PL/SQL funkce z klientských Java aplikací, pracovat s objektovými typy databáze atd. Také JDBC se používá k vytváření dynamických SQL dotazů. API je nezávislé na konkrétně databázi, všichni významní tvůrci databáze poskytuji ovladače JDBC pro jejich produkt.

Základem práce s JDBC je správce ovladačů – DriverManager. Každý výrobce JDBC rozhraní poskytuje vlastní drivery. Oracle poskytuje takových driver několik, každý z nich lze použít pro jiný typ klientské aplikace (Thin driver, OCI driver, server-side Thin driver, server-side internal driver).

3.1.5 Vývojové prostředí IntelliJ IDEA

IntelliJ IDEA od společnosti JetBrains je jedním s tří hlavních vývojových prostředí pro jazyk Java. Využívá se většinou pokročilejšími programátory a nadšenci. Pro vytvořenou aplikace bylo zvoleno IntelliJ IDEA verze 2016.3.6 Ultimate Edition.

Návrháři IntelliJ IDEA mají cit pro nastupující trendy a rychle doplňují podporu nových produktů. Například IntelliJ byl jedním z prvních řešení s robustní podporou Groovy a Grails a taky jedním z prvních, kdo nabízel vyspělou podporu pro JavScript včetně debuggeru. IDEA má velice kvalitní integraci pro Ant a Maven, podporu pro frameworky, jako Spring, Vaadin, právě to se stalo důvodem k použití IntelliJ IDEA.

Dnes toto IDE má možností a vlastnosti, které jsou lépe implementované než u konkurence. Mezi další výhody patří větší počet možných refaktoringů programu, zabudovaná kontrola syntaxe, která nejen hledá chyby, ale také místa, která sice pracují správně, ale jsou napsaná neohrabaně (např. příliš složité funkce nebo testování vždy pravdivé podmínky), svůj vlastní nástroj sledování programu, který zobrazuje v IDE, které řádky programu byly provedeny daným počtem testů atd. [14], [15].

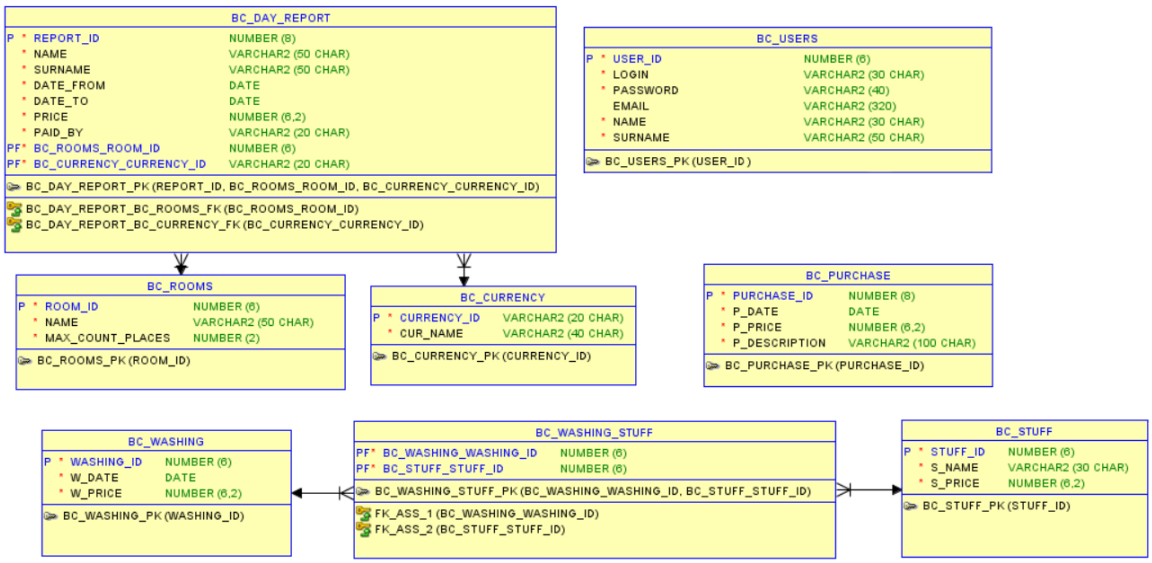
3.2 Databázová vrstva aplikace

Model databáze byl vytvořen v programu Oracle SQL Developer Data Modeler. Zde byl vytvořen konceptuální a relační model databáze. Data Modeler umožnuje automatické generování DDL skriptů při převodu z logického na fyzický model databáze. K aplikování a následnému zpracování vygenerovaného skriptu byl využit program Oracle SQL Developer, který poskytuje základní operace nad tabulkami, vytvoření a úpravu PL/SQL skriptů, zobrazení metadat a mnoho dalších funkcí. Primární klíče tabulek jsou automaticky generované pomoci příslušných sequence a triggeru.

Relační model obsahuje 8 tabulek. Následující jsou nejhlavnější:

- BC_DAY_REPORT – hlavní tabulka sloužící pro uchování transakci (platby za ubytování). Je to hlavní zdroj dosažení zisku pro hotel.
- BC_PURCHASE – tabulka obsahující údaje o nákladech.
- BC_WASHING – tabulka obsahující data o praní, což je velmi důležitou, ale i drahou časti práce hotelu.

S těchto třech tabulek se pak spočítají statistické údaje, jako jsou například čistý zisk a ztráta, buď to roční nebo měsíční.



Obrázek 12 – Relační model databáze

Pro správu s daty byla vytvořena řada příslušných procedur a funkcí. Funkce, určené pro zpracování statistických údajů:

- BC_GET_CURRENCY_MONTH – funkce vrátí nejpoužívanější měnu měsíce.
- BC_GET_MONTH_PROFIT – funkce spočítá měsíční zisk.
- BC_GET_NET_PROFIT – spočítáni čistého zisku vybraného měsíce.
- BC_GET_STATISTIC – funkce vrátí rozdíl zisku mezi vybraným a předchozím měsícem.
- BC_GET_YEAR_PROFIT – spočítaní ročního zisku.
- BC_GET_YEAR_NET_PROFIT – funkce vrátí čistý zisk za vybraný rok.

Ostatní funkce:

- BC_CREATE_WASHING – procedura určená pro přidáváni dat do tabulky BC_WASHING.
- MD5HASH – funkce, která má za úkol hashovaní hesel.
- BC_DELETE_WASHING – funkce, která v aplikace bude použita pro vymazání záznamu z tabulky BC_WASHING.

```
CREATE OR REPLACE FUNCTION BC_GET_NET_PROFIT(in_month NUMBER, in_year NUMBER)
  RETURN NUMBER IS
  var_purchases NUMBER;
  var_washing   NUMBER;
  var_profit    NUMBER;
  BEGIN
   SELECT NVL(SUM(P_PRICE), 0)
   INTO var_purchases
   FROM BC_PURCHASE
   WHERE EXTRACT(MONTH FROM p_date) = in_month AND EXTRACT(YEAR FROM p_date) = in_year;
   SELECT NVL(SUM(W_PRICE), 0)
   INTO var_washing
   FROM BC_WASHING
   WHERE EXTRACT(MONTH FROM w_date) = in_month AND EXTRACT(YEAR FROM w_date) = in_year;

   SELECT BC_GET_MONTH_PROFIT(in_month, in_year)
   INTO var_profit
   FROM dual;
   var_profit := var_profit - (var_purchases + var_washing);
   RETURN var_profit;
END BC_GET_NET_PROFIT;
```

Obrázek 13 – Ukázka implementace funkce BC_GET_NET_PROFIT v databáze

3.2.1 Třida SimpleJDBCConnectionPool

Pro práci s JDBC ovladačem byla použita třida SimpleJDBCConnectionPool[9], kterou poskytuje Vaadin. Je to zjednodušená implementace rozhraní JDBCConnectionPool. Třída má na starosti načtení ovladače JDBC, nastavení, používaní a uzavření připojení. Používaní této třidy usnadňuje práce programátoru, není potřeba psát ručně metody pro připojení k databázi, což je dobrým způsobem šetření času. Třída má tři metody - reserveConnection(), releaseConnection() a destroy().

```
private void setUpConnection() {
    try {
        connectionPool = new SimpleJDBCConnectionPool(
                driverName: "oracle.jdbc.OracleDriver",
                connectionUri: "jdbc:oracle:thin:@fei-sql1.upceucebny.cz:1521:ee11",
                userName: "st43280",
                password: "********",
                initialConnections: 2,
                maxConnections: 5);
         lblInfo.setValue("Connected to database!");
    } catch (SQLException ev) {
        ev.printStackTrace();
         lblInfo.setValue("Connection failed...");
    }}
```

Obrázek 14 – Ukázka kódu metody setUpConnection()

[9] Dokumentace SimpleJDBCConnectionPool: http://computerworld.cz/vyvoj/nejlepsi-nastroje-na-programovani-v-jave-42939

Metoda setUpConnection() (obrázek 14) vytvoří objekt třídy SimpleJDBCConnectionPool. Tento objekt je pak využit v metodách pracujících s databázi. Přiklad takové metody je uveden na obrázku 15.

```
private Double getGetYearNETProfit(int year) {
    Double var_result = 0.0;
    try {
        Connection conn = connectionPool.reserveConnection();
        String query = " SELECT BC_GET_YEAR_NET_PROFIT(?) as result from dual";

        PreparedStatement stm = conn.prepareStatement(query);
        stm.setInt( parameterIndex: 1, year);
        ResultSet rs = stm.executeQuery();
        while (rs.next()) {
            var_result = rs.getDouble( columnLabel: "result");
        }
        connectionPool.releaseConnection(conn);
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return var_result;
}
```

Obrázek 15 – Ukázka práce s objektem třídy SimpleJDBCConnectionPool

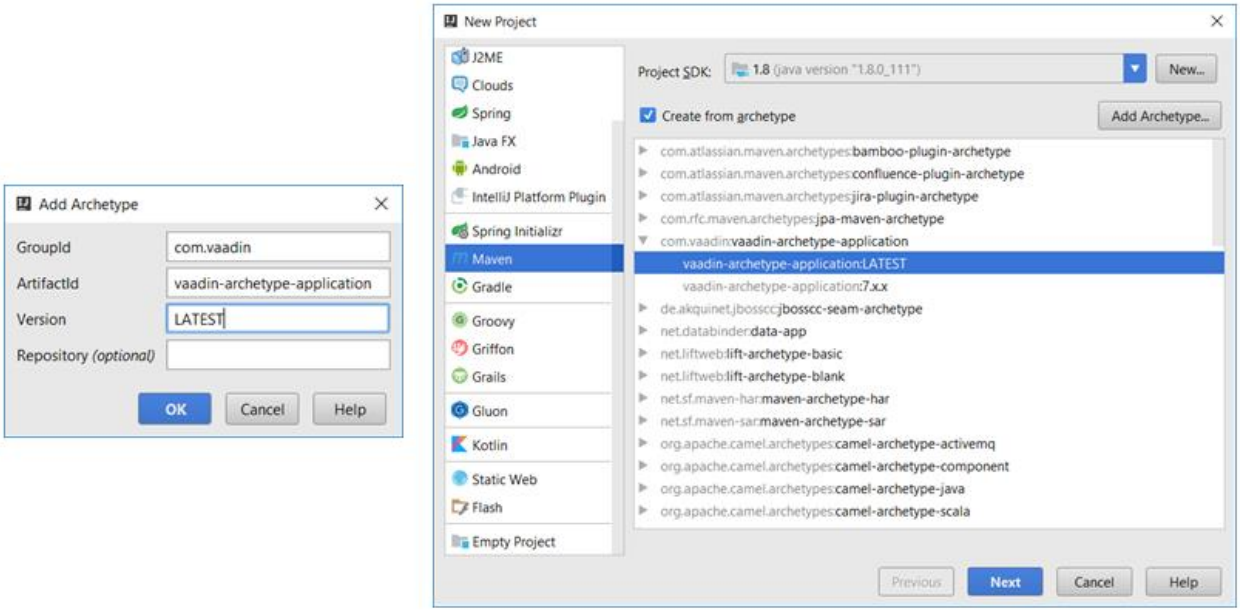
3.3 Aplikační vrstva aplikace

Tato podkapitola obsahuje postup vytvoření aplikace, hlavní třídy, metody a nuance, které je zapotřebí vědět pro práci s frameworkem Vaadin.

3.3.1 Vytvoření a první spouštění Vaadin aplikace

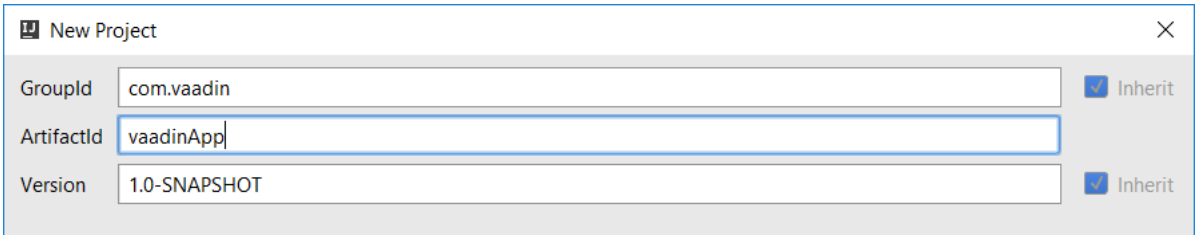
Pro účely této aplikace byl do IntelliJ IDEA nainstalován plugin pro rozhraní Vaadin. Postup vytváření aplikaci ve Vaadinu je následující:

1. Vybrat New Project v menu File.
2. V New Project zvolit nastroj Maven.
3. Připojit Java SDK, které se použije v projektu (pro projekt Vaadin je zapotřebí mít minimálně Java 8).
4. Zaškrtnout Create from archetype a uvést následující údaje:
   GroupId: com.vaadin,
   ArtifactId: vaadin-archetype-application,
   Version: LATEST (případně vybrané číslo verze) (obrázek 16).
5. Pak po přidaní archetypu se volí com.vaadin: vaadin-archetype-application: LATEST a pokračuje dal (obrázek 17). V následujících projektech už je zapotřebí jenom vybrat archetype a zmačknout Next.



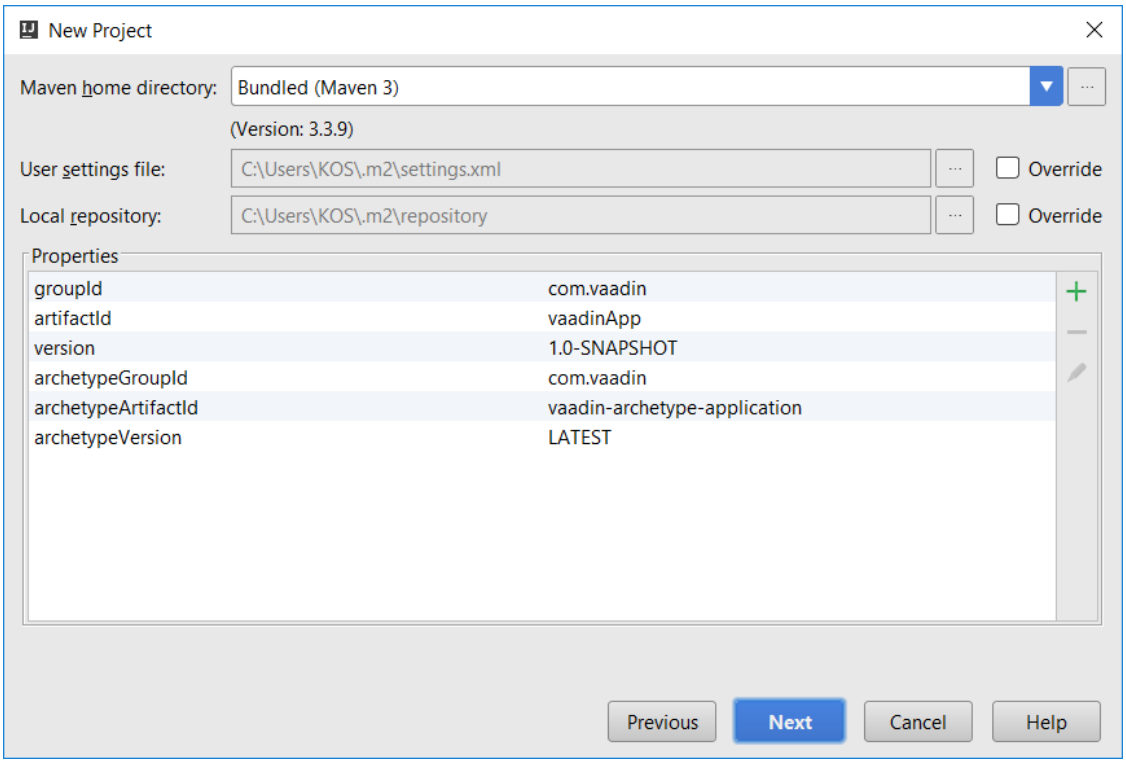
Obrázek 16 – Přidávaní archetypu Vaadin

6. Následujícím krokem je pojmenování GroupID, ArtifactID a Version (je možnost použit výchozí nastaveni).



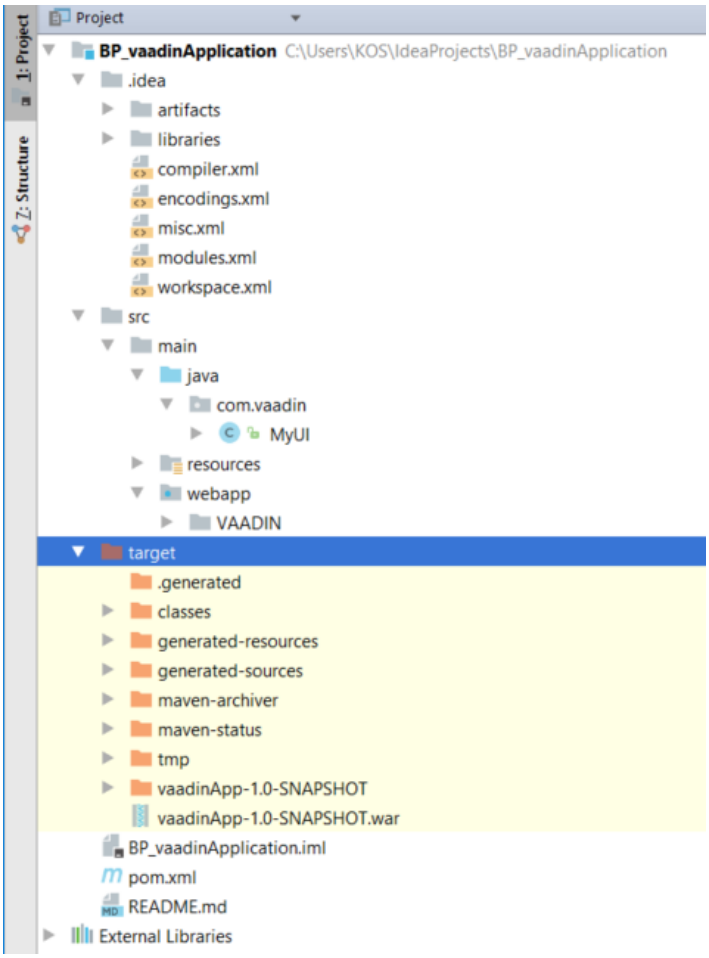
Obrázek 17 – Vytvoření projektu Vaadin (krok 6)

7. V dalším okně důkladně zkontrolovat údaje (zejména při prvním vytvoření projektu). (obrázek 18), pak pojmenovat projekt.

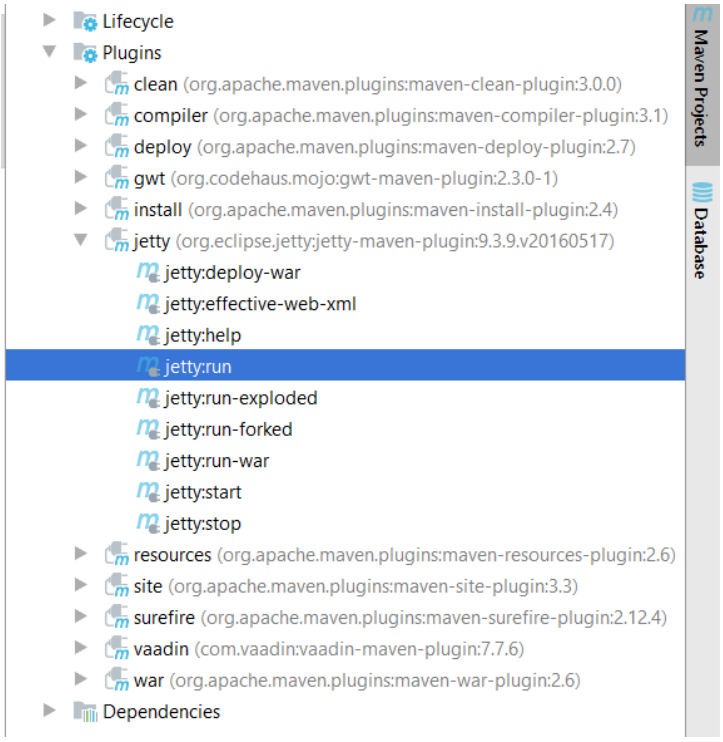


Obrázek 18 – Vytvoření projektu Vaadin (krok 7)

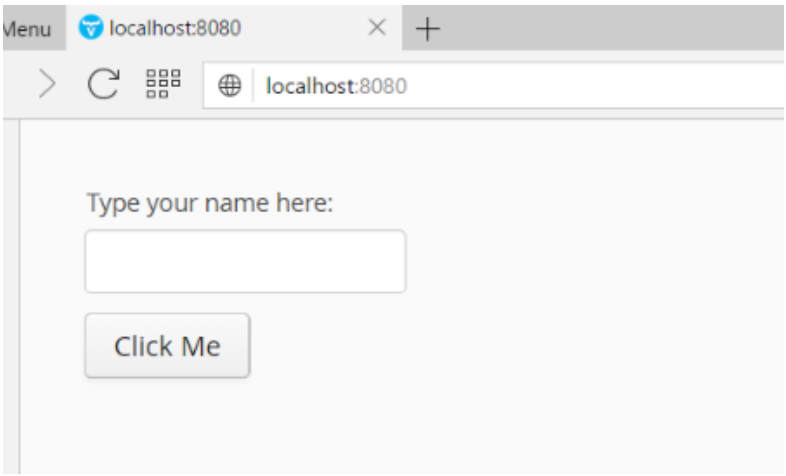
Po vytvoření prázdného projektu je vygenerována jeho typická struktura. Předtím, než se začne psát jakýkoliv kód aplikace, je zapotřebí stáhnout nutné knihovny, přidat závislosti a sbalit projekt. Toto všechno lze udělat pomoci Mavenu. Proto v panelu nástrojů Maven Project v sekci Lifecycle zmačknout Clean, pak – Install. Takovým způsobem do projektu se připojí potřebné knihovny a struktura projektu bude mít vzhled, jako na obrázku 19. Dalším krokem otevřít Maven Projects, zvolit Plugins a spustit aplikace pomoci jetty:run Tato volba spustí jetty-server na portu 8080.



Obrázek 19 – Struktura projektu nově vytvořené Vaadin aplikace



Obrázek 20 – Ovládací panel Mavenu

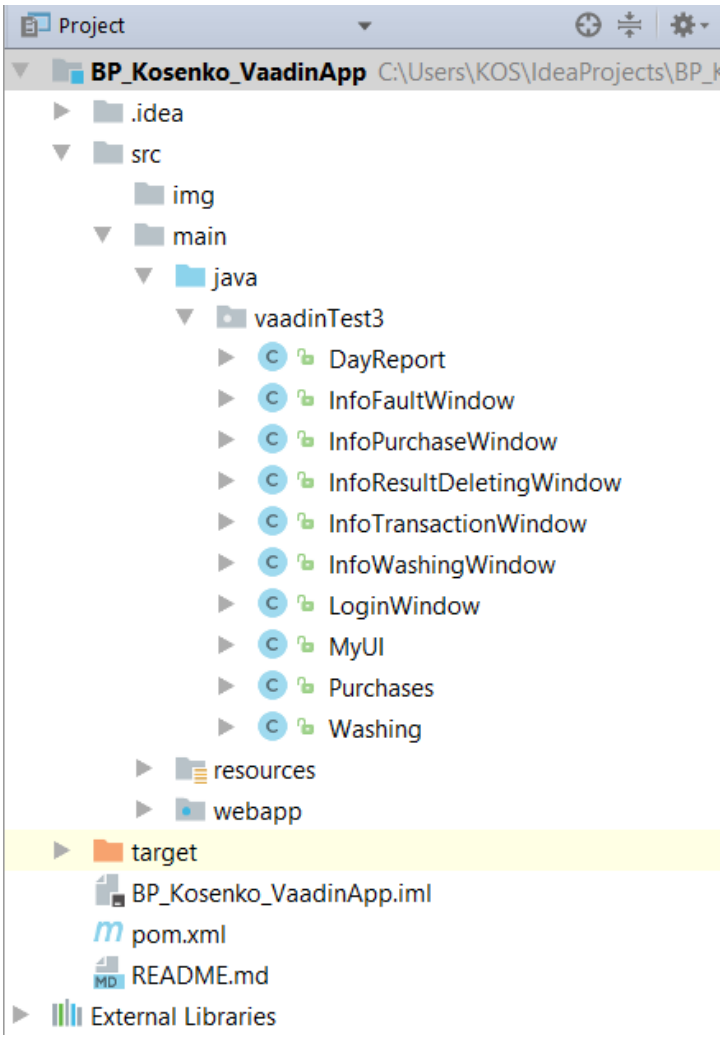


Obrázek 21 – Ukázka vytvořené Vaadin aplikace s výchozími nastavení

V adresáři target se nachází sbalený war-soubor. Zdrojové kódy aplikace se nachází ve složce src. Důležitým je soubor pom.xml, kde jsou uvedené všechny závislosti. Knihovny potřebné pro fungování aplikace se nachází ve složce Libraries. Soubory se šablonami, které definují styly jsou ve složce webapp/VAADIN [16].

3.3.2 Struktura projektu

Hlavní třídou aplikace je MyUI, která obsahuje 90 % funkcionality aplikace. Ostatní třídy jsou pomocné a slouží buď pro vytváření objektu, který se pak použije při naplnění tabulek daty (DayReport, Purchases, Washing) nebo pro zobrazení dialogového okna s výsledky činnosti uživatele (InfoFaultWindow, InfoPurchaseWindow, InfoResultDeletingWindow, InfoWashing Window) nebo pro přihlášení (LoginWindow).



Obrázek 22 – Struktura aplikace

3.3.3 Přihlášení uživatele

K ověření a přihlášení uživatele slouží třida LoginWindow, která obsahuje 3 metody:

- LoginWindow(), což je v podstatě konstruktor této třidy. V něm se nastaví všechny potřebné parametry a elementy,
- setUpConnection() – metoda určená pro připojení k databáze.
- getLogin() – metoda se spojí s databázi a porovnává uvedené údaje s daty, která se nachází v tabulce BC_USERS.

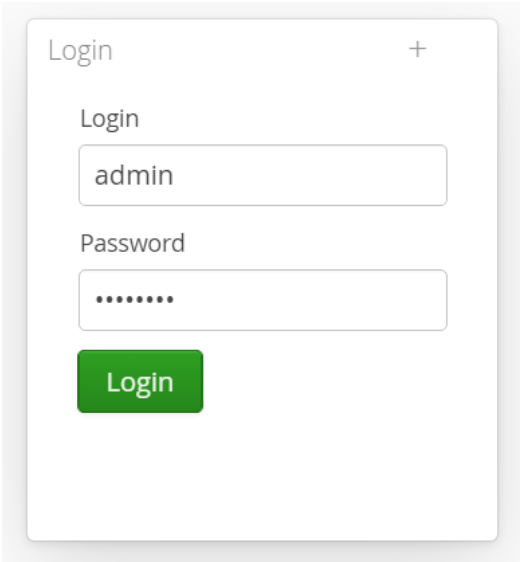
Třída LoginWindow představuje dialogové okno, jehož hlavními komponentami jsou TextField a PasswordField, kam uživatel zadává své přihlašovací údaje. Pokud parametry jsou správně zadané, dialogové okno se zavře a uživatel může dal pracovat s aplikací. Pokud údaje nebudou správné, uživatel nemůže pokračovat dál. Je to způsobeno absencí tlačítka zavření okna.

```
LoginWindow loginWindow = new LoginWindow();
UI.getCurrent().addWindow(loginWindow);
```

Obrázek 23 – Ukázka kódu otevření dialogového okna

```
private void getLogin() {
    if (txtFLogin.isEmpty() || pswd.isEmpty()) {
        lblInfo.setValue("Enter password and login");
    } else {
        int rowCount = 0;
        try {
            Connection conn = connectionPool.reserveConnection();
            String query = "SELECT COUNT(*) as row_count FROM BC_USERS " +
                    "WHERE LOGIN=? AND PASSWORD=MD5HASH(?)";
            PreparedStatement stm = conn.prepareStatement(query);
            stm.setString( parameterIndex: 1, txtFLogin.getValue());
            stm.setString( parameterIndex: 2, pswd.getValue());
            ResultSet rs = stm.executeQuery();
            while (rs.next()) {
                rowCount = rs.getInt( columnLabel: "row_count");
            }
            connectionPool.releaseConnection(conn);
            if (rowCount == 1) {
                lblInfo.setValue("Info: Successful!");
                getSession().setAttribute("admin", txtFLogin.getValue());
                connectionPool.destroy();
                close();
            } else {
                pswd.clear();
                lblInfo.setValue("Wrong password or login");
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }}}
```

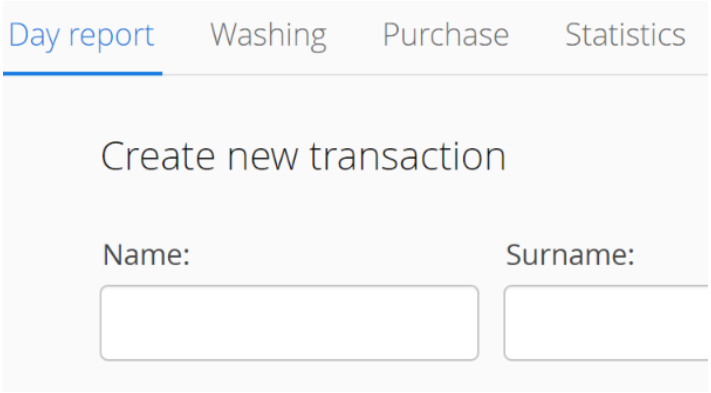
Obrázek 24 – Ukázka kódu metody getLogin() pro přihlášení uživatele



Obrázek 25 – Dialogové okno přihlášení

3.3.4 Ovládací panel

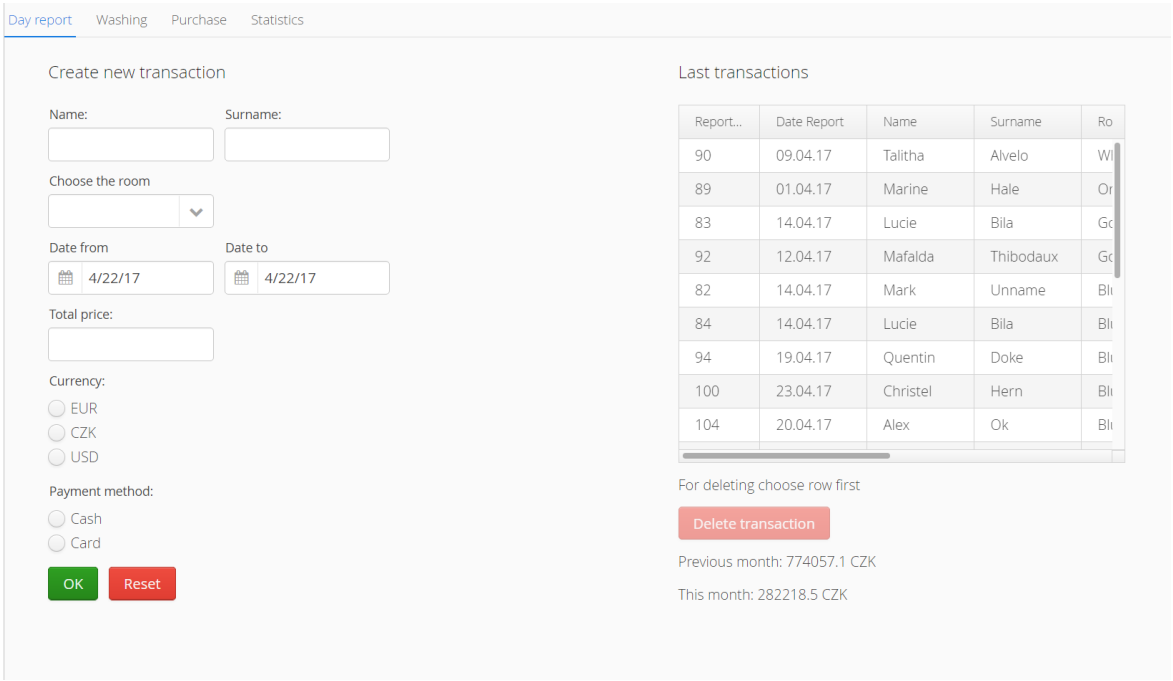
Jako ovládací panel byl použit TabSheet. Je to kontejner, který umožňuje přepínání mezi komponenty pomocí záložek. Přepínač pro záložky se nachází v levé horní části aplikace.



Obrázek 26 – Ovládací panel

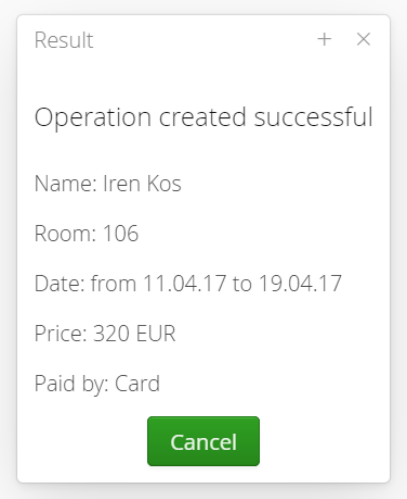
Day Report

Záložka má na starosti vytvoření, vymazání a přehled jednotlivých transakcí. Pro vytvoření transakce uživatel musí zadat jméno a příjmení hosta, vybrat pokoj který byl rezervován, datum příjezdu a odjezdu, cenu, měnu, a způsob platby. Jsou to základní údaje, které je zapotřebí mít pro vedení účetnictví.



Obrázek 27 – Sekce „Day Report“

Za vytvoření transakce je zodpovědná metoda createReport(), která sčítá data z příslušných komponent a pošle požadavek o vytvoření transakce databází pomoci objektu třídy PreparedStatement. V případě úspěchu se otevře dialogové okno s výsledkem (viz obrázek 28). Kód této metody je znázorněn na obrázku 29.



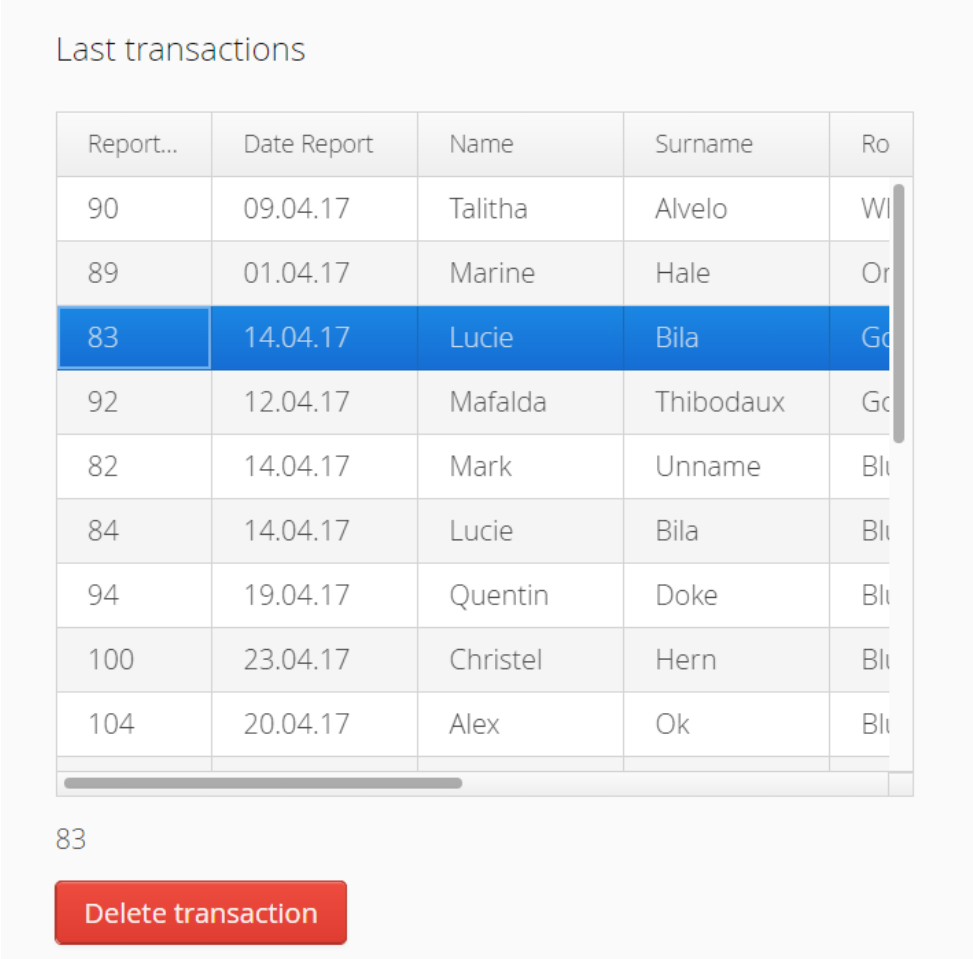
Obrázek 28 – Dialogové okno s výsledkem přidáni nové transakce

```
private void createReport() {
    Date v = rep_dateFrom.getValue();
    String var_dateFrom = new SimpleDateFormat( pattern: "dd.MM.yy").format(v);
    Date v2 = rep_dateTo.getValue();
    String var_dateTo = new SimpleDateFormat( pattern: "dd.MM.yy").format(v2);
    String strName = rep_name.getValue() + " " + rep_surname.getValue();
    String strDates = "from " + var_dateFrom + " to " + var_dateTo;
    String strPrice = rep_TxtPrice.getValue() + " " + rep_OptionCurrency.getValue();
    String strRoom = rep_comboRoom.getValue().toString();
    String strPaidBy = rep_OptionPayment.getValue().toString();
    try {
        Connection conn = connectionPool.reserveConnection();
        String insert = "INSERT INTO BC_DAY_REPORT(name, surname, date_from, date_to," +
                "price,paid_by, bc_rooms_room_id,bc_currency_currency_id)" +
                "values(?, ?, TO_DATE(?),TO_DATE(?),?,?,?,?)";
        PreparedStatement stmt = conn.prepareStatement(insert);
        stmt.setString( parameterIndex: 1, rep_name.getValue());
        stmt.setString( parameterIndex: 2, rep_surname.getValue());
        stmt.setString( parameterIndex: 3, var_dateFrom);
        stmt.setString( parameterIndex: 4, var_dateTo);
        stmt.setDouble( parameterIndex: 5, Double.parseDouble(rep_TxtPrice.getValue()));
        stmt.setString( parameterIndex: 6, rep_OptionPayment.getValue().toString());
        stmt.setInt( parameterIndex: 7, (Integer) rep_comboRoom.getValue());
        stmt.setInt( parameterIndex: 8,  x: 101);
        stmt.executeUpdate();
        conn.commit();
        connectionPool.releaseConnection(conn);
        transactionResult = new InfoTransactionWindow(strName, strDates, strRoom, strPrice, strPaidBy);
        UI.getCurrent().addWindow(transactionResult);
    } catch (SQLException e) {
        e.printStackTrace();
        faultResult = new InfoFaultWindow();
        UI.getCurrent().addWindow(faultResult);
    }
    rep_resetAll();
}
```

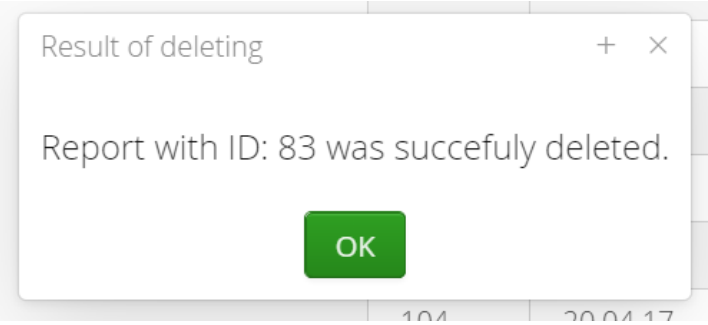
Obrázek 29 – Metoda určená pro přidáni nové transakce do tabulky BC_DAY_REPORT

V pravé části této záložky se nachází tabulka s vytvořenými transakcemi, tlačítko pro vymazání a jednoduchá statistika, která ukazuje zisk za předchozí a daný měsíc. Víc podrobnější statistiku uživatel obdrží v záložce Statistics.

Pro vymazání záznamu z tabulky zaprvé je zapotřebí zpřístupnit tlačítko „Delete transaction“, které není k dispozici, pokud není vybrán záznam v tabulce. Po vybraní řádku v tabulce (vybraný řádek bude označen modrou barvou), tlačítko se zpřístupní a uživatel může řádek odstranit (obrázek 30). Pokud vymazání proběhne v pořádku na obrazovce se objeví dialogové okno s výsledkem (obrázek 31).



Obrázek 30 – Vymazaní řádku z příslušné tabulky



Obrázek 31 – Dialogové okno s výsledkem vymazaní záznamu

```
private void deleteTransaction(int id) {
    try {
        Connection conn = connectionPool.reserveConnection();
        String delete = " DELETE FROM BC_DAY_REPORT WHERE REPORT_ID = ?";
        PreparedStatement stmt = conn.prepareStatement(delete);

        stmt.setInt( parameterIndex: 1, id);
        stmt.executeUpdate();

        conn.commit();
        connectionPool.releaseConnection(conn);
        infoResult = new InfoResultDeletingWindow( result: "Report with ID: " + id + " was succefuly deleted.");
        UI.getCurrent().addWindow(infoResult);
    } catch (SQLException e) {
        e.printStackTrace();
        faultResult = new InfoFaultWindow();
        UI.getCurrent().addWindow(faultResult);
    }

}
```

Obrázek 32 – Metoda deleteTransaction() zodpovědná za vymazaní záznamu z tabulky BC_DAY_REPORT

Velice podobně byly zpracované i záložky Purchase a Washing, hlavní odlišnost od Day Report je v tom, že v sekci Purchase se pracuje s tabulkou BC_PURCHASE, ve Washing – s tabulkou BC_WASHING.

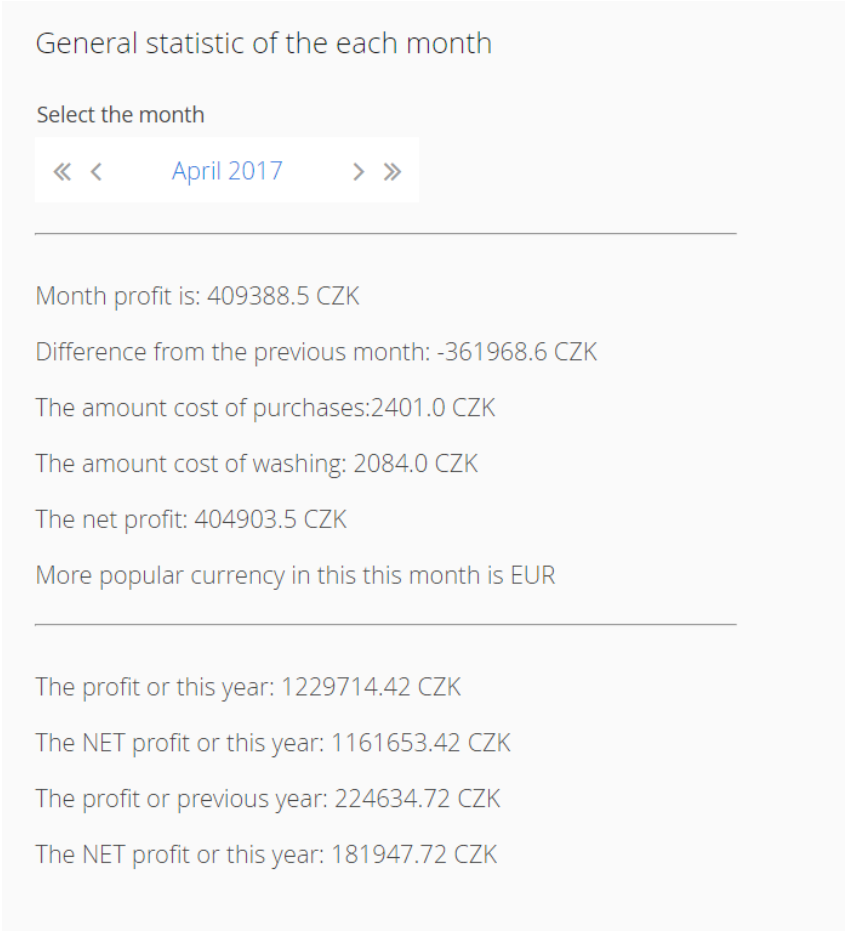
Statistics

Pro úspěšné vedení podniku je vždy potřeba vědět o finančním rozvoji, případně úpadku firmy. Pro to se používají statistické a analytické údaje. Nejjednodušší způsob zjištění stavu firmy je porovnání její zisku a ztráty za předchozí rok, nebo za předchozí měsíc. Z toho důvodu do aplikace byla přidaná část obsahující aspoň minimální statistické údaje.

V této částí aplikace pracuje s tabulkou BC_DAY_REPORT a také s řadou procedur a funkcí (viz kapitola 3.2).

V levé části se nachází měsíční a roční statistika. Pomocí komponenty InlineDateField, která je zpracovaná takovým způsobem, že se zobrazuje pouze měsíc a rok, uživatel může velmi rychle vybrat potřebný měsíc a prohlédnout jeho údaje (obrázek 34).

Pravá část záložky obsahuje 2 DataField, pomocí kterých uživatel může vybrat period „Od – do“ a prohlédnout tabulku se všemi transakcemi za tento period (viz obrázek 36).

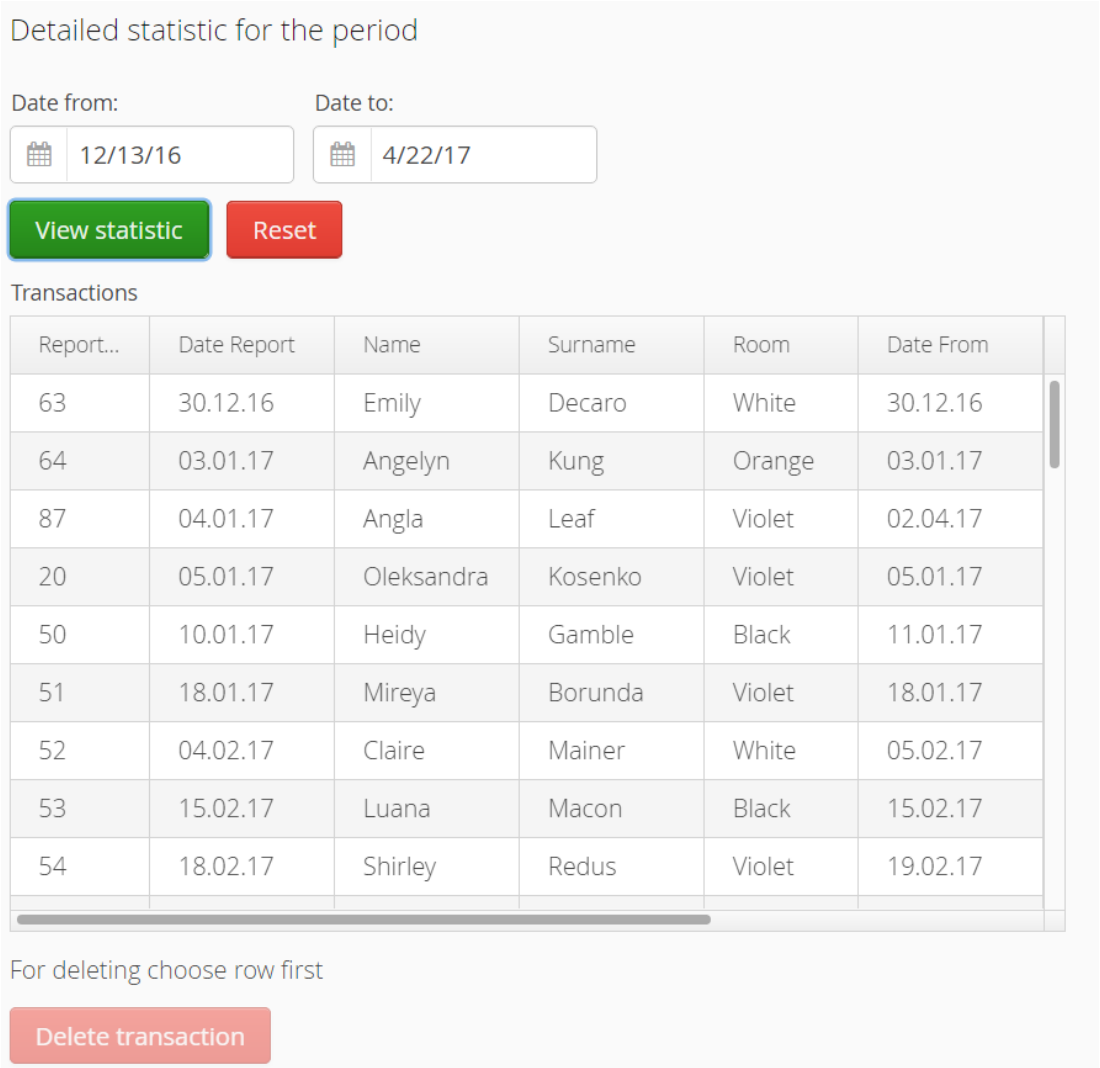


Obrázek 33 – Zobrazeni statistických údajů v aplikaci

```
private void getGetNetProfit(int month, int year) {
    String var_result = null;
    try {
        Connection conn = connectionPool.reserveConnection();
        String query = "SELECT BC_GET_NET_PROFIT(?,?) as result from dual";

        PreparedStatement stm = conn.prepareStatement(query);
        stm.setInt( parameterIndex: 1, month);
        stm.setInt( parameterIndex: 2, year);
        ResultSet rs = stm.executeQuery();
        while (rs.next()) {
            var_result = rs.getString( columnLabel: "result");
        }
        stat_LblNetProfit.setValue("The net profit: " + var_result + " CZK");
        connectionPool.releaseConnection(conn);
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

Obrázek 34 – Ukázka kódu jedné z příslušných metod pro zobrazeni statistických údajů



Obrázek 35 – Zobrazeni tabulky, která obsahuje transakce za vybraný period

4 ZÁVĚR

Hlavní náplní bakalářské práce bylo seznámení s technikami spuštění desktopové aplikace na webu, jejich problematika a vyřešení.

V teoretické části zaprvé byly připomenuté druhy Java aplikace, jejích výhody a nevýhody. Dále v kapitole 2 byly popsané techniky spuštění desktopové aplikace na webu, mezi které patři donedávna nejpoužívanější Java Applet a všemi známý Java Web Start. Byla vysvětlena problematika práce z Java Applety.

V praktické části bakalářské práce byla úspěšně navržená a implementována aplikace ve Vaadinu. Důvodem k tomu, že byl použit framework Vaadin je totální zablokování Java Appletu, které už v době psaní bakalářské práce nebylo možné spouštět ve většině webových prohlížečů, pravděpodobně z bezpečnostních důvodů. Vaadin (stejně jako i JSF) se nejvíc blíží k desktopovému návrhu. Aplikace se programuje pomoci Vaadin komponent, které jsou v podstatě už webové, ale jsou velice podobné komponentám knihovny Swing nebo JavaFX. V této kapitole také byl podrobněji popsán postup vytvoření a spuštění Vaadin aplikace, což se docela liší od vytvoření obyčejné desktopové aplikace, kvůli práci s Mavenem.

Tato aplikace by mohla být použita podnikatelem, který je začátečníkem v hotelnictví a zatím nepotřebuje rozsáhlý systém na spravování účetnictví. Aplikace ale se dá rozšiřovat o další komponenty a přidávat větší funkcionalitu. V budoucnosti by se mohla stát plnohodnotným systémem vedení hotelu.

V současné době uživatelé dávají přednost více webovým aplikacím než desktopovým. Složitost vývoje www aplikace se občas stává problémem pro vývojáře, zejména pro začátečníky, nebo pro ty, kteří mají větší zkušenosti s tvorbou desktopových GUI aplikací. Existují však technologie, které by mohli stát za vyřešením nebo aspoň kompromisem v takové situaci. Mezi takové patří Java Web Start a frameworky.

Na závěr dá se říct, že v současné době nejlepším řešením problému spuštění GUI aplikace na webu, případně vývoje webové aplikace podobně jako desktopové, je využití frameworku, kvůli jejích popularitě, stabilní a průběžné aktualizaci a podpoře, ale jsou to v podstatě už webové klient-server aplikace.

POUŽITÁ LITERATURA

[1] MAPLE, Simon: *Java Tools and Technologies Landscape Report 2016.* [online]. Release date: July 14, 2016. URL: https://zeroturnaround.com/rebellabs/java-tools-and-technologies-landscape-2016/

[2] LOY, Marc a Robert ECKSTEIN: *Java Swing*. 2nd ed. Sebastopol, CA: O'Reilly, c2003. ISBN 0596004087.

[3] SCHILDT, Herbert: *Java: the complete reference.* Ninth edition. ISBN 978-0-07-180855-2.

[4] PUREWAL, Semmy: *Learning web app development*. 1 edition. O'Reilly Media, 2014. ISBN 978-1449370190.

[5] BOLLIER, David a Bharathi NATARAJAN: *JSP Java Server Pages: podrobný průvodce začínajícího tvůrce.* Praha: Grada, c2003. Moderní programování. ISBN 80-247-0340-8.

[6] HALL, Marty: *Java Servlety a stránky JSP*. 1.vyd. Praha: Neocortex s.r.o., 2001. 586 s. ISBN 80-86330-06-0

[7] BURNS, Ed., Chris. SCHALK a Neil GRIFFIN: *JavaServer faces 2.0: the complete reference*. New York: McGraw-Hill, c2010. ISBN 9780071625098.

[8] JOHNSON, Rod a další: *Spring framework Reference Documentation* [online]. SpringSource Inc, 2012. URL: http://static.springsource.org/spring/docs/3.1.x/spring-framework-reference/html/

[9] GRÖNROOS, Marko: *Book of Vaadin: Vaadin 7 Edition - 7th Revision.* Vaadin Ltd., Published: 2016-04-19. Dostupné z: https://vaadin.com/download/book-of-vaadin/vaadin-7/pdf/book-of-vaadin.pdf

[10] Kolektiv autoru: *The Final Countdown for NPAPI.* Blog.chromium.org [online]. 2014. URL: https://blog.chromium.org/2014/11/the-final-countdown-for-npapi.html

[11] SMEDBERG, Benjamin: *NPAPI Plugins in Firefox.* Future Releases [online]. URL: https://blog.mozilla.org/futurereleases/2015/10/08/npapi-plugins-in-firefox/

[12] MARINILLI, Mauro: *Java Deployment with JNLP and Webstart*. Indianapolis, Ind.: Sams, c2002. ISBN 0-672-32182-3.

[13] Kolektiv autoru: *Welcome to Apache Maven.* Apache Maven Project [online]. Last Published: 2017-04-27. URL: http://maven.apache.org/index.html

[14] FIELDS, Duane K., Stephen. SAUNDERS a Eugene. BELAYEV. *IntelliJ IDEA in action*. London: Pearson Education, 2006. ISBN 978-193-2394-443.

[15] BINSTOCK, Andrew: *Nejlepší nástroje na programování v Javě: Eclipse, IntelliJ IDEA, NetBeans i Oracle JDeveloper pokračují v tradici bohatých a pestrých vývojových nástrojů pro Javu.* COMPUTERWORLD [online] 2011-03-14. URL: http://computerworld.cz/vyvoj/nejlepsi-nastroje-na-programovani-v-jave-42939

[16] Kolektiv autoru: *Creating a Project with IntelliJ IDEA.* Vaadin. [online] URL: https://vaadin.com/docs/-/part/framework/getting-started/getting-started-idea.html

PŘÍLOHY

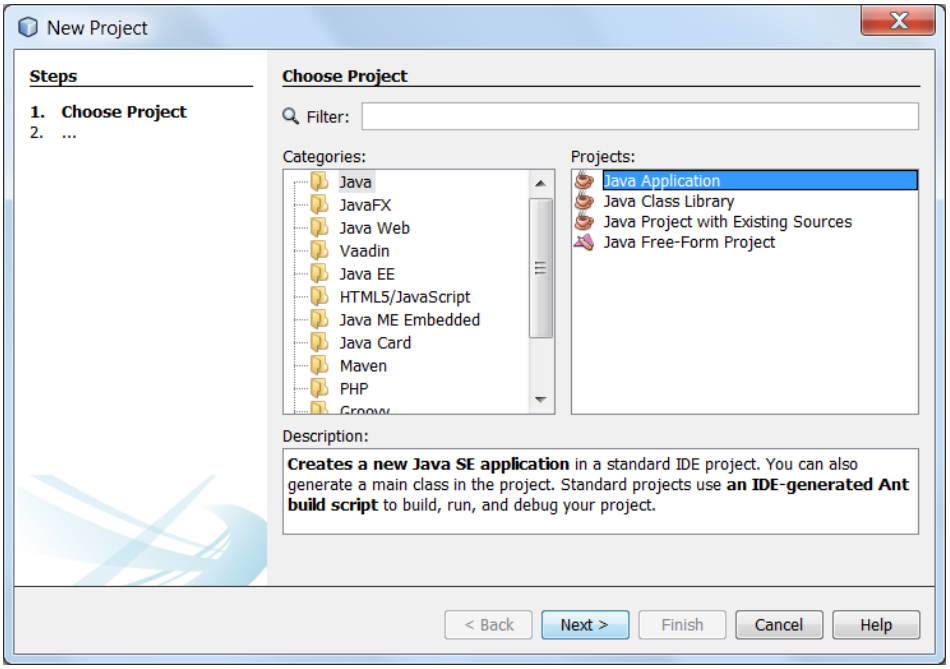
OBSAH CD

- KosenkoO_DesktJavaApNaWeb_ZŠ_2017.pdf
- VaadinApp_BP_Kosenko.zip
- DB_Script.pdf

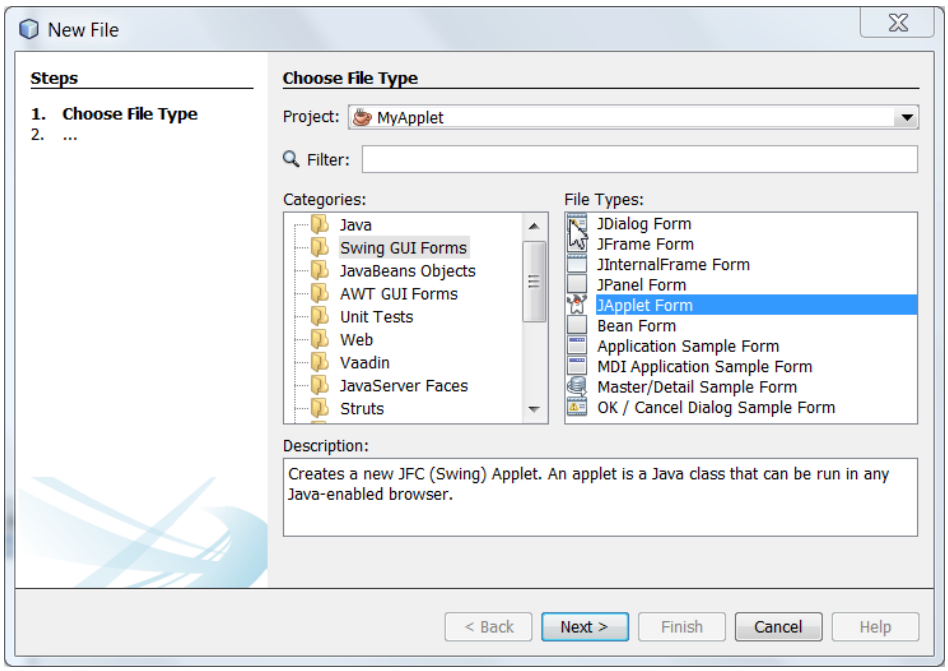
Příloha A – *Vytvoření a spuštěni Java Appletu ve webovém prohlížeče – Firefox*

Příloha popisuje postup práce s Java Appletem, jeho vytvoření a spuštění ve webovém prohlížeči Firefox. Tento postup byl vytvořen v roce 2016. Od roku 2017 spuštěni appletu ve Firefoxu už není možné.

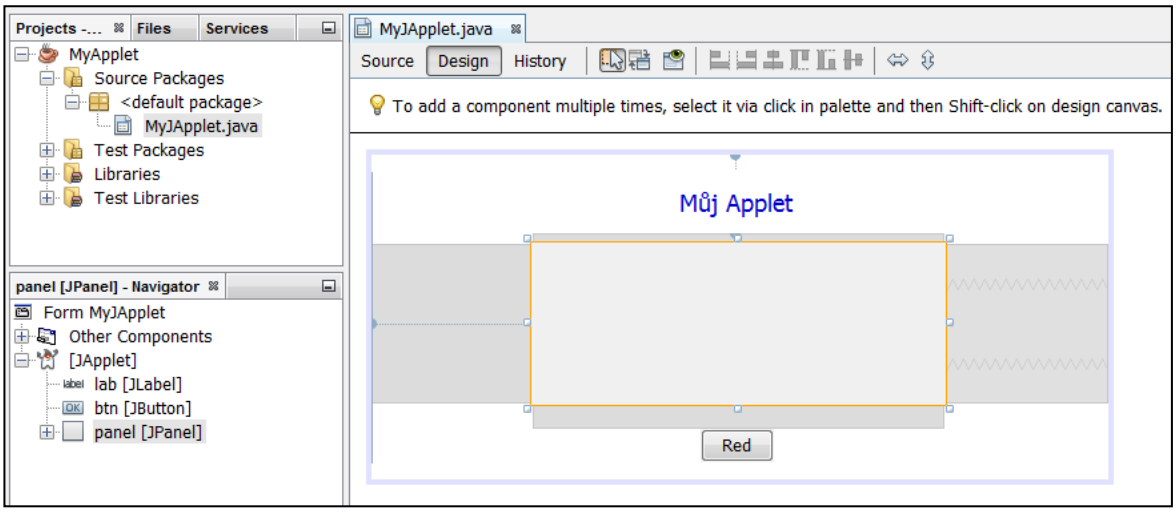
Prvním krokem je vytvořeni prázdného Java Application projektu.



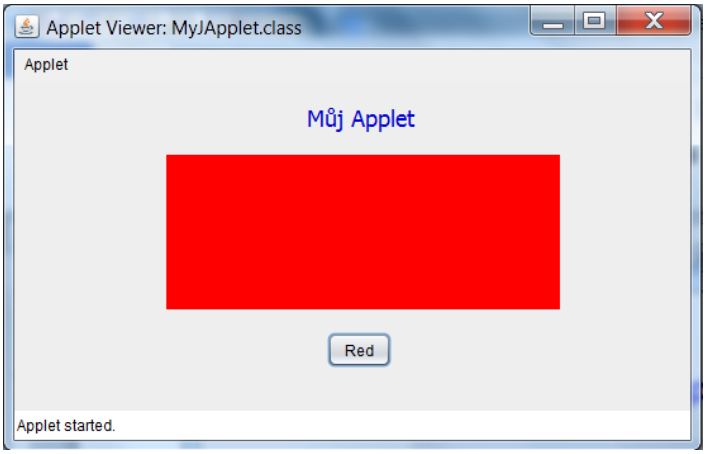
Dalším krokem je přidáni JApplet From do projektu.



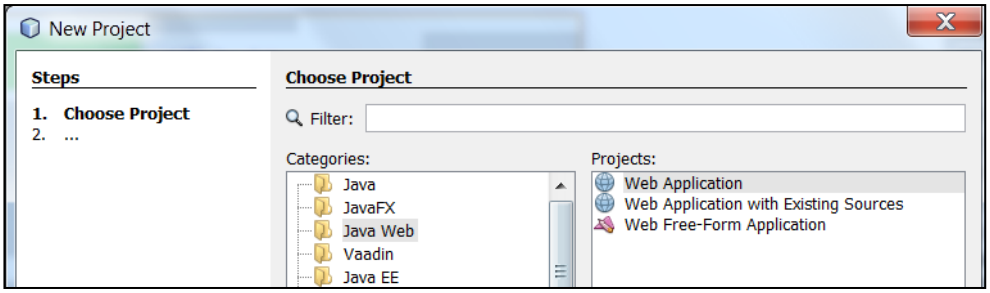
Dále bude realizován jednoduchý design. Čtenář samozřejmě může vytvořit jiný design podle svých požadavků.



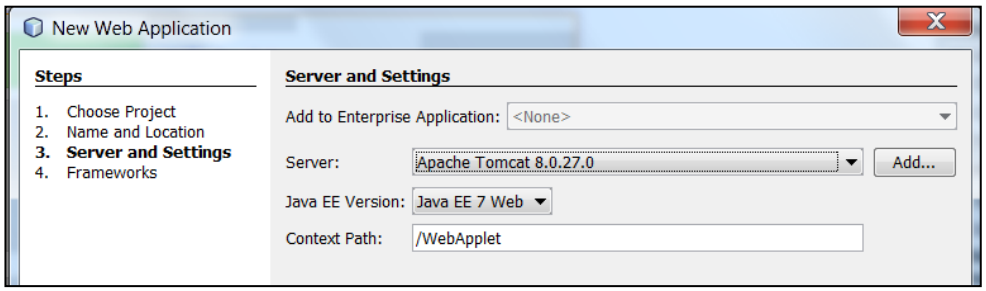
Pro tlačítko přidat následující kód, který po stisknuti tlačítka změní barvu pozadí. Spustit applet pomoci Shift-F6 a stisknout tlačítko.



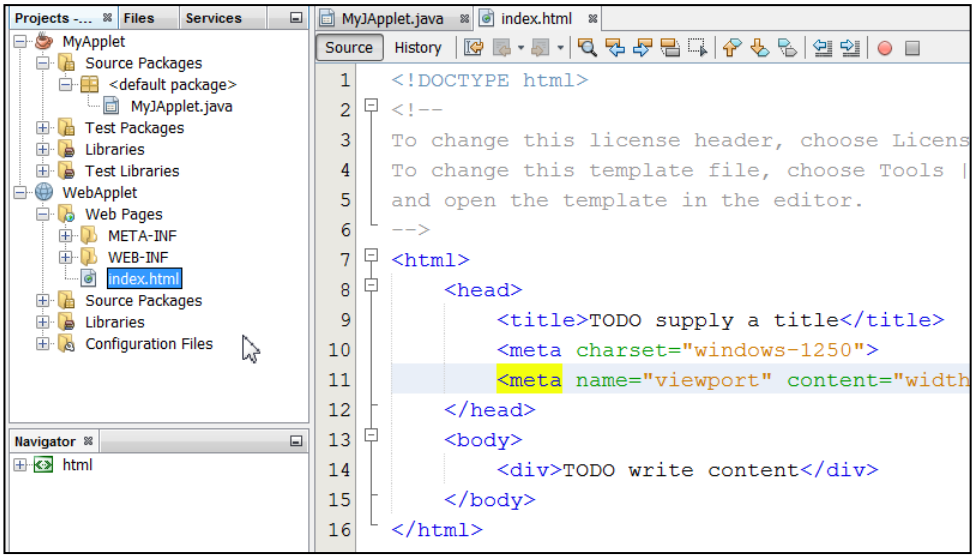
Po úspěšnému spuštěni apletu jako GUI aplikace ho musíte nyní vložit do webové aplikace. Vytvořte nový Web Project a pojmenujte ho WebApplet (Java Web -> Web Application):



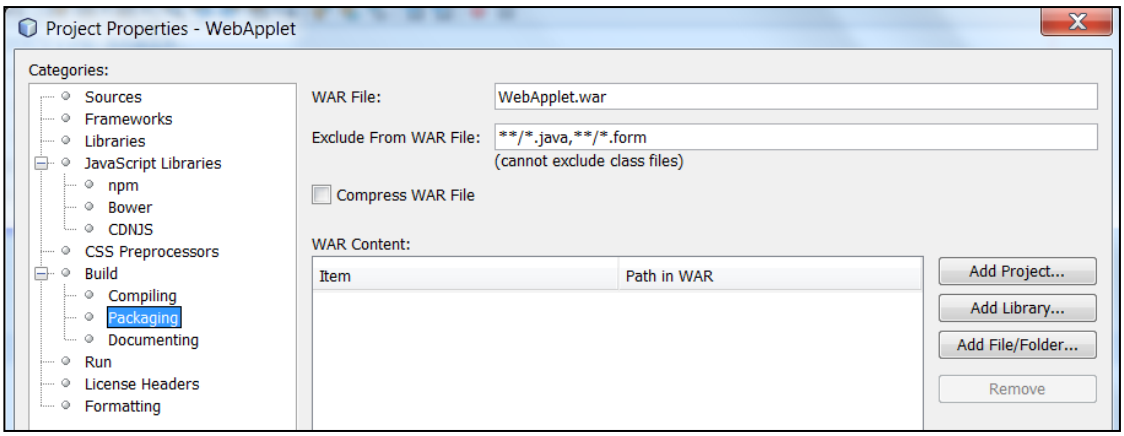
Následně vyberte vhodný aplikační server. V našem případě je to Apache Tomcat.



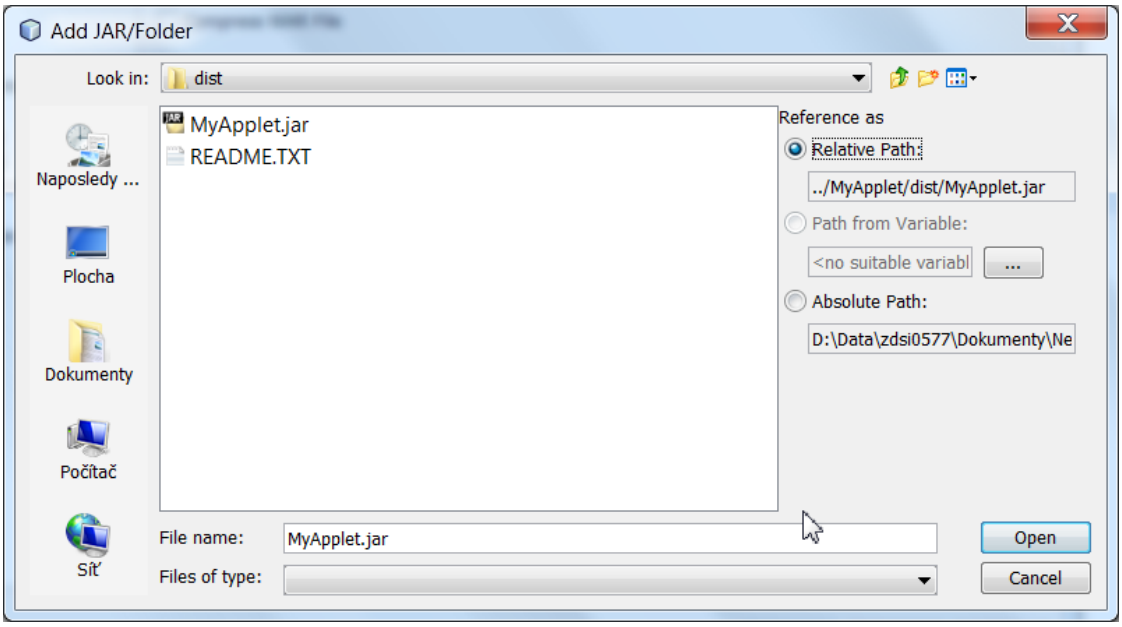
A po stisknuti tlačítka „Finish“ mate hotovou vygenerovanou HTML stránku index.html.

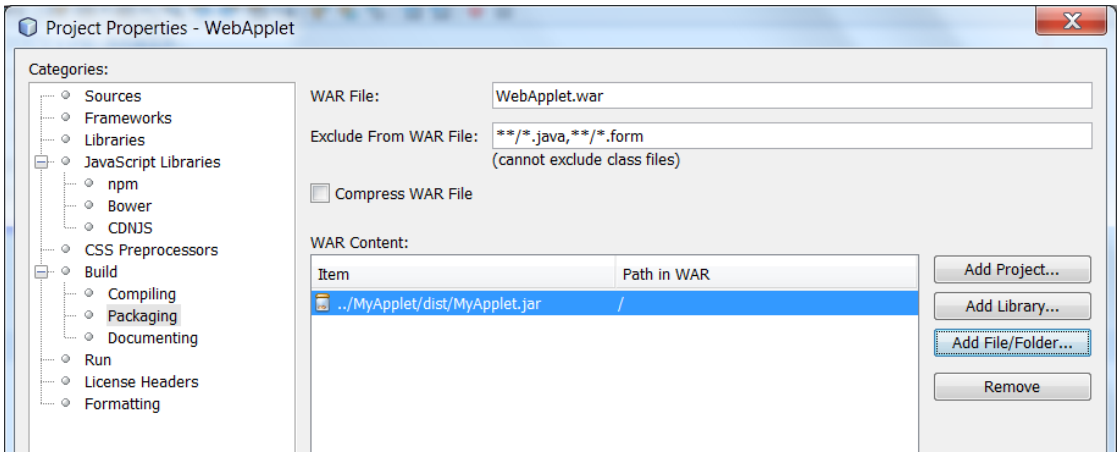


Dalším krokem je přidání souboru MyApplet.jar, který se nachází ve složce „dist“, do webového projektu. Otevřete Properties projektu WebApplet, v Packaging stisknete tlačítko Add File/Folder.



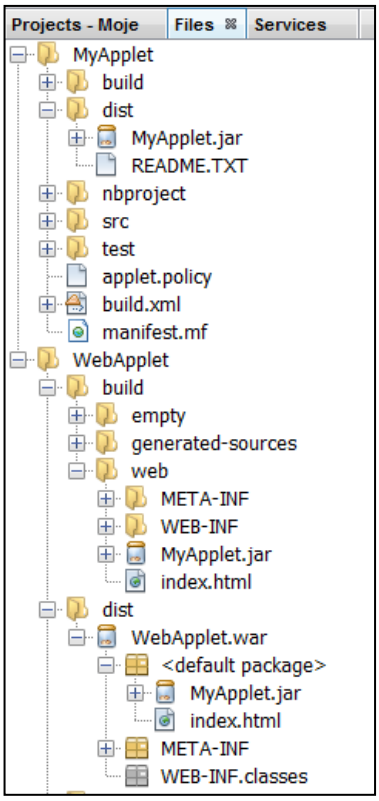
Otevře se okno, kde pak najdete potřebný soubor. Pote zmačkněte tlačítko Open.





V Menu Run zvolte Build Project (WebApplet) anebo zmačkněte klávesu F11.

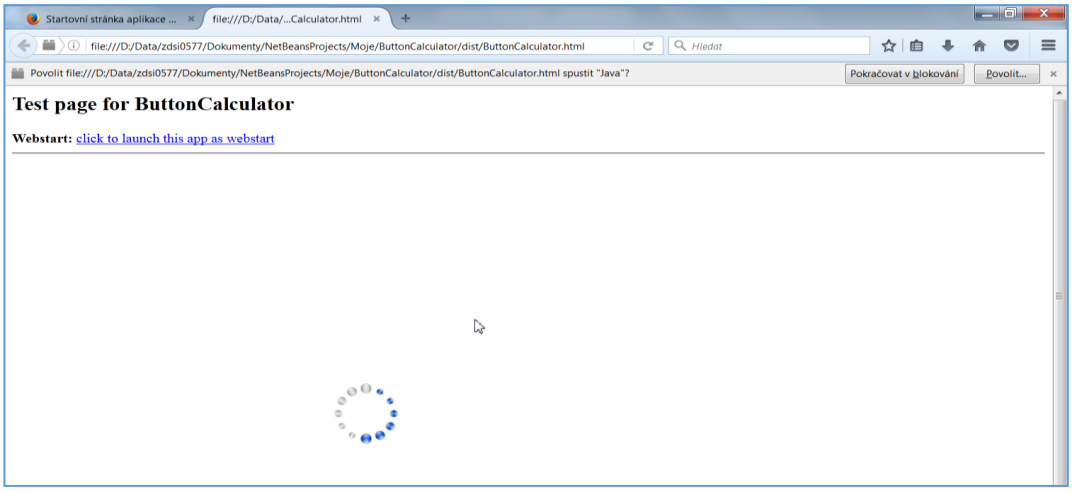
Po buildingu webového projektu je soubor MyApplet.jar přidán do souboru WebApplet.war (spolu s index.html), tuto informace můžete zkontrolovat v záložce Files.



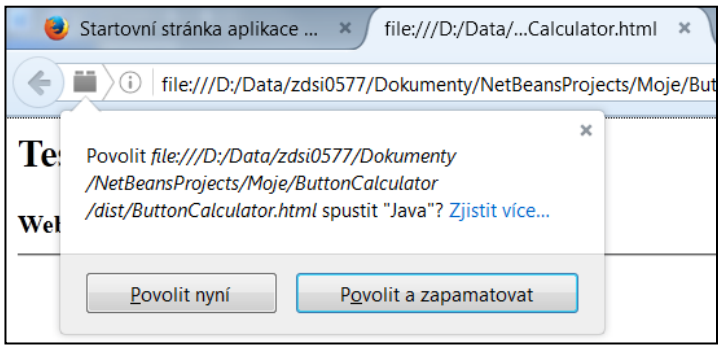
Následně modifikujte stránku index.html vložením následujícího kódu:

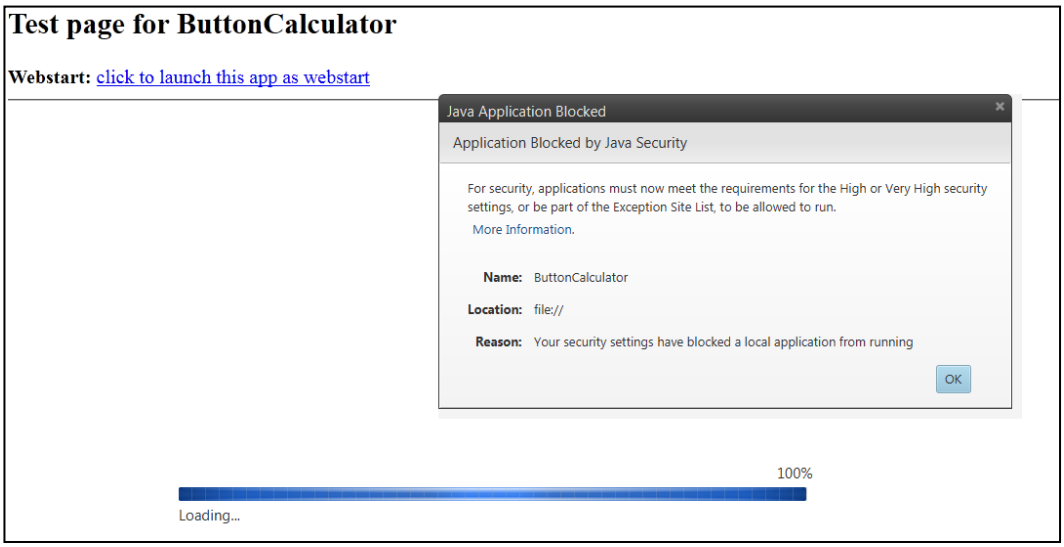
```
<body>
    <h3>A jsme s apletem na webu</h3>
    <applet code="MyJApplet" archive="MyApplet.jar" width="450" height="220" />
</body>
```

Pro spuštěni appletu v prohlížeči musíte přidat výjimku. Spusťte vygenerovanou HTNL stránku ve FireFoxu a začne se spouštět Vaše aplikace. Stiskněte tlačítko Povolit.

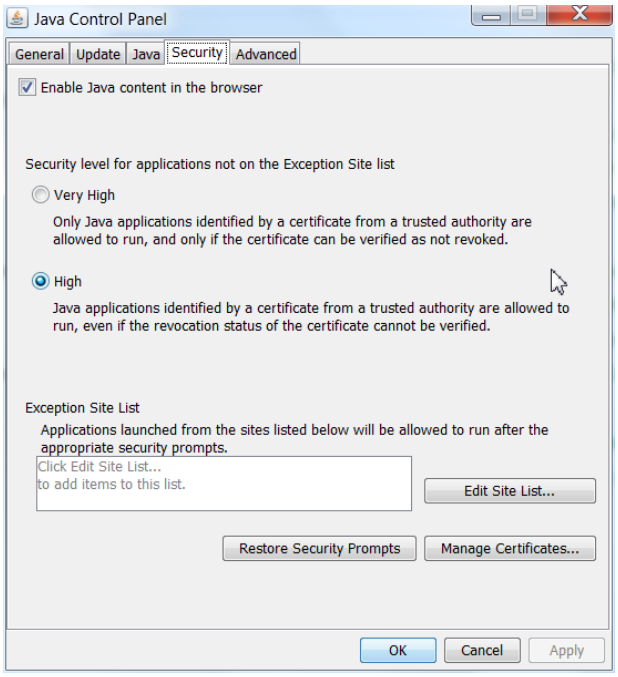


Zvolte volbu „Povolit nyní“ a stiskněte OK pro další kroky odblokování zabezpečeni.

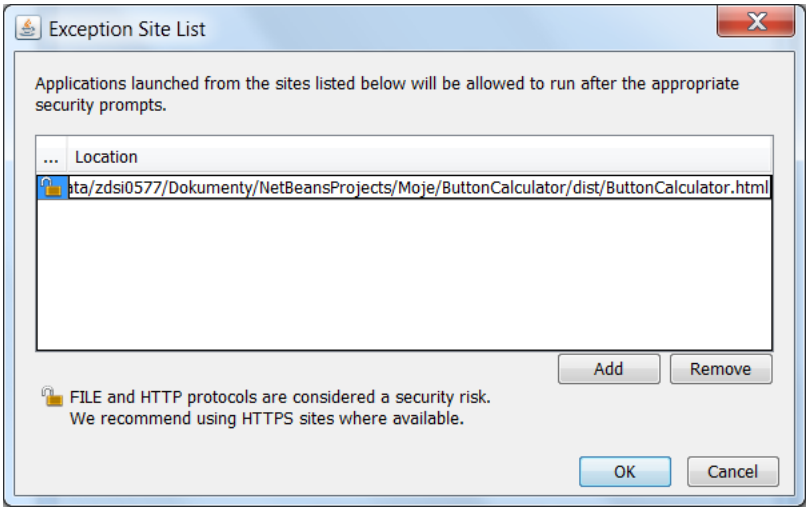




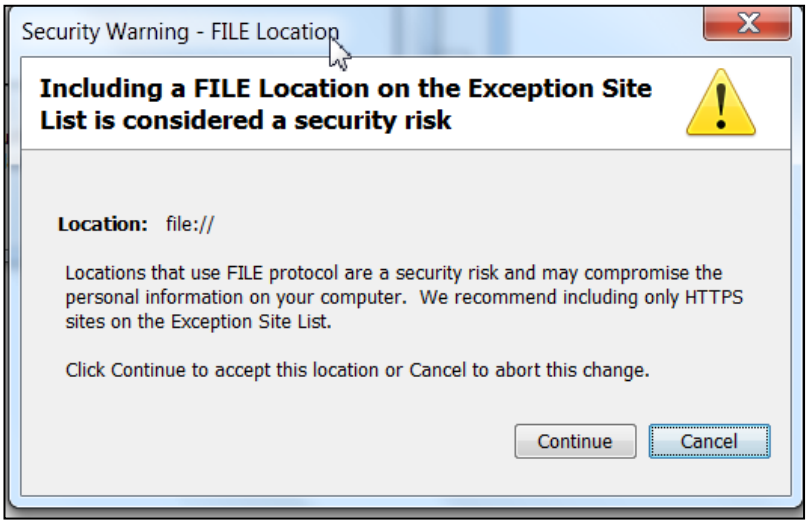
Spusťte správu Javy přes Control Panel (Ovládací panely), v záložce Security stiskněte Edit Site List.

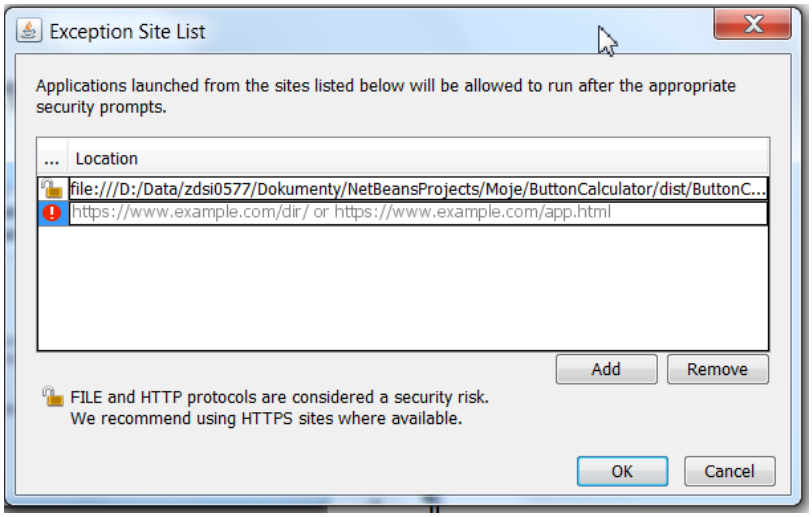


V dialogovém okně vyberte Add a překopírujte do řádku celou adresu Vaší aplikace a stiskněte znovu Add.



Objeví-li se okno s varováním, zvolte Continue a pak stiskněte OK.





Po přidáni výjimky spusťte applet v prohlížeči Firefox. Zaprvé zvolte Build a pak Run.