Mendelova univerzita v Brně
Provozně ekonomická fakulta

---

# Návrh webové aplikace pro interakci s mobilním robotem

**Bakalářská práce**

Vedoucí práce:
Ing. František Ostřížek

Martin Kedroutek

Brno 2016

Děkuji svému vedoucímu bakalářské práce Ing. Františkovi Ostřížkovi, konzultantům Ing. Janu Kolomzaníkovi, Ph.D. a Ing. Jiřímu Lýskovi, Ph.D. za jejich rady při psaní bakalářské práce. Dále chci poděkovat své rodině za poskytnutou podporu v mém studiu a v neposlední řadě všem mým spolužákům a přátelům z kolejí.

**Čestné prohlášení**

Prohlašuji, že jsem tuto práci: **Návrh webové aplikace pro interakci s mobilním robotem**
vypracoval samostatně a veškeré použité prameny a informace jsou uvedeny v seznamu použité literatury. Souhlasím, aby moje práce byla zveřejněna v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách ve znění pozdějších předpisů, a v souladu s platnou *Směrnicí o zveřejňování vysokoškolských závěrečných prací.*
Jsem si vědom, že se na moji práci vztahuje zákon č. 121/2000 Sb., autorský zákon, a že Mendelova univerzita v Brně má právo na uzavření licenční smlouvy a užití této práce jako školního díla podle § 60 odst. 1 Autorského zákona.
Dále se zavazuji, že před sepsáním licenční smlouvy o využití díla jinou osobou (subjektem) si vyžádám písemné stanovisko univerzity o tom, že předmětná licenční smlouva není v rozporu s oprávněnými zájmy univerzity, a zavazuji se uhradit případný příspěvek na úhradu nákladů spojených se vznikem díla, a to až do jejich skutečné výše.

V Brně dne 22. května 2016 .....

## Abstract

Kedroutek, M. Design of a web application to interact with the mobile robot. Bachelor thesis. Brno, 2016.

Bachelor thesis deals with design and implementation of web application provading interaction with the mobile robot. As part of the application is created the user interface, which through a robot providing a guide role, allowing his calling and guide on user-selected location.


**Keywords:** Java EE, JSF, geolocation, robot


## Abstrakt

Kedroutek, M. Návrh webové aplikace pro interakci s mobilním robotem. Bakalářská práce. Brno, 2016.

Bakalářské práce se zabývá návrhem a implementací webové aplikace zajišťujicí interakci s mobilním robotem. V rámci aplikace je vytvořeno uživatelské rozhraní, které prostřednictvím robota zastávajícího roli průvodce, umožní jeho přivolání a průvod arboretem na uživatelem zvolené místo.


**Klíčová slova:** Java EE, JSF, geolokace, robot

# Obsah

**1 Úvod a cíl práce** **11**
1.1 Úvod do problematiky . . . . . 11
1.2 Cíl práce . . . . . 11

**2 Technologie pro vývoj a implementaci aplikace** **12**
2.1 Programovací jazyk Java . . . . . 12
2.2 Enterprise Java Beans . . . . . 12
2.3 JavaServer Faces . . . . . 15
2.4 Prime Faces . . . . . 15
2.5 JavaScript . . . . . 16
2.6 Asynchronní webové aplikace s technologií AJAX a AJAJ . . . . . 17
2.7 Značkovací jazyk HTML . . . . . 18
2.8 CSS . . . . . 19
2.9 Databázový systém . . . . . 20
2.10 Testování webových aplikací . . . . . 21
2.11 Geolokace . . . . . 22
2.12 IBM bluemix . . . . . 23
2.13 Autonomní robot BuggyMan . . . . . 24
2.14 Použité vývojové nástroje . . . . . 25

**3 Metodika řešení** **26**
3.1 Požadavky aplikace . . . . . 26
3.2 Vývoj aplikace . . . . . 26

**4 Praktická část** **27**
4.1 Grafický návrh aplikace . . . . . 27
4.2 Průběh průvodu . . . . . 28
4.3 Adresářová struktura aplikace . . . . . 29
4.4 Konfigurační soubory . . . . . 30
4.5 Prezentační vrstva . . . . . 31
4.6 JavaScrpit . . . . . 33
4.7 Backendová vrstva . . . . . 35
4.8 Testování aplikace . . . . . 40

**5 Závěr** **41**

**6 Seznamy** **42**

**7 Reference** **43**

# 1 Úvod a cíl práce

## 1.1 Úvod do problematiky

Bakalářská práce se zabývá tvorbou webové aplikace. Je realizována v rámci projektu BuggyMan, vyvíjeného týmem Aistorm Mendelovy univerzity v Brně.

Návrh a řešení webové aplikace zprostředkovává průvod uživatele po areálu arboreta Mendelovy univerzity s pomocí autonomního robota BuggyMan, jenž zastupuje roli průvodce po arboretu. Tento autonomní robot je určen pro pohyb po zpevněných cestách, doprovází návštěvníka arboreta na zvolené místo, které návštěvník vybere prostřednictvím webové aplikace. Tato unikátní aplikace je jedinečná svým specifickým využitím, je vyvinuta za účelem propagace univerzitního arboreta, skupiny Aistorm a celé Mendelovy univerzity.

Webová aplikace je vyvíjena na platformě IBM Bluemix, v prostředí J2EE (Java Enterprise Edition). Platforma Bluemix umožňuje neomezený přístup a úpravu aplikace.

Podmínkou pro použití aplikace je provoz na zařízení s lokalizačním systémem. Lokalizace uživatele je nutná k odeslání polohy uživatele pro BuggyMana. V aplikaci je řešeno zarezervování robota na doprovod, pokud BuggyMan v danou chvíli provádí jiné uživatele po arboretu, tak je potřeba zajistit, aby nově příchozí uživatel nemohl přerušit právě probíhající doprovod.

Pro vlastní implementaci aplikace je využito několika technologií z prostředí Java Enterprise Edition, jako jsou JavaServer Faces, Apache Maven a další webové technologie JavaScript, MySQL apod. Paralelně s webovou aplikací je zároveň vyvíjen nový modul pro BuggyMana, jehož prostřednictvím je webové aplikaci umožněno ovládání a přístup k BuggyManovi.

## 1.2 Cíl práce

Cílem bakalářské práce je navrhnout a implementovat webovou aplikaci, která zajišťuje interakci s mobilním robotem. V rámci aplikace je vytvořeno uživatelské rozhraní, které prostřednictvím robota zastávajícího roli průvodce umožní přivolání a průvod arboretem na uživatelem zvolené místo. Průvod arboretem zajišťuje mobilní robot BuggyMan, ten po zvolení vytipovaného místa uživatelem provede přesun na zvolené místo. Navržený koncept je implementován pomocí programovacího jazyka JAVA a J2EE technologí jako jsou JSF (JavaServer Faces), PrimeFaces apod. do webové aplikace.

Dalším krokem je nutné vytvořit seznam vytipovaných míst v areálu arboreta, ke kterým je umožněn doprovod pomocí BuggyMana. K vytipovaným místům je mimo jejich popis nutné zaevidovat lokalizační data.

# 2 Technologie pro vývoj a implementaci aplikace

Pro implementaci webové aplikace je nutné zvolit vhodné vývojové frameworky, které se postupem času neustále zdokonalují. Díky frameworku se zdokonaluje bezpečnost webových aplikací a zjednodušuje se práce při vývoji celé aplikace. Zobrazení prvků na stránce zajišťují značkovací jazyky HTML, XHTML atd. Design webových stránek se vytváří pomocí kaskádových stylů, tyto styly jsou pro tvorbu webové aplikace nezbytné. Pro ukládání dat bude zvolena vhodná forma databáze.

Aplikace je realizována prostřednictvím cloudové platformy IBM Bluemix, jenž zajišťuje provoz aplikace.

## 2.1 Programovací jazyk Java

Programovací jazyk Java byl původně vyvíjen v laboratořích firmy SUN jako jazyk pro malé domácí spotřebiče, pracovně nazvaný Oak. Jeho tvůrci vycházeli z jazyka C++. V roce 1995 byl Oak přejmenován a byla představena první verze jazyka Java. (HEROUT, P., 2003)

Java je rozdělena do tří edic dle cílového zařízení. Základní edice Java SE (Standard Edition) slouží k vývoji aplikací pro stolní počítače. Její podmnožinou je Java ME (Micro Edition), která je určena pro vestavná zařízení (mobily, domácí elektronika atp.). Poslední edicí, která přidává do Javy SE funkcionalitu pomocí dodatečných knihoven, je Java EE (Enterprise Edition). Java EE má rozsáhlé uplatnění na poli serverových aplikací, ať se již jedná o bankovní aplikace, informační systémy nebo pouze o dynamické webové stránky. (Čápka, 2015)

Hlavní výhody jazyka Java:

- **objektově orientovaný** s výjimkou osmi primitivních datových typů jsou všechny ostatní datové typy objektové
- **distribuovaný** podporuje různé úrovně síťového spojení a umožňuje vytvářet distribuované klientské aplikace a servery
- **interpretovaný** místo skutečného strojového kódu se vytváří bajtkód, to přináší nezávislost na architektuře počítače nebo zařízení
- **robustní** je určen pro psaní vysoce spolehlivého softwaru
- **nezávislý na architektuře** vytvořená aplikace běží na libovolném operačním systému nebo libovolné architektuře

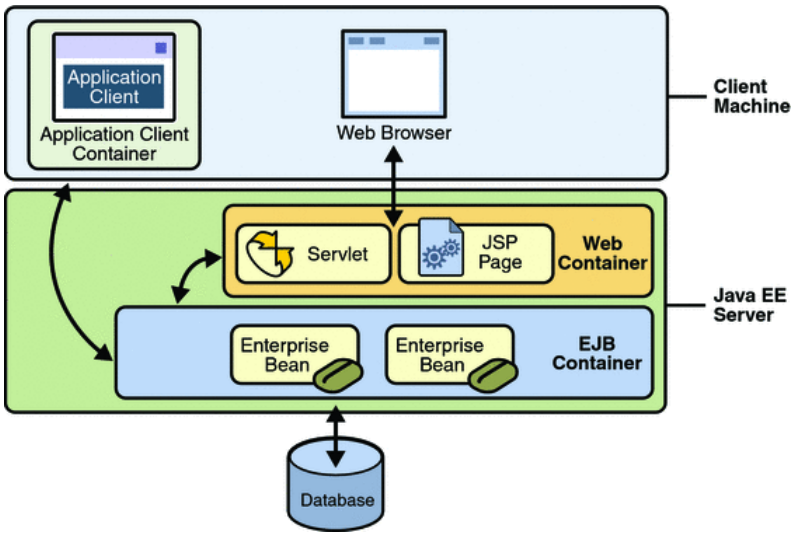
## 2.2 Enterprise Java Beans

Enterprise Java Beans (EJB) jsou řízené, serverové komponenty sloužících pro modulární vývoj podnikových aplikací a je součástí množiny aplikačního programového rozhraní (API) definující Java Enterprise Edition (Java EE). Na obrázku 1 je znázorněna hlavní myšlenka EJB, obrázek znázorňuje oddělení aplikační business logiky

od prezentační (JSF) a persistentní vrstvy (CRUD operace), ale také znázorňuje možnost spojení s ostatními technologiemi – JDBC, JNDI, JPA apod. EJB komponenty běží v kontejneru EJB jako například GlassFish nebo Tomcat. Tento kontejner poskytuje runtime prostředí pro webové komponenty, včetně zabezpečení, řízení životního cyklu servletů, zpracování transakcí a dalších webových služeb. Díky těmto vlastnostem je možné rychle vytvářet a nasazovat Enterprise Beans, které tvoří jádro transakčních aplikací. Použití technologie EJB je vhodné v případech, kde je zapotřebí reagovat na rostoucí počet uživatelů distribuováním aplikačních komponent na více zařízeních k zajištění transakční integrity, k zajištění komunikace s větším množství klientů. (Oracle, 2014)

Hlavní výhody EJB:

- přístup k systémovým službám jako jsou transakce, zabezpeční apod.
- není nutné řešit implementaci business pravidel nebo databázový přístup
- zaměření vývoje aplikace na prezenční vrstvu
- přenositelnsot EJB komponent s definovaným rozhraním
- efektivnější budování aplikací na již existující logice




Obrázek 1: EJB container system level

Zdroj: http://docs.oracle.com/javaee/5/tutorial/doc/figures/overview-serverAndContainers.gif

V technologii EJB existují dva základní typy java bean, které svůj typ označují pomocí anotací. Prvním z nich jsou Sessions Bean, které provádějí úkoly zadané klientem a mohou implementovat webové služby, druhým typem jsou Message-driven Bean jenž pracují jako posluchači pro určitý typ zpráv, na který reagují. (Oracle, 2014)

### Session Bean

Session Bean zajišťuje zapouzdření obchodní logiky aplikace, klient ji může vyvolat lokálně, vzdáleně nebo pomocí webových služeb. Tento typ beany není persistentní, z toho důvodu nejsou žádná její data ukládána do databáze. Session Bean se dále dělí na tři základní typy: Stateful, Stateless a Singleton.
Stateful beana se skládá z hodnot jeho proměnných a představují jedinečné spojení mezi klientem a beanou. Neprovádí sdílení mezi klienty, při ukončení spojení s klientem je ukončena a není dále přístupná. Používá se zejména na spojení s konkrétním klientem, uložení informace o klientovi a k řízení činnosti ostatních bean.
Stateless Session Bean představují opak Stateful, tudíž neuchovávají vnitřní stav. Jejich stav je uchován pouze po dobu jednoho spojení s klientem a po ukončení spojení je stav zapomenut. Použití Stateless Bean je například implementace webových služeb, kde nejsou potřeba konkrétní informace o klientovi například odeslání potvrzovacího e-mailu.
Posledním typem je Singleton Session Bean jenž implementuje návrhový vzor Jedináček (Singleton) a umožňuje použít podobných vlastností jako u Stateless Session Bean. Mohou udržovat svůj stav mezi klientskými voláními, ale při vypnutí serveru jsou znovu uvedeny do výchozího stavu. Využívají se v prípadech, kdy je potřeba udržet stav celé aplikace a v situacích, kdy k jedné Beaně přistupuje více vláken současně. (Oracle, 2014)

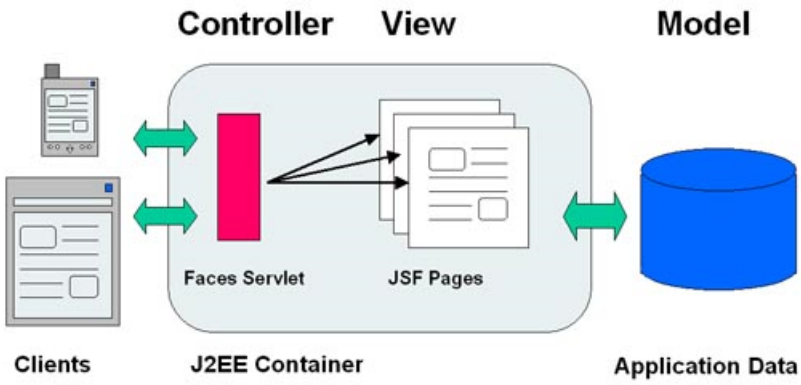
### Message-driven Bean

Message-Driven Bean umožňují aplikacím zpracování asynchronních zpráv. Zastávají funkci posluchačů Java Message Service (JMS) zpráv, které mohou být zaslány klientem, jinou EJB nebo jinou J2EE aplikací. Zasílání zpráv řídí webový kontejner za pomoci zdrojů serveru. Největším rozdílem oproti Session Beans je, že se klient nespojuje s Beans přes rozhraní ale přes JMS, nicméně si jsou podobné se Stateless Session Bean tím, že si nezachovávají stav. Message Driven Bean se spouštějí asynchronně po přijetí zprávy klienta, mohou aktualizovat data v databázi a provádět transakce. (Oracle, 2014)

## 2.3 JavaServer Faces

JavaServer Faces je framework, který se používá na straně serveru k oddělení definice uživatelského rozhraní od programování aplikační logiky v jazyce Java. Faces jsou soubory speciálních XML značek, kterým se specifikují odkazy na Java beans uložené v aplikačním serveru. Tato technologie vyvinutá společností Sun Microsystems, Inc. je součástí Java Enterprise Edition.

Hlavní myšlenkou je možnost využití MVC(Model View Controller) architektury při vývoji webových aplikací viz obrázek 2. Controller a šablonová část běží v J2EE kontejneru a je tím oddělena od datové části. Uživatel komunikuje s frameworkem prostřednictvím Faces Servletu, který poskytuje uživateli požadované JSF pages a překládá je do HTML kódu. Interface je definován pomocí speciálních XML tagů, kterým jsou předávána data k zobrazení nebo editaci ze standardních Java beanů. Díky tomuto frameworku je webová aplikace rozdělena čistě na uživatelské rozhraní a aplikační logiku. (Oracle, 2015)




Obrázek 2: JSF-Simple request flow

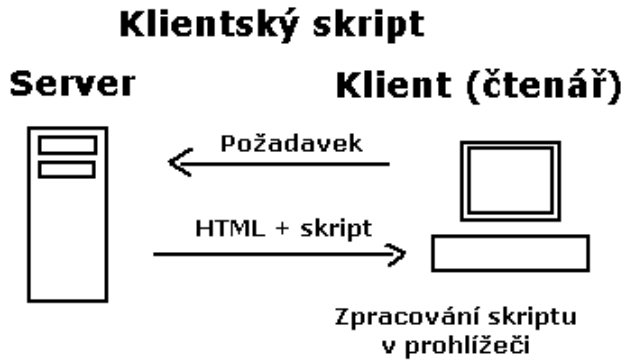
Zdroj: http://cdn.javabeat.net/wp-content/uploads/2014/03/JSF-Simple-Request-Flow.jpg

## 2.4 Prime Faces

PrimeFaces je open-source knihovna uživatelských rozhraní a komponent pro framework JavaServer Faces. Knihovna byla vytvořena firmou PrimeTek v roce 2009. PrimeFaces je velmi efektivním nástrojem pro tvorbu uživatelského rozhraní webové apliakce. Jeho hlavní výhodou je snadné použití prostřednictvím závislostí a částečné odstranění nutnosti psaní JavaScriptu. (PrimeFaces, 2015)

## 2.5 JavaScript

JavaScript je scriptovací jazyk pro vývoj dynamických webových stránek a webových aplikací. Jeho velkou výhodou je, že se zapisuje přímo do HTML kódu. Javascript je klientský script, což znamená, že jeho běh je zajišťování prostřednictvím uživatelova prohlížeče. To znamená, že se program odesílá se stránkou na klienta (do prohlížeče) a teprve tam je vykonáván viz obrázek 3. (Janovský D., 2007)




Obrázek 3: Princip JavaScriptu

Zdroj: http://www.jakpsatweb.cz/javascript/javascript-uvod.html

Syntaxe JavaScriptu je podobná jazyku Java a C++. Jedná se o vysokoúrovňový, dynamický, interpretovaný programovací jazyk a je vhodný pro objektově orientované, nebo funkcionální programování. Mezi jeho nevýhody patří zejména závislost na prohlížeči, kde může být zakázán uživatelem a tím znemožněn jeho běh. (Janovský D., 2007)

### JavsScript knihovny

Při tvorbě webových aplikací s využitím JavaScriptu se ve většině případů využívá práce s JavaScriptovými kihovnami a frameworky. Tento přístup vývojářům umožňuje snadnější a rychlejší vývoj aplikací, jelikož knihovny mají implementovány běžně používané nástroje pro programování v JavaScriptu. Nejvíce využívanou knihovnou je v dnešní době jQuery, jenž umožňuje snazší použití JavaScriptu na webových stránkách. Tato knihovna zastupuje většinu běžných úloh, které v JavaScriptu vyžadují spoustu řádků kódu navíc a zabalí je do jednoduchých metod. Použitím jQuery se vývojářům zjednodušuje DOM a CSS manipulace, metody událostí, efekty a animace, práce s techonolií AJAX a mnohé další výhody. Jedná se o velmi populární knihovnu, kterou využívají velké společnosti jako například Google nebo Microsoft. (W3Schools, 2015)

## 2.6 Asynchronní webové aplikace s technologií AJAX a AJAJ

AJAX (Asynchronous JavaScript and XML) je v oblasti webových aplikací označení pro technologie interaktivních prezentací. Díky ajaxu je možno dynamicky měnit obsah webových stránek bez nutnosti jejich kompletního znovunačítání za pomoci asynchronního zpracování webových stránek, které nám umožňuje JavaScript a jeho knihovny např. jQuery. Oproti klasickým webovýcm aplikacím poskytují ajaxové aplikace uživatelsky příjemnější prostředí, ale je nutné využití aktuálních verzí webových prohlížečů.

Prostřednictvím ajaxu proběhne odeslání zprávy uživatele na pozadí, server zašle jen ty části stránky, které se změnily a následně je uživateli na stránce aktualizují. Díky ajaxovému přístupu je možné snížit zátěž na webové servery a počítačové sítě. Jelikož není potřeba při každém požadavku sestavit celý HTML dokument, ale pouze provedené změny, je množství vyměňovaných dat výrazně nižší a teoreticky to může mít příznivý vliv i na zátěž databázových serverů či dalších backendových systémů. Nevýhodou ajaxu je naopak možné zvýšení počtu vyměňovaných HTTP požadavků. I když pomocí ajaxu přenášíme nižší množství dat, tak při nevhodné implementaci zátěž neklesne. (DARIE, C. a kol., 2006)

Data posílaná technologií AJAX jsou formátovaná pomocí značkovacího jazyka XML. Práce s jazykem XML a extrahování dat z této struktury je někdy komplikované, navíc obsahuje velké množství řídících dat. S řešením těchto problémů přišel Douglas Crockford, který představil datový formát používaný pro značení objektů a polí dat s názvem JavaScript Object Notation (JSON). Formát JSON je založen na syntaxi JavaScriptu viz následující ukázka, který nám umožňuje vytvářet datový formát schopný reprezentovat data podobně jako JavaScript, například kolekce dvojic název-hodnota, který reprezentuje objekt JavaScriptu obsahující pojmenované vlastnosti. (DARIE, C. a kol., 2006)

Ukázka JSON:

```
{
    id: 1548784,
    name: 'Jan',
    surname: 'Novak',
    gender: 'm',
    mogile: '123546789',
    address: {
        street: 'Olomoucak 54',
        city: 'Brno'
    }
}
```

## 2.7 Značkovací jazyk HTML

HTML (HyperText Markup Language) je standardní značkovací jazyk sloužící ke tvorbě vizuální části webových stránek. Základ jazyka HTML tvoří množina tagů (značek)
a jejich vlastností (atributů). Tento zápis byl převzán ze SGML a dále upravován v závislosti na aktuálních trendech Internetu. Mezi tagy jsou uzavřeny části dokumentu, tím se určuje sémantika obsaženého textu. Jednotlivé tagy jsou tvořeny úhlovými závorkami a popisem tagu. Značky jsou ve většině případů použití párové, přičemž koncová značka je shodná se značkou počáteční, jen má před názvem znak lomítko. Například označení odstavce: text v odstavci. Dále je v HTML možné využít i nepárové značky, například při zobrazení obrázku, nebo zobrazovat struktury vyskytujicích se v běžných dokumentech (formuláře, seznamy, odkazy, nadpisy apod.). HTML umožňují vložit vložit skripty napsané v jiných jazycích, jako je například JavaScript, díky čemuž je možné ovlivnit chování HTML stránky. (Písek, S., 2014)

### HTML5

HTML5 je poslední a aktuální verzí značkovacího jazyka HTML. Tato verze řeší především podporu multimediálních obsahů na internetu napříč všemi zařízeními (osobními počítači, tablety a chytrými telefony) a prohlížeči. V HTML5 byly zavedeny nové tagy pro rozvržení těla stránky (article, section, nav, footer, aside, header), tyto tagy nám přináší efektivnější pracování dat roboty. Dále byly přídány tagy pro práci s multimediálním obsahem pro audio a video, rozhraní pro určování geografické polohy uživatele, podpora formátu SVG v HTML dokumentu, podpora atributu canvas, který umožňuje vykreslovat 2D grafiku. (HOGAN, B. P., 2013)

## 2.8 CSS

CSS (Cascading Style Sheets) je jazyk pro popis způsobu zobrazení elementů na stránkách napsaných v jazycích HTML, XHTML nebo XML. Jazyk CSS byl navržen organizací W3C. Hlavním důvodem vzniku kaskádových stylů bylo umožnit vývojářům oddělit vzhled dokumentu od jeho struktury a obsahu. Ve starších verzích HTML se vyskytovaly elementy, které nijak nepopisují obsah a strukturu dokumentu, ale i způsob jak jsou elementy zobrazeny. Tento přístup je při vyhledávání informací v dokumentu nežádoucí. (CSS, 2015) Syntaxe kaskádových stylů je velice jednoduchá a skládá se z několika pravidel. Pravidla jsou složena ze selektoru a bloku deklarací. V každém bloku deklarací se nacházejí deklarace oddělené středníky. Jednotlivé deklarace jsou složeny z identifikátoru vlastnosti, za identifikátorem následuje dvojtečka a hodnota vlastnosti viz ukázka CSS. (CSS, 2015)

Ukázka CSS:

```
footer {
    padding-top: 10px;
    background: #232323;
    min-height: 60px;
    text-align: center;
    border: solid #666 2px,
    background-color: rgba(0,0,0,0.5)
}
```

### CSS3

CSS3 je poslední verzí jazyka CSS. Podobně jako HTML5 rozšiřuje původní CSS o nové funkce a vlastnosti. Většina funkcí CSS3 je v aktuálních prohlížečích bez problémů podporována, problémy s podporou nastávají při zavedení nových vlastností, které starší verze prohlížečů mohou mít problém zpracovat. (HOGAN, B. P., 2013) Některé výhody CSS3:

- zaoblení rohů u HTML prvků (border-radius)
- stín u blokového prvku(box-shadow)
- transformace prvků(rotate,translate,scale)
- animace
- průhledná pozadí
- přechody

## 2.9 Databázový systém

Databázový systém představuje spojení databáze a systému řízení báze dat (SŘBD). Samotná databáze je množina homogenních souborů s pevně definovanou strukturou, tyto soubory jsou vzájemně propojeny pomocí klíčů. Z hlediska způsobu práce s daty se databáze můžeme databáze rozdělit do několika datových modelů. (POKORNÝ, J. - HALAŠKA, I., 2003)

- Síťový model
- Hiearchický model
- Relační model
- Objekotvý model
- Not only SQL (NoSQL) model

Pro řešení bakalářské práce je využit relační databázový model. Provádění operací s daty zajišťuje systém řízení báze dat, jedná se o software umožňujicí vkládat, vybírat, modifikovat a odstraňovat jednotlivé záznamy v databázi. Mezi jeho další funkce patří například správa klíčů, autentizace uživatelů, správa transakcí, spouštění triggerů a procedur nebo zajištění integrity dat. Existuje několik typů SŘBD, nicméně v relačních databázích jsou nejvíce využívány systémy MySQL, PostgreSQL a Oracle.

### MySQL

Databáze MySQL vyvíjená firmou Oracle, je díky snadné instalaci, dobrému výkonu a open-source licenci hojně využívanou databází. Mezi její výhody patří zejména rychlá odezva, jednoduchá správa, dostupnost pro různé operační systémy či využití pro aplikace vyvíjené ve více programovacích jazycích například Java, C++, PHP apod. Jeho nevýhodou je omezenější podpora programátorských konstrukcí a nižší výkon při použití v robustních webových aplikacích. I přes tyto nedostatky je možné MySQL využít v okruhu projektů velkých firem, ale i obyčejných uživatelů. (POKORNÝ, J. - HALAŠKA, I., 2003)

## 2.10 Testování webových aplikací

Testování je nezbytnou fází vývoje webových aplikací. Díky testování je možné vyhledat a odladit množství chyb, které nebyly při vývoji zjištěny. V praxi existuje velké množství testovacích metod, nicméně se jedná o časově velmi náročný proces, proto je dobré tuto část vývoje softwaru automatizovat.

Pojem automatizace vyjadřuje využití samočinných řídicích systémů k řízení (ovládání) zařízení a procesů. Na rozdíl od mechanizace, která ulehčuje lidem práci na mechanizovaném zařízení automatizace snižuje nutnost zapojení člověka při vykonávání výrobních určitých procesů. (Automatizace, 2016)

### Selenium

Selenium je sada automatizovaných testovacích nástrojů s licencí open-source a je určen k testování výhradně webových aplikací. Je uzpůsoben pro běh v různých prohlížečích, operačních systémech a velkém množství dnes využívaných programovacích jazyků. Selenium se skládá ze čtyř samostatně fungujicích částí. (Selenium, 2016)

- **Selenium IDE:** Selenium Integrated Development Environment (IDE) je plugin určený pro webový přohlížeč Firefox. Poskytuje grafické uživatelské rozhraní pro ukládání záznamů uživatelských akcí (skriptů). Tyto skripty mohou být převedeny do různých programovacích jazyků, jenž jsou Seleniem podporovány a provedeny na jiných prohlížečích. (Selenium, 2016)
- **Selenium RC:** Selenium Remote Control (RC) je v dnešní době téměř nevyužívaný. Jeho principem je komunikace klientských knihoven se Selenium RC serverem, kde se zpracovávají jenotlivé příkazy, které jsou následně předány do prohlížeče. (Selenium, 2016)
- **Selenium Grid:** Tento nástroj se využívá pro běh paralelních testů napříč rozdílnými zařízeními a prohlížeči současně, což testování přináší zkrácenou dobu spuštění. (Selenium, 2016)
- **Selenium WebDriver:** Jedná se o nejvyužívanější nástroj Seleniových testů, který narozdíl od Selenium RC spolupracuje přímo s prohlížečem bez jakéhokoliv prostředníka. Používá se například pro testování aplikací ve více webových prohlížečích, komplexní navigaci webu či testování ajaxově založených prvků stránky. (Selenium, 2016)

Přes všechny výhody, které automatizovaného testování přináší, je dobré zmínit i jeho nevýhody, mezi ně nůžeme zahrnout nereálná očekávání od vývojářů, nutnost mít dobrou testovací praxi a náročná údržba a aktualizace automatizovaných testovacích skriptů, jenž je nutné měnit při jakékoliv změně aplikace. (HLAVA, T., 2015)

## 2.11 Geolokace

Na webových aplikacích každým rokem narůstá počet uživatelů využívajicích mobilní zařízení, díky tomu vzniká množství webových i mobilních aplikací, které zajišťují funkci geolokace. V této oblasti existuje celá řada technik a datových senzorů, které lze využít pri zjištění geografické polohy uživatele s různou úrovní přesnosti. Nejběžnější jsou stručně popsány.

- **GPS:** Jedná se o nejrozšířenější způsob určování zeměpisné polohy prostřednictvím GPS (Global Positionig System) příjmače, jenž má většina moderních zařízení standartdně zabudvaný. Tento původně vojenský systém funguje na principu detekce signálu z několika satelitů umístěných na oběžné dráze země, typycky jsou pro lokalizaci uživatele zapotřebí čtyři a více satelity. Systém využívá matematickou techniku trilaterace, díky této metodě může zařízení určit svoji polohu v závislosti na časování satelitních signálů. Nevýhodou GPS je signál náchylný na rušení a stínění, nelze jej využít například v budovách. (Global Positioning System, 2016)
- **Assisted-GPS(A-GPS):** Systém A-GPS pomohá řešit některé nedostatky GPS tím, že rozšiřuje jeho možnosti prostřednictvím datových sítí od poskytovatelů mobilních a internetových služeb, které pomáhají lokalizovat uživatelské zařízení. A-GPS pracuje ve dvou módech, prvním z nich je Mobile Station Assisted (MSA) kdy zařízení příjmá akviziční pomoc, referenční čas a dalších asistenčních data od poskytovatele mobilních služeb. Poskytovatel průběžně zaznamenává informace z GPS satelitů za použití A-GPS serveru ve svém systému. S pomocí výše uvedených údajů server vypočítává polohu a pošle ji zpět k uživateli. Mobile Station Based (MSB) mód pracuje na podobném principu, ale výpočet polohy provádí uživatelské zařízení. (Assisted GPS, 2016)
- **Wi-Fi positioning system:** Tato metoda nachází využití v místech, kde satelitní navigační systémy nedokážou poskytovat signál v důsledku různých příčin. WPS se využívá k lokalizaci uvnitř objektů. Základním konceptem této metody je měření intenzity přijatého signálu. (Wi-Fi positioning system, 2016)
- **Cell Triangulation:** Zjišťování polohy mobilního telefonu, ať už stacionární nebo pohyblivý může být proveden pomocí rádiových signálů mezi několika vysílači mobilního signálu. Pro použití této metody musí zařízení umožňovat roaming. Princip fungování je založen na síle signálu telefonu do okolních anténní stožáry. (Mobile phone tracking, 2016)

## 2.12 IBM bluemix

IBM Bluemix je open-standards cloudová platforma pro budování, provoz a správu aplikací. S Bluemix, se mohou vývojáři soustředit na budování aplikací s flexibilními výpočetními možnostmi, širokým výběrem vývojářských nástrojů, výkonnou sadou IBM API a služeb třetích stran. Bluemix poskytuje mobilní a webově vývojářský přístup k softwaru IBM, který zajišťuje integraci, bezpečnost, transakce a jiné klíčové funkce. (Bluemix, 2015)

Bluemix svým uživatelům poskytuje následujicí funkce:

- množství služeb, které umožní uživateli rychle vytvářet a rozšiřovat webové a mobilní aplikace
- univerzální programovací modely a služby
- ovladatelnost služeb a aplikací
- nepřetržitou dostupnost

V cloudovém řešení Bluemix je umožněno rychle vyvíjet aplikace v populárních a světově nejužívanějších programovacích jazycích. V nabídce je možné vytvořit mobilní aplikace v iOS a Android. Pro webové aplikace, je možné využít jazyky JavaScript, Ruby, PHP, Java, Python nebo NodeJS. Dále je možné přenést existující aplikaci na Bluemix a plně využít rozhraní, které poskytuje pro běh aplikace. (Bluemix, 2015)

## 2.13 Autonomní robot BuggyMan

Pro vývoj řídícího softwaru byl zvolen programovací jazyk Java. BuggyMan má implementován vlastní framework umožňující snadné rozšíření nezávislých modulů, které obsluhují jednotlivé hardwarové části robota. Framework zajišťuje také mezimodulovou komunikaci. Výhodou tohoto prostředí jsou technologie, které umožňují běh modulů ve vlastních vláknech, snadnou zaměnitelnost a vytvoření testovacích tříd. (ONDROUŠEK, V., 2015)

Obrázek 4: BuggyMan

Zdroj: https://aistorm.mendelu.cz/cz/projekty/buggyman

## 2.14 Použité vývojové nástroje

Před zahájením vývoje je nutné zvolit vhodné vývojové prostředí pro přípravu aplikace. Vývojář má teorticky volné rozhodování při výběru vývojového prostředí, nicméně ne všechna prostředí jsou vhodná pro vývoj java aplikací, jelikož nemusí podporovat například našeptávání pro vybraný programovací jazyk, nebo ani neobsahují příslušný překladač. Proto je nutné před zahájením vývoje, dobře zvážit výběr vývojového prostředí.

### NetBeans

Pro vývoj jednotlivých java bean, javascriptu, xhtml šablon a css stylů zvoleno vývojové prostředí NetBeans. Jedná se o bezplatně šířitelný produkt pod OpenSource licencí a není nijak omezené jeho využití. Samotné vývojové prostředí je vytvořeno v jazyce Java, přesto je v něm možné vyvíjet prakticky v jakémkoliv programovacím jazyce např. C++ nebo PHP. Pro NetBeans existuje množství modulů, jenž toto vývojové prostředí rozšiřují. Jeho další výhodou je buildovací nástroj ApacheMaven a aplikační servery, které jsou součástí NetBeans a výrazně usnadňují vývoj aplikace.

### Apache Maven

Maven je automatický nástroj pro sestavení projektu, primárně určen pro Java projekty. Jeho největším přínosem pro vývojáře je automatické stahování potřebných knihoven a pluginů z veřejného repozitáře, které jsou nezbytné pro chod aplikace. Po stažení všech závislostí provede Maven sestavení projektu, pro sestavení využívá konfigurační soubor pom.xml, jenž bývá umístěn v kořenovém adresáři aplikace.

### Glassfish server

Glassfish je aplikační server určený pro platformu JEE. Tento server není primárně určen pro provoz aplikací, ale slouží převážně jako ukázka nových rysů specifikace platformy JEE, nicméně pro vývoj aplikace je zcela dostačující.

# 3 Metodika řešení

V této kapitole jsou stručně popsány požadavky na aplikaci a jednotlivé fáze vývoje webové aplikace pro interakci s mobilním robotem.

## 3.1 Požadavky aplikace

Výstupem bakalářské práce je jednoduchá a uživatelsky přívětivá webová aplikace, disponující několika funkcemi pro předávání zpráv mobilnímu robotu. Mezi požadavky můžeme zahrnout rezervování robota uživatelem na dobu potřebnou pro průvod, které je řešeno pomocí uživatelova session ID a jeho následné uvolnění v případě uživatelovi delší nečinnosti, dále implementaci realtimového komunikačního api (WebSockets), přes který bude robot komunikovat se servrem, návrh a implementaci .xhtml šabolny, asynchronní zobrazování pozice na mapě a asynchronně měnící se ovládací menu aplikace. Na závěr vytvoření WebSocket koncového uživatele, jenž bude simulovat funkci robota.

## 3.2 Vývoj aplikace

V první fázi vývoje je nutné navrhnout proces průvodu uživatele arboretem od jeho příchodu do arboreta až po ukončení celého procesu průvodu. Po vytvoření návrhu bude navržen vzhled a struktura stránek pomocí jednoduchého drátěného modelu. Jakmile bude návrh hotov, vytvoříme adresářovou strukuru aplikace pomocí nástroje Apache Maven a provedeme konfiguraci souborů pom.xml a web.xml. Další fáze vývoje je zaměřena na vytvoření šablony a jednoduchého responzivního vzhledu. Současně s přípravou šablon provedeme implementaci JavaScript funkcí pro práci s Google maps, dynamicky se měnícího menu a dalších pomocných funkcí. Po vytvoření šablon budou implementovány jednotlivé Java beany a komunikační WebSocket api, dále bude implementován xml soubor, kde bude zaznamenán seznam cílových míst. V poslední fázi vývoje bude otestována funkčnost aplikace, pro otestování bude vytvořena samostatná aplikace nahrazující fyzického robota.

# 4 Praktická část

V praktické časti jsou podrobněji zdokumentovány implementační postupy při vývoji aplikace. V kapitole je popsán návrh grafického rozhraní, průběh průvodu, konfigurace prostředí a implementace dílčích částí od šablon až po testování aplikace.

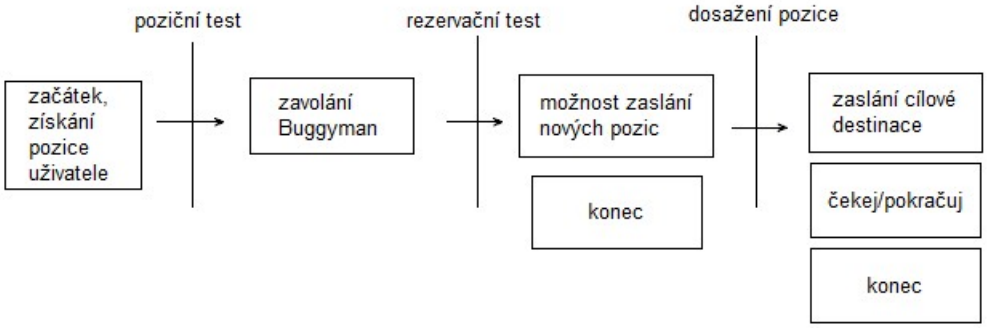
## 4.1 Grafický návrh aplikace

Vyvíjená webová aplikace nemá žádné velké požadavky na grafické řešení. Pro její správné fungování stačí pouze jeden navigační panel s ovládacími tlačítky a informační banner, který informuje uživatele o stavu aplikace. Obsahem a nejdůležitější částí apliakce je interaktivní mapa, v mapě se zobrazují pozice uživatele a robota. Dále jsou v mapě zobrazeny zajímavé destinace, ke kterým se může uživatel nechat zavést. Jako mapový podklad je použita Gmap ortofoto mapa, jednotlivé objekty v mapě jsou zobrazeny pomocí icon, získaných z aplikace Iconfinder, jedná se databázi placených ale i free icon pro volné využití. Pro grafický návrh stránky je vytvořen jednoduchý wireframe neboli drátěný model, jedná se o jednoduchý nákres rozložení jednotlivých prvků na webové stránce. Vytváří se zpravidla na záčátku vývoje webové aplikace, pro ujasnění informačního a navigačního návrhu. Musí obsahovat všechny důležité informační prvky stránky, jako je například záhlaví či navigace. Při jeho vytváření nepoužíváme žádné složité grafické prvky, pouze jednoduché tvary s popisem. Pro návrh drátěného modelu slouží grafické editory, ale je možné jej nakreslit i ručně. Vyvíjená webová aplikace se skládá ze dvou jednoduchých stránek. První stránka je čistě informační a skládá se pouze z popisového pole aplikace a inputu pro zadání jména návštěvníka. Druhá stránka se skládá z navigačního panelu, ve kterém jsou umístěny ovládací tláčítka, informačního panelu a mapového podkladu vyplňujicí celý obsah stránky viz obrázek 5. Grafický návrh se jeví jako značně jednoduchý, nicméně je zapotřebí brát v úvahu využívání aplikace na mobilních zařízeních, z tohoto důvodu je zvoleno tak jednoduché řešení, které lze efektivně zobrazit i na mobilním zařízení.


Obrázek 5: Drátěný model

## 4.2 Průběh průvodu

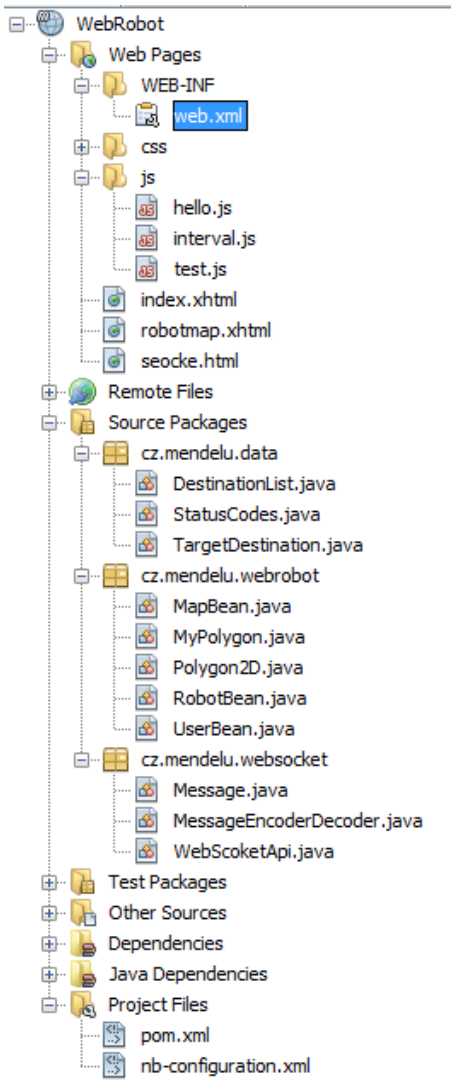
Průvod arboretem, a s ním spojedné částečné ovládání robota, je prováděno pomocí sady definovaných funkcí. Při vstupu uživatele do aplikace, je nejprve nutné určit jeho geografickou pozici. To je vyřešeno pomocí funkce Begin guide, při provolání této funkce proběhne zjištění uživatelovi pozice a ověření zda se nachází v areále arboreta, z aplikačně bezpečnostních důvodů je nutné toto ověření provést na backendu aplikace. Pokud je výsledek ověření uživatelovi pozice kladný, aplikace automaticky zpřístupní další krok, tím je zavolání BuggyMana. Při volání této funkce se na backendu aplikace ověřuje zda již robot není využíván jiným uživatelem. Pokud tomu tak je, provede se vypsání hlášky, že BuggyMan je v současné době rezervovaný jiným uživatelem. V opačném případě se provode zarezervování BuggyMana, který se automaticky vydá na přesun k uživatelovi. V průběhu přesunu je uživateli umožněno poslat BuggyManovi svoji novou polohu pro případ, že pozici změnil, dále je umožněno zavolat ukončení celého procesu a vrácení do stavu uživatele a i BuggyMana do počátečního stavu. Při dosažení uživatelovi pozice se uživateli zpřístupní zaslání souřadnic cílovho místa, kterou si během čekání na Buggymana vybere. Dále může uživatel libovolně měnit cílovou destinaci, zavolání funkce wait/continue, při které robot zahájí čekání/pokračování v jízdě. Jakmile uživatel společně s Buggymanem dosáhnou cílové destinace, tak je uživatel dotázán, zda chce zvolit další destinaci nebo ukončit průvod.




Obrázek 6: Diagram průběhu průvodu

## 4.3 Adresářová struktura aplikace

Adresářová struktura viz obrázek 7, vytvořená z Maven Web Aplication projectové šablony, kterou standardně Netbeans poskytují. Struktura aplikace je složená z několika adresářů. Základním adresářem je Source Packages, v něm se nacházejí zdrojové kódy backendové části aplikace a jsou rozděleny do tří balíčků. V datovém balíčku je řešen přístup k datům o jednotlivých destinacích, webrobot obsahuje jádro celé aplikace a websocket komunikační api. Druhým důležitým adresářem je Web Pages, který obsahuje xhtml šabolny stránek, javascript, kaskádové styly a adresář WEB-INF, kde je umístěn konfigurační soubor web.xml. Dále jsou tu adresáře Test Packages, kde jsou umístěny testy, dependencies se staženým závislotmi a adresáře Project Files, zde se nachází nejdůležitější konfigrační soubor aplikace pom.xml.




Obrázek 7: Adresářová struktura

## 4.4 Konfigurační soubory

### Pom xml

Jedním ze základních konfiguračních souborů pro vývoj aplikací na JSF frameworku je soubor pom.xml (Project Object Model). Tento soubor popisuje aplikaci nejen z pohledu zdrojového kódu, ale i co se týče závislostí na externích knihovnách, popis buildovacího procesu a funkcí s ním spojených, jako je například spouštění testů. Dále umožňuje definovat konstanty, které mohou využít apliakční pluginy. Pro popis aplikace slouží definovaná xml struktura, která se nachází v adresáři Project Files. V případě, kdy je projekt složen z více modulů je pom.xml umístěn v každém jednotlivém modulu a dědí vlastnosti z nadřazeného souboru. S pomocí takto definované struktury je možné projekt sestavit použitím jediného příkazu. (Apache Maven, 2016)

Konfigurační soubor pom.xml použitý ve vyvíjené aplikaci je složen z hlavičky, kde je určena verze xml, kódování a xml namespace, jenž jsou předem definované. Důležitou součástí jsou identifikátory projektu viz. Obrázek 8, které se skládají z groupId, artifactId, verze a typu balíčku. Dále je nutné definovat jednotlivé závislosti(dependencies), které budou staženy při sestavení projektu. Jednotlivé dependency se skládají z podobných atributů jako identifikátory projektu (groupId,artifactId,version). Nicméně mohou obsahovat i další atributy jako například scope.

V aplikaci je použito celkem pět dependecies. Prvním z nich je commos-lang. Commos-lang nabízí celou řadu pomocných utilit pro java.lang API, jsou to zejména metody pro manipulaci s řetězci, základní numerické metody, serializace apod. Dále obsahuje základní vylepšení java.util.Date a další řadu utilit, jako je například hashCode či equals. Javaee-web-api nám poskytuje například množství EJB anotací, utility pro práci s JSON formátem, validace, servlety a websockets, které jsou nedílnou součástí aplikace. Balík com.sun.faces poskytuje aplikaci MVC framework JSF, stejně tak nám poskytuje balík org.primefaces stejně nazvaný framework. Balík org.seleniumuhq je potřebný pro automatizované testování a org.jdom je využit pro parsování xml souborů.

### Web xml

Deployment Descriptor file (Web xml) popisuje nasazení servlet containeru jako například Glassfish nebo Tomcat. Tento xml dokument definuje potřebné informace, které používá servlet container při spouštění aplikace, jsou to například contextové parametry, nastavení servletu, session nebo uvítací stránky.

## 4.5 Prezentační vrstva

### Šablony xhml

První šablonou, ze které se návštěvníkovi vygeneruje úvodní stránka je index.xhtml. Šablona se skládá pouze z nadpisu a jednoduchého formuláře pro zadání jména. Formulář se zobrazuje pouze tehdy, je-li navázáno spojení mezi Buggymanem a aplikací, toho je docíleno pomocí tagu, který je obdobou funkce swicth. Samotné testování je provedeno pomocí tagu viz ukázka. Zde se z beany s názvem robotBean získává atribut runable, pokud je tento atribut true, dojde ke zpřístupnění aplikace.

Ukázka funkce if:

```
<c:when test="#{robotBean.runAble==true}">
    <h:form>
        <p:outputLabel value="Enter your name:" for="name"/>
        <p:inputText id="name" value="#{userBean.name}"/>
        <p:commandButton value="Enter" action="robotmap?faces-redirect=true"/>
    </h:form>
</c:when>
</c:when>
    <h2>The Buggyman is not available</h2>
</c:otherwise>
```

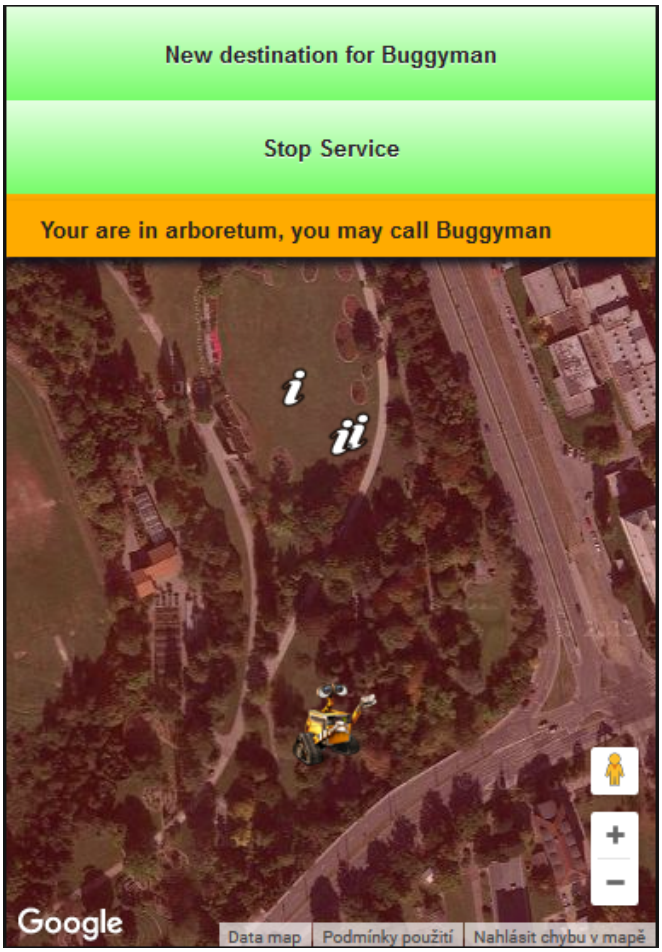
Druhou šablounou je robotmap.xhtm, její nedílnou součástí jsou 2 formuláře, jenž obsahují ovládací tlačítka aplikace <:commandButton>. Tato tlačítka je nutné umístit do formuláře, jelikož to vyžaduje syntaxe Primefaces. První formulář je součástí navigačního panelu a jsou v něm umístěny tlačítka pro uživatelské ovládání aplikace. Po spuštění aplikace je většina tlačítek skryta pomocí css vlastnosti "display:none", jak uživatel postupně prochází aplikací, tak jsou talčítka postupně zobrazována pomocí vlastnosti "display:inline-block".

Ukázka tlačítka:

```
<p:commandButton style="display: none" id="stopService"
            value="Stop Service" class="myButton"
            action="#{mapBean.stopService()}" >
     <f:param name="available" value="false"/>
</p:commandButton>
```

Tlačítka commandButton obsahují parametr available defaultně nastaveného na hodnotu false, pokud je tlačítko skryto. Při zobrazení tlačtíka se parametr nastaví na true, tento parametr se kontroluje uvnitř metod v backend beanách, které jsou pomocí atributu action volány, je-li nastaven na false, tak se funkce v těle metody neprovedou.

Druhý formulař není uživateli zobrazen, slouží jako ”prostředník” pro volání metod v Java beanách, jenž je zapotřebí volat automaticky z JavaScriptu. Metody se volají z javascriptu z důvodu geolokace, které je řešeno prostřednictvím GoogleMap api fungujicího pouze v JavaScriptu. Skrytá tlačítka obsahují dva paramtery, zeměpisnou šířku a zeměpisnou délku nastavenou JavaScriptem. Poslední částí šablony je panel zobrazující zprávy. Je řešen pomocí div bloku s jedním paragraphem ”<p>”, ten se updatuje automaticky prostřednictvím JavaScriptu.




Obrázek 8: Šablona robotmap

## 4.6 JavaScrpit

### GoogleMaps JavaScript API

Tento interface je nezbytnout součástí aplikace, zajišťuje získání uživatelské polohy, inicializaci mapy a pozic všech prvků na mapě. Pro její používání je nutné vygenerovat autentifikační klíč, který se zadává do zdrojového URL jako parametr key, viz ukázka.

```
<script src=”https://maps.googleapis.com/maps/api/js?key=
    AIzaSyDLV0F6ae9YZ90&amp;callback=initMap”
    type=”text/javascript” defer=”true” ></script>
```

První funkcí pracující s mapou je funkce initMap, jenž slouží k inicializaci mapového podkladu. Nastavují se atributy mapy, jako jsou zoom sloužící k přiblížení, center pro zobrazení vybrané části mapy a mapTypeId pro typ mapového podkladu. API je dále využito pro přidávání a odebírání objektů v mapě. Ty se vytváří pomocí tzv. značek viz ukázka kódu:

```
robotMarker = new google.maps.Marker({
            icon: 'robot.png',
            position: robotPosition,
            map: map
});
```

U značky můžeme nastavit různé atributy, v našem případě nastavujeme iconu objektu, která může být defaultní od Googlu, nebo je možné použít vlastní obrázek, dále GPS souřadnice objektu a instanci mapy, kde se má objetk zobrazit. Pozice objektů je nutné libovolně během běhu aplikace měnit, to je umožněno pomocí funkce setPosition s parametrem position, používající se přímo na vytvořené instanci. Pro odstranění objektu z mapy slouží funkce setMap s parametrem null viz ukázka:

```
destinationMark.setMap(null);
robotMarker.setPosition(robotPosition);
```

Důležitou funkcí potřebnou pro průběh aplikace je locate(), v ní se získávají pozice uživatele pomocí HTML Geolocation API. Uživatelské souřadnice se získávají prostřednictvím instance navigator, na ní je umožněno zavolat metody geolocation a getCurrentPosition.

```
navigator.geolocation.getCurrentPosition(showPosition);
```

Parametr showPosition je název instance, ve které je možné přistoupit k GPS souřadnicím 'showPosition.coords.latitude'.

### Pomocné funkce

Pro dynamický průběh aplikace je nutné měnit jednotlivá ovládací tlačítka a informační panel. Pro provádění změn v informačním panelu slouží jednoduchá funcke updateMessage(ms) s parametrem 'ms'. Tuto funkci je možné zavolat ze kterékoliv Managed Beany v níž jsou funckci předávány jednotlivé zprávy. Při zavolání funkce dojde ke změně HTML obsahu odstavce pomocí jQuery metody .html a pro vizuální efekt je použita metoda .animate v níž se mění průhlednost informačního panelu, tím dojde k jeho problikání.

```
function updateMessage(ms) {
    $("#message").html(ms);
    $("#messageBox").animate({opacity:'0.2'},"slow");
    $("#messageBox").animate({opacity:'1.0'},"slow");
}
```

Změna jednotlivých tlačítek je řešena několika funkcemi, kde se pomocí document object modelu získají přes ID jednotlivá tlačítka a vytvoří se jejich instance. Po vytvoření instancí je možné měnit vlastnosti jednotlivých tlačítek. Nejdůležitější změnou je CSS vlastnost "display", která se mění pomocí stejně jmenované jQuery metody. U této vlastnosti se nastavují pouze dvě hodnoty, "none" pro skrytí a "inline-block" pro zobrazení. Před zobrazením tlačítka je ještě nutné nastavit parametr tlačítka "available" na hodnotu "true".

```
var callbuggyman = $("#navForm:callbuggyman");
callbuggyman.onclick = function () {
    PrimeFaces.ab({s: "navForm:callbuggyman", pa: [{name: "available",
    value: "true"}]});
    return false;
    };
$(callbuggyman).css("display", "inline-block");
```

Poslední JavaScript funkcí je interval, díky němuž je možné automaticky spouštět metody v Managed Beanách. Samotné volání metod je řešeno pomocí skrytého tlačítka, to se aktivuje ve funkci interval s využitím jQuery funkce "click()". Ve funkci se dále provádí volání funkce JavaScritp funkce "locate()", to aplikaci zajistí automatické získávání informací o uživatelově pozici. Spuštění funkce se provede ihned po načtení stránky, to je zajištěno pomocí metody "ready".

```
var button = $("#hiddenForm:intervalCall");
setInterval(function () {
    locate();        button.click();
}, 5000);
```

## 4.7 Backendová vrstva

Backendová vrstva je rozdělena do několika balíčků. Jádrem celé apliakce je balíček webrobot, v něm jsou umístěny nejdůležitější Managed beany.

### RobotBean

RobotBean udržuje důležité informace o BuggyManovi na straně serveru a je v ní uchováno několik atributů. Hlavními atributy jsou ”latitude” a ”longitude”, ty uchovávají informace o GPS souřadnicích BuggyMana, dalšími atributy ”destLatitude” a ”destLongitude” uchovávající informace o cílové pozici, na kterou se musí robot přesunout. Dále jsou tu uchovány ”userSessionId” pro uložení Id uživatele, který má aktuálně přístup k ovládání, ”runable” zaznamenává fyzický stav robota zda je provozuschopný, atribut ”book” testuje je-li robot v danou chvíli zarezervován a ”statusCode” uchovává aktuální statusovou zprávu od BuggyMana.

Pro správnou funkčnost je nutné zajistit, aby vznikla pouze jedna instance RobotBeany v rámci celé aplikace, tzv. Singleton. Toho je dosaženo pomocí anotace @Singleton, díky této anotaci tak vznikne pouze jedna instance v rámci celé aplikace. Další důležitou anotací je @AppliactionScoped, ta zajistí udržení beany v běhu po celou dobu provozu aplikace. Beany označeny anotací @ApplicationScoped vznikají při zavolání prvního Http requestu a zanikají při ukončení aplikace. V neposlední řadě se musí beany, ke kterým je přistupováno ze šablony označit anotací @ManagedBean.

Vzniklá instance obsahuje kromě getterů a setterů několik důležitých metod využívaných pro průvod arboretem. Nejdůležitější metoda je callBuggyman, ta je volána při stisku tlačítka ”Zvalot Buggymana”, při provádění této metody je nutné otestovat zda již není BuggyMan využíván jiným uživatelem. Tento problém je řešen pomocí atributu book, který při hodnotě false zpřístupní uživateli průvod. Při zpřístupnění se nastavují atributy destLatitude, destLongitude, a userSessionId, tyto tři parametry jsou předány v hlavičce aplikace. UserSessionId se eviduje z důvodu zabezpečení aplikace, při zavolání jiných metod dojde k ověření userSessionId. Pokud se userSessionId neshoduje, tak se provede vypsání chybové hlášky při odeslání request contextu zpět k uživateli. Request context je využíván pro spouštění funkcí v javaScriptu viz ukázka:

```
RequestContext rc = RequestContext.getCurrentInstance();
String s = ”showOtherService()”;
rc.execute(s);
```

### MapBean

Jedná o managedBeanu označenou anotací @SessionScoped tzn., že její instance vznikne v době, kdy k ní uživatel přistoupí a skončí při opuštění aplikace (ukončení session). V MapBeaně je nutné zajistit přístup k metodám v RobotBean, to je umožněno pomocí dependency injection, atribut RobotBean je označen anotací @ManagedProperty, tato anotace zajístí injekci instance RobotBean do třídy MapBean. Z důvodu existence pouze jedné instance RobotBean, musí být tento atribut statický.

Balíčky primefaces v aplikaci umožňují poměrně snadné vytváření mapových objektů na straně serveru. Toho je využito například při vytváření polygonu arboreta, ten se vytváří v metodě init(). Pro vytvoření polygonu je nutné zadat jednotlivé vrcholy pomocí GPS souřadnic. V následující ukázce je uveden příklad vytvoření polygonu arboreta, který se předává pomocí request scope do javaScriptu.

```
polygonModel = new DefaultMapModel();
LatLng coord1 = new LatLng(49.2113831, 16.6128231);
LatLng coord2 = new LatLng(49.2109272, 16.6131664);
...
LatLng coordN = new LatLng(49.2123922, 16.6158994);

Polygon polygon = new Polygon();
polygon.getPaths().add(coord1);
polygon.getPaths().add(coord2);
...
polygon.getPaths().add(coordN);
polygonModel.addOverlay(polygon);
```

V této beaně je nutné implementovat ověření pozice uživatele, zda se nachází uvnitř polygonu. Tento problém je řešen v metodě testUserPosition(), která přebírá parametry latitude a longitude. K ověření pozice uživatele dojde při každém odeslání nových souřadnic. Jelikož PrimFaces neumožňuje otestovat, zda se bod o souřadnicích x,y nachází v daném polygonu, bylo nutné implementovat vlastní třídu Polygon2D, která tuto funkci umožňuje.

Ostatní metody této beany slouží pro přístupu k metodám v RobotBeaně z důvodu jednoduššího snadného získání userSessionId, jenž je v této beaně uchováno jako atribut.

### UserBean

Stejně jako MapBean je tato beana označená anotací @SessionBean. Pomocí dependency injection je umožněno přistoupit k MapBean instanci, která se vytvoří současně s UserBean. Dále tato beana uchovává jméno uživatele, sessionId a informace o jeho pozici. Mimo konstruktor obsahuje beana pouze metodu locate(), ta neobsahuje žádné parametry v hlavičce metody, ale pouze přebírá pomocí instance FacesContext a nastavuje uživatelovu pozici v mapBean, kde se testuje.

Získání parametrů z FacesContext:

```
FacesContext fc = FacesContext.getCurrentInstance();
Map<String, String> params = fc.getExternalContext().getRequestParameterMap();
setLatitude(Double.parseDouble(params.get("lat")));
setLongitude(Double.parseDouble(params.get("lon")));
```

### WebSocketApi

WebSocketApi je využíváno pro komunikaci s fyzickým zařízením Buggymana. Jedná se o full-duplex komunikaci mezi klientem (BuggyMan) a serverem. Tato komunikace je udržována po celou dobu provozu obou zařízení. Na straně serveru je implementována prostřednictvím Managed beany, je označena anotací @ServerEndpoint, v této anotaci jsou prostřednictvím parametru přidány dekódery pro zprávy v JSON formátu. Podobně jako v MapBeaně je zde umožněn přístup k RobotBeaně pomocí @ManagedProperty, kterou je označena statická proměná RobotBean.

Třída WebSocketApi je označena jako managed beana s parametrem eager nastaveným na hodnotu true. Díky managed beaně můžeme do api injectovat statický atribut RobotBean a komunikovat přímo s hlavní beanou aplikace. Aby mohlo WebSocketApi správně fungovat, tak je potřeba označit třídu anotací @ServerEndpoint, jak již název napovídá, jedná se o koncový uzel, ke kterému mohou uživatelé (Buggyman) přistupovat. Do parametru této anotace je dobré doplnit atributy decoder, encoder a value. Value je řetězcový paramater, jenž obashuje část URL (například "/get"), dále jsou tu parametry encoder a decoder, u těchto parametrů je nastavena třída MessageEncoderDecoder. Tato třída zajišťuje překládání zpráv mezi JSON formátem a stringovým řetězcem. Díky tomuto řešení je zdrojový kód websocketu přehlednější, protože se již nemusí zabývat překladem zpráv.

V komunikačním Api jsou na straně serveru implementovány pouze tři základní websocket metody. První je metoda open, která se volá při otevření socketu na základě anotace @OnOpen, jíž je metoda označena. Obdobou metody open je close, ta funguje úplně stejně, ale volá se při uzavření socket a je označena anotací @OnClose. Jádrem celé komunikace je metoda onMessagge se stejně nazvanou anotací. V hlavičce metody je mimo parametr session navíc parametr message.

Instance message vznikne dekódováním JSON zprávy v třídě MessageEncoder-Decoder, viz ukázka:

```
Reader reader = new StringReader(jsonMessage);
JsonReader jsonReader = Json.createReader(reader);
JsonObject object = jsonReader.readObject();
Message m = new Message();
m.setMessage(object.getJsonString("message").getString());
m.setRunAble(object.getBoolean("runAble"));
m.setLatitude(Double.parseDouble(object.getJsonNumber("latitude").toString()));
m.setLongitude(Double.parseDouble(object.getJsonNumber("longitude").toString()));
```

Kominakce mezi BuggyManem a serverem funguje na principu výměny stavových zpráv. Metoda onMessage je v podstatě listener, který při příchodu nové zprávy nejprve zjistí typ zprávy z atributu message, podle něj se pak provede určená série příkazů viz následujicí ukázka, kde je v atributu message nastaven typ zprávy #hello. Po ověření atributu je do RobotBeany nastavena pozice robota a je zpřístupněna aplikace pomocí metody setRunAble.

```
if (StatusCodes.hello.equals(message.getMessage())) {
        bean.setLatitude(message.getLatitude());
        bean.setLongitude(message.getLongitude());
        bean.setRunAble(message.getRunAble());
        bean.setStatusCode(StatusCodes.WAIT);
        bean.testUserInSession();
}
```

Při každém přichodu nové zprávy se vždy testuje, zda je uživatel stále připojen k aplikaci, nebo zda se nečekaně neodpojil. Pokud by měl uživatel zarezervovaného robota a odpojil se od aplikace aniž by před tím ukončil průvod, zůstal by Buggyman stále zarezervován a byl nepřístupný pro ostatní uživatele. Tento problém je vyřešen pomocí jednoduchého testování, kde se pomocí času zjišťuje poslední interakce uživatele, při níž je zaznamenán čas. Pokud je tento čas starší než 1 minuta, tak se automaticky provede uvolnění robota pro jiné uživatele.

### Seznam cílových míst

Seznam cílových míst je zaznamenán v souboru data.xml, jenž se nachází v adresáři resources. Celý soubor je při vytvoření Map beany načten do třídy DestinationList, kde jsou data z xml souboru pharsována do listu, který je složen z jednotlivých destinací. Každá destinace je uložena jako třída TargetDestination s atributy name, description, latitude a longitude. Po jejich vytvoření je list předán do map beany z níž je dále předán do JavaScriptu, ten zajišťuje vytvoření a umístění jednotlivých značek (cílových míst) na mapě.

Ukázka souboru data.xml:

```
<destination id=”1”>
    <name>Tulipan</name>
    <description>popis</description>
    <latitude>49.2136519</latitude>
    <longitude>16.6154161</longitude>
</destination>
```

### Lokalizace

Pro případ rozšíření aplikace o další jazykové překlady je v aplikaci implementována lokalizace stránek. Jednotlivé lokalizační properties soubory jsou uloženy v adresáři resources. Pro lokalizaci je nutné nastavit konfigurační soubor faces-config.xml, kde se nastavují jazykové mutace a cesta k properties souborům viz ukázka:

```
<application>
    <locale-config>
        <default-locale>cs</default-locale>
        <supported-locale>en</supported-locale>
    </locale-config>
    <resource-bundle>
        <base-name>cz.mendelu.resources.messages</base-name>
        <var>msg</var>
    </resource-bundle>
</application>
```

O změnu jednotlivých mutací se již stará JSF kontejner prostřednictvím metody localeChanged, jenž je implementována ve třídě Local. Textové zprávy a jednotlivé ovládací prvky mají v resource záznamech evidován svůj název, k těmto záznamům je možné jednoduše přistoupit přímo v šabloně, prostřednictvím proměné ”msg”, která je definována v konfiguračním souboru.

## 4.8 Testování aplikace

Pro testování aplikace bylo využito jednak testování uživatelem, díky čemu se podařilo odhalit a vyřešit množství chyb, které při implementaci vznikly. Jako příklad můžeme uvést incident, při kterém se uživatel odpojí od aplikace v době, kdy má zarezervovaného BuggyMana a díky tomu zůstane BuggyMan nedostupný pro ostatní uživatele.

Pro automatizaci testování byl zvolen nástroj Selenium WebDriver v kobinaci s jUnit testy, kde bylo vytvořeno několik testů. Na začátku každého testu je nejprve vytvořena instance WebDriveru. Pomocí metody get přistoupíme k webové stránce ze zadaného URL. Jakmile se zobrazí požadovaná webová stránka, je možné na ní simulovat libovolné operace stejným způsobem jako uživatel. WebDriver zpřístupňuje například stisk tlačítka, klknutí na odkaz, vyplnění formuláře, získání textu ze stránky apod. V následujicí ukázkovém kódu probíhá testování úvodní stránky z inputem pro jméno, testuje se kliknutí na tlačítko pomocí metody findElementy, po nalezení tlačítka podle jeho id, se provede stisknutí metodou click. V první částí kódu se nejprve spustí simulační prostředí BuggyMana a provede otevření websocketu a zpřístupnění aplikace, poté se provede přepnutí do nové záložky pomocí klávesové zkratky ctrl+t, kde se načte aplikace webRobot. Po implicitním čekání se provede stisk tlačítka, při kterém by se měla zobrazit chybová hláška. Pokud se zobrazí, provede se její otestování pomocí assertu.

```
WebDriver driver = new FirefoxDriver();
driver.get("http://localhost:8080/WebRobot/faces/seocketBuggyman.html");
driver.findElement(By.id("open")).click();
driver.findElement(By.id("hello")).click();
driver.findElement(By.cssSelector("body")).sendKeys(Keys.CONTROL + "t");
driver.get("http://localhost:8080/WebRobot/");
driver.manage().timeouts().implicitlyWait(5, TimeUnit.SECONDS);
driver.findElement(By.id("indexForm:send")).click();
Thread.sleep(1000);
WebElement we = driver.findElement(By.className("ui-message-error-icon"));
Assert.assertNotNull(we.getText());
driver.quit();
```

V dalším testu je ověřováno zarezervování robota, testování probíhá podobným způsobem jako první test, ale je tu přidán další WebDriver pro prohlížeč Google Chrome, kde se spustí druhý uživatel. Na začátku testování se opět provede připojení BuggyMana k aplikaci, dále se v prvním driveru provede proklikání aplikací až do stavu, kdy se zavolá (zarezervuje BuggyMan). Po tomto kroku se sputí Google Chrome, provede se stejný postup a v assertu se otestuje, zda se zobrazí správná hláška. Ve třetím testu je simulován průchod celou aplikací, jednotlivé kroky jsou testovány pomocí assertu podobně jako v předchozích testech.

# 5 Závěr

Cílem této bakalářské práce bylo navrhnout a implmentovat webovou aplikaci pro interakci s mobilním robotem. Principem aplikace je umožnit návštěvníkům arboreta využit mobilního robota BuggyMan jako průvodce po areálu. Ačkoliv v současně době není robot k této funkci uzpůsoben, byl tento cíl práce splněn a pro otestování aplikace byl BuggyMan pouze simulován jednoduchou aplikací.

V aplikaci byl kompletně navržen a implementován systém, který zadanou funkci průvodu zajišťuje a díky tomu robot nemusí zasahovat do logiky procesu. Jeho jedinou funkcí je posílat navržené aplikaci svoje GPS souřadnice, informaci o svém stavu a přijímat cílové souřadnice, na které se má přesunout. Dále bylo vytvořeno grafické prostředí aplikace včetně responzivního designu, jelikož se předpokládá, že většina návštěvníků bude využívat mobilní zařízení.

V závěru vývoje bylo provedeno testování navržené aplikace zejména uživatelským testováním, při němž byly zjištěny a opraveny některé nedostatky, které při implementaci vznikly. Pro automatizaci testování bylo napsáno několik jednotkových testů, při těchto testech nebyly zjištěny žádné nedostatky, které by znemožňovaly správnému fungování aplikace.

Jako možná vylepšení je možné úvest například navržení nových značek do mapy, doplnění seznamu cílových míst, jelikož aktuální seznam slouží pouze pro otestování aplikace a obsahuje nepřesné informace. Dalším zlepšením by mohlo být vytvoření uživatelského rozhraní pro editaci cílových míst. Pro nasazení do provozu a navázání komunikace s robotem není vyloučeno, že dojde k drobné úpravě komunikačních zpráv pro potřeby obou stran.

# 6 Seznamy

## Seznam obrázků

**Obrázek 1: EJB container system level** **13**

**Obrázek 2: JSF-Simple request flow** **15**

**Obrázek 3: Princip JavaScriptu** **16**

**Obrázek 4: BuggyMan** **24**

**Obrázek 5: Drátěný model** **27**

**Obrázek 6: Diagram průběhu průvodu** **28**

**Obrázek 7: Adresářová struktura** **29**

**Obrázek 8: Šablona robotmap** **32**

# 7 Reference

HEROUT, P. *Učebnice jazyka Java*. České Budějovice: Kopp, 2010. ISBN 978-80-247-2653-3.

Písek S. *HTML začínáme programovat*. Praha: Grada, 2014. ISBN 978-80-247-5059-0.

HOGAN, B. P. *HTML5 a CSS3 Výukový kurz webového vývojáře*. Praha: Computer Press, 2013.ISBN 978-80-251-3576-1..

DARIE, C. a kol. *AJAX a PHP : tvoříme interaktivní webové aplikace profesionálně*. Brno: Zoner Press, 2006. 80-86815-47-1.

ZAKAS, NICHOLAS C. *JavaScript pro webové vývojáře*. Vyd. 1. Brno: Computer Press, 2009, 832 s. ISBN 978-80-251-2509-0.

POKORNÝ, J. – HALAŠKA, I. *Databázové systémy*. Praha: Vydavatelství ČVUT, 2003, ISBN 80-01-02789-9..

Mobile phone tracking. *wikipedia.org*. [online]. [cit. 2016-3-20]. Dostupné z: https://en.wikipedia.org/wiki/Mobilephonetracking.

Enterprise Java Beans. *Oracle.com*. [online]. [cit. 2014-05-10]. Dostupné z: https://docs.oracle.com/javaee/7/firstcup/java-ee001.htm.

Čápka, D. Úvod do jazyka Java *ITNetwork.cz*. [online]. 2015 [cit. 2015-8-5]. Dostupné z: http://www.itnetwork.cz/java/zaklady/java-tutorial-uvod-do-jazyka-java.

Janovský D. Úvod do JavaScriptu *jakpsatweb.cz*. [online]. 2007 [cit. 2007-5-15]. Dostupné z: http://www.jakpsatweb.cz/javascript/javascript-uvod.html.

JavaServer Faces Overview. *Oracle.com*. [online]. [cit. 2015-12-10]. Dostupné z: http://www.oracle.com/technetwork/java/javaee/overview-140548.html.

jQuery Introduction. *w3schools.com*. [online]. [cit. 2015]. Dostupné z: http://www.w3schools.com/jquery/jquery_intro.asp.

Bluemix overview. *Bluemix.net*. [online]. [cit. 2015-12-15]. Dostupné z: https://www.ng.bluemix.net/docs/overview/index.html.

ONDROUŠEK, V. Mobilní robot Buggyman. *aistorm.mendelu.cz*. [online]. [cit. 2015-12-10]. Dostupné z: https://aistorm.mendelu.cz/cz/projekty/buggyman.

PrimeFaces. *primefaces.org*. [online]. [cit. 2015-12-28]. Dostupné z: http://www.primefaces.org/documentation.

CSS. *w3.org*. [online]. [cit. 2015]. Dostupné z: https://www.w3.org/Style/CSS/.

Assisted GPS. *wikipedia.org*. [online]. [cit. 2016-3-20]. Dostupné z: https://en.wikipedia.org/wiki/Assisted_GPS.

Global Positioning System. *wikipedia.org*. [online]. [cit. 2016-3-20]. Dostupné z: https://en.wikipedia.org/wiki/Global_Positioning_System.

Wi-Fi positioning system. *wikipedia.org*. [online]. [cit. 2016-3-20]. Dostupné z: https://en.wikipedia.org/wiki/Wi-Fi_positioning_system.

Apache Maven. *wikipedia.org*. [online]. [cit. 2016-3-24]. Dostupné z: https://en.wikipedia.org/wiki/Apache_Maven.

Fáze a úrovně provádění testů. *Testování softwaru* [online]. [cit. 2015-3-1]. Dostupné z: http://testovanisoftwaru.cz/tag/akceptacnitestovani/system.

Automatizace. *wikipedia.org*. [online]. [cit. 2016-1-25]. Dostupné z: https://cs.wikipedia.org/wiki/Automatizace.

Selenium. *seleniumhq.org*. [online]. [cit. 2016-4-25]. Dostupné z: http://www.seleniumhq.org/.