Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

BAKALÁŘSKÁ PRÁCE

Aplikace pro sledování funkčnosti webu

2017

Michal Gold

Vedoucí práce: Mgr. Jiří Zacpal, Ph.D.

Studijní obor: Aplikovaná informatika, prezenční forma

Bibliografické údaje

Autor:	Michal Gold
Název práce:	Aplikace pro sledování funkčnosti webu
Typ práce:	bakalářská práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2017
Studijní obor:	Aplikovaná informatika, prezenční forma
Vedoucí práce:	Mgr. Jiří Zacpal, Ph.D.
Počet stran:	42
Přílohy:	1 CD/DVD
Jazyk práce:	český

Bibliograhic info

Author:	Michal Gold
Title:	Application for monitoring the functioning of the Web
Thesis type:	bachelor thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2017
Study field:	Applied Computer Science, full-time form
Supervisor:	Mgr. Jiří Zacpal, Ph.D.
Page count:	42
Supplements:	1 CD/DVD
Thesis language:	Czech

Anotace

Cílem práce je vytvořit nástroj pro monitorování webových serverů (dostupnost, rychlost odezvy, proběhlé zálohy, poslední objednávky). Vytvořená webová aplikace bude pomocí vhodného grafického rozhraní zpřístupněna jednak vlastníkům monitorovaných webů, tak administrátorovi celé aplikace. Součástí práce je také responzivní design, který umožní snadné ovládání skrze mobilní zařízení.

Synopsis

The aim of this work is to create a web server monitoring tool measuring availability, response rate, backups, and last orders. The resulting web application will be made available to the owners of the monitored sites and to the administrator of the entire web application using the appropriate graphical interface. Part of the work is also a responsive design that allows easy control over mobile devices.

Klíčová slova: Monitorování; Java; Oracle; Hibernate; EJB3; Liferay; JMS; MDB; JSF; Webová služba; WSDL; SOAP

Keywords: Monitoring; Java; Oracle; Hibernate; EJB3; Liferay; JMS; MDB; JSF; Web Services; WSDL; SOAP

Poděkovat bych chtěl především svému vedoucímu práce Mgr. Jiřímu Zacpalovi, Ph.D, za odborné vedení, cenné rady a připomínky v průběhu psaní této bakalářské práce. Ve firmě M.I.T. Consulting, s.r.o. pak Bc. Petrovi Freibergovi za cenné programátorské rady. Dále bych chtěl poděkovat rodině a přátelům za podporu během celého studia.

Místopřísežně prohlašuji, že jsem celou práci včetně příloh vypracoval/a samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.

datum odevzdání práce podpis autora

Obsah

Seznam obrázků

Seznam tabulek

Seznam vět

Seznam zdrojových kódů

1 Úvod

V dnešní době, kdy je dostupnost webových aplikací stále kritičtějším požadavkem, a už krátkodobý výpadek může u velkých, ale i menších společností způsobit rozsáhlé škody, je monitoring nezbytnou výbavou každého IT oddělení. Správci takových aplikací však často na správný monitoring zapomínají, nebo mu nevěnují dostatečnou pozornost. To může být způsobeno jak nízkým povědomím o celé problematice monitorování, tak nedostatkem lidských kapacit k vybudování kvalitního monitorovacího nástroje.

V minulých letech na trhu monitorovacích nástrojů začala podnikat celá řada firem, které se snaží poskytovat profesionální monitorovací nástroje. Mnoho aplikací zabývajících se monitoringem webů je zaměřena především na dostupnost nebo zjištění přítomnosti určitého prvku na stránce. Většina aplikací je samozřejmě zpoplatněna nebo zdarma poskytuje jen značně omezenou funkcionalitu. Cílem práce je vytvořit aplikaci, která bude poskytovat podobnou funkcionalitu a bude moci těmto aplikacím konkurovat. Byly zvoleny takové technologie a nástroje, pomocí kterých je možné zajistit do budoucna snadnou škálovatelnost. První verze aplikace je zaměřena na co největší obecnost, aby v dalších verzích bylo snadné přidávat novou funkcionalitu.

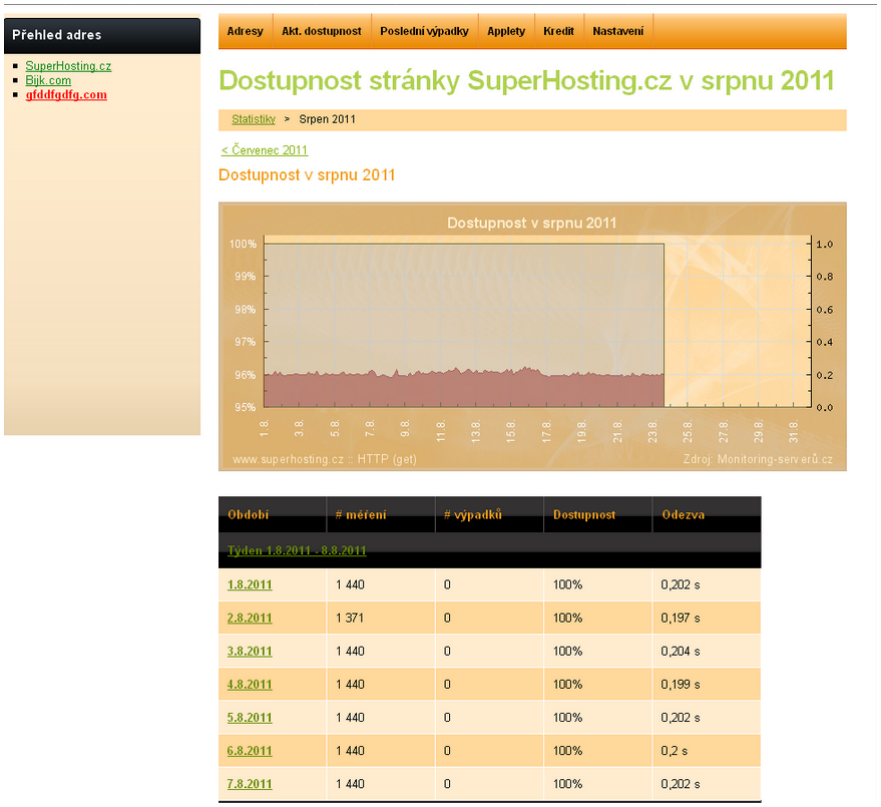
Aplikaci jsem začal původně vyvíjet na základě požadavku firmy IZON s.r.o. Postupem času firma o aplikaci přestala mít zájem, a tak jsem mohl vývoj směřovat dle svého vlastního uvážení. Bylo tedy jen na mě, jakou funkcionalitu aplikace bude nabízet, a jaké technologie budou použity. Z tohoto důvodu jsem upustil od mobilní aplikace a rozhodl se pro vývoj za použití responzivní technologie na GUI (grafické uživatelské rozhraní) části a Javy namísto PHP (skriptovací programovací jazyk) na straně serveru. Implementace pomocí responzivní technologie umožňuje stejně kvalitní zobrazení jak na desktopových, tak mobilních zařízeních. V rámci vývoje a dalšího rozvoje tedy není nutné spravovat více grafických podob, což mi vzhledem k tomu, že na aplikaci pracuji sám, velmi vyhovuje.

Bakalářská práce je členěna následovně: v následující kapitole rozebírám alternativní řešení, která jsem před vývojem své vlastní aplikace prostudoval. Ve třetí kapitole popisuji zvolené technologie, které byly použity při implementaci aplikace, kdy hlavním měřítkem byla především snadná škálovatelnost. Čtvrtá kapitola demonstruje aplikaci a práci s ní tak, aby čtenář po jejím přečtení mohl aplikaci snadno používat. V závěru nastiňuji budoucí vývoj, ve kterém plánuji dále pokračovat.

2 Alternativní řešení

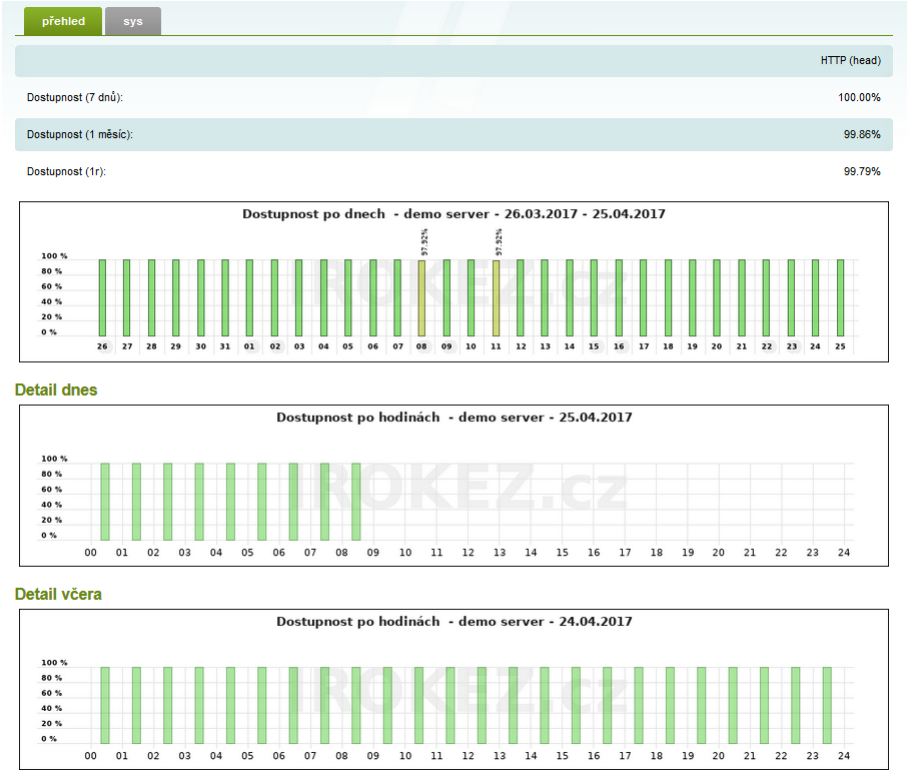
Aplikace se stejným zaměřením již existují, ale ne všechny jsou zcela zdarma. V této části je popsáno několik vybraných aplikací zabývajících se stejnou problematikou:

1. SuperNetwork s.r.o. - nalezneme na adrese www.monitoring-serveru.cz. Společnost monitoruje připojení na server/službu. Způsob dotazovaní je závislý na povaze služby (může jít o požadavek na www stránku, připojení k FTP atd.). Zákazník zde může nastavit časový interval v jakém se bude jeho web kontrolovat, možnost zaslání informace o výpadku či statistik. Cena služby se odvíjí od počtu monitorovaných instancí a časovém intervalu, ve kterém probíhá monitoring. Kupříkladu pokud se u dvou webových stránek dotazujeme v intervalu pěti minut a u třetí stránky v intervalu jedné minuty, pak měsíční paušál za tuto službu činí **400,50 Kč vč. DPH**. Ukázku přehledu měřeného webu u společnosti SuperNetwork s.r.o můžeme vidět na obrázku č. 1.



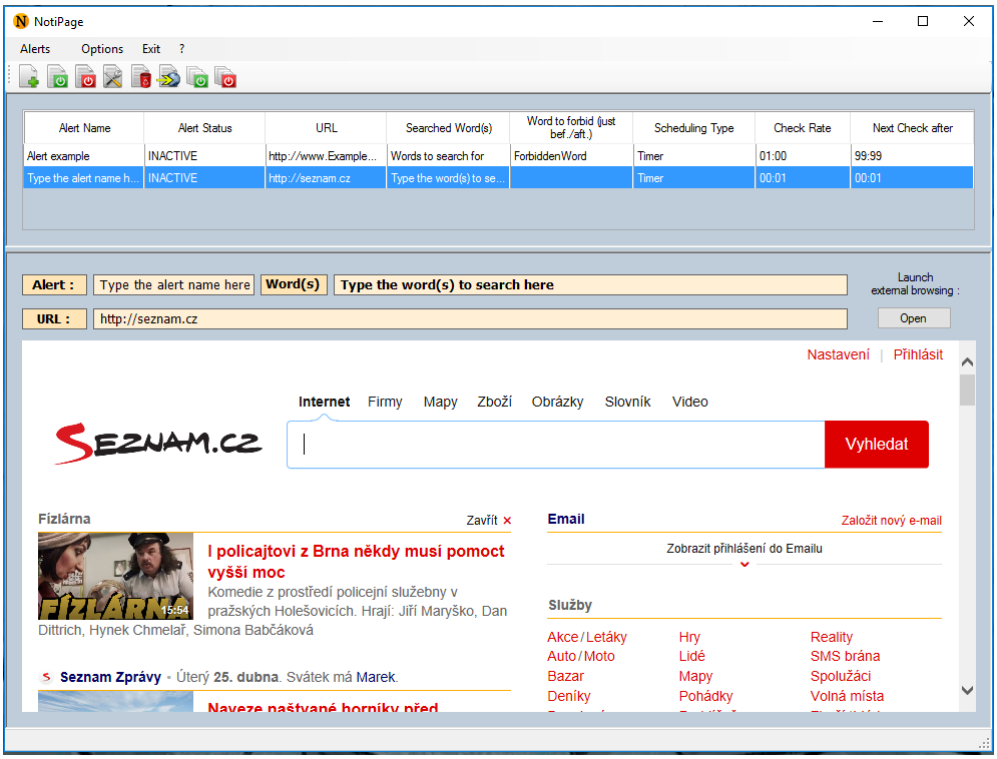
Obrázek 1: Náhled aplikace společnosti SuperNetwork s.r.o

2. Irokez - nalezneme na adrese www.irokez.cz. Společnost monitoruje známé české webové stránky jako je například Seznam.cz nebo Stream.cz. Monitorování probíhá v časových intervalech dle tarifu od 1 do 30 minut. V případě, že server nestihne odpovědět v určeném časovém limitu nebo je vrácen jiný kód než 200 OK (u HTTP), je automaticky vyhodnocen jako nefunkční. Irokez dále podporuje upozornění zákazníka prostřednictvím emailu nebo SMS. Zákazník má k dispozici vizuální přehled vytíženosti v určitém časovém úseku pomocí grafu. Kupříkladu pokud se u dvou webových stránek dotazujeme v intervalu pěti minut a u třetí stránky v intervalu jedné minuty, pak měsíční paušál za tuto službu činí **219 Kč vč. DPH**. Do paušálu se nepočítají zaslané upozorňující SMS, přičemž nastavení upozornění pomocí SMS je volitelné. Cena za jednu odeslanou SMS je 1 Kč. Náhled detailu webu je zobrazen na obrázku č. 2.



Obrázek 2: Náhled aplikace společnosti Irokez

3. NotiPage - jedná se o desktopovou aplikaci, kterou si každý uživatel muže zdarma stáhnout a nainstalovat do svého počítače. Aplikace není tak rozsáhlá jak výše popsané, ale nenáročným uživatelům určitě poslouží. Poskytuje monitorování funkčnosti stránky, kontrolu aktualizací a obsahu. Monitorování funkčnosti lze nastavit jak na interval, tak i na konkrétní datum. Jakmile je změněn například obsah stránky, je uživatel upozorněn vyskakujícím oknem a zvukovým efektem. Text i zvuk upozornění může každý uživatel změnit dle svého vlastního uvážení. Aplikace nevyžaduje žádné licence, její **provoz je tedy zcela zdarma**. Náhled detailu aplikace je zobrazen na obrázku č. 3.



Obrázek 3: Náhled aplikace NotiPage

Aplikaci je možné využít například k upozornění na nové články či přidání nového komentáře u webu, který nepodporuje RSS čtečky. Uživatel tedy nemusí kontrolovat oblíbenou stránku, stačí jen nastavit do aplikace adresu stránky a při změně obsahu je automaticky upozorněn.

4. Hlidam.to - nalezneme na adrese www.hlidam.to. Společnost momentálně monitoruje okolo 50 webů. Nabízí monitorování dostupnosti v intervalu 1, 5 a 10 minut. K dispozici je přehled dostupných i nedostupných měření pomocí grafu a tabulky. Při výpadku je zákazníkovy zasláno upozornění emailem nebo prostřednictvím SMS. Kupříkladu pokud se u dvou webových stránek dotazujeme v intervalu pěti minut a u třetí stránky v intervalu jedné minuty, pak měsíční paušál za tuto službu činí **106 Kč vč. DPH**. Do paušálu se nepočítají zaslané SMS při upozornění, přičemž nastavení upozornění pomocí SMS je volitelné. Cena za jednu odeslanou SMS je 2 Kč. Náhled aplikace je vidět na obrázku č. 4.

Obrázek 4: Náhled aplikace společnosti HlídámTo

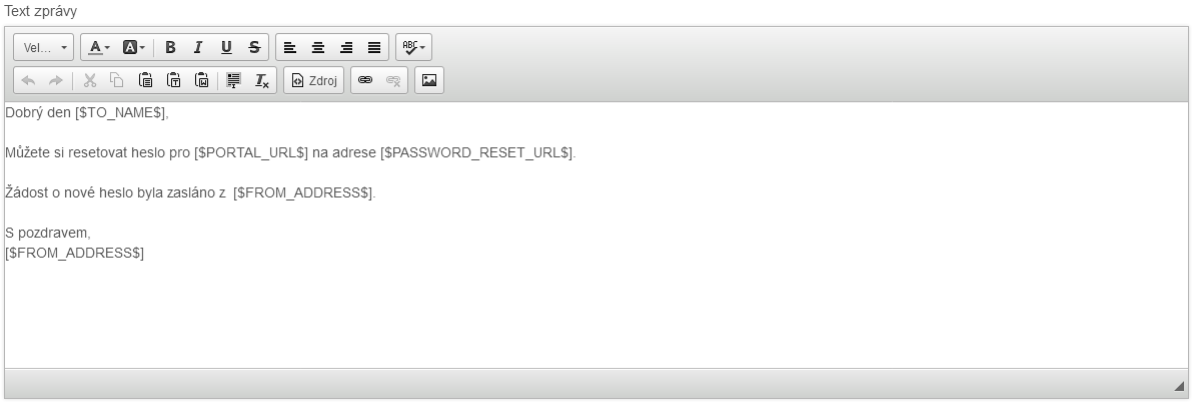
3 Technologie

V následující kapitole a jejích podkapitolách jsou popsané technologie, které jsem použil při implementaci mé aplikace. U každé z nich jsem se zaměřil na obecný popis a detailněji popsal jen ty dané technologie části, se kterými jsem pracoval. U některých technologií je popsáno i konkrétní využití v rámci aplikace.

3.1 Liferay Portal

Liferay Portal je bezplatný open-source podnikový portál založený na jazyce Java a distribuovaný pod licencí GNU Lesser General Public Licence a dalšími proprietárními licencemi. Umožňuje správu dat, aplikací a procesů z jednoho centrálního uživatelského rozhraní. Portál se skládá z jednotlivých funkčních jednotek, které se nazývají portlety. [1]

Dále portál podporuje správu uživatelů a jejich obsluhu (např. odeslání upozorňujících emailu při založení nového uživatele, žádost o zapomenuté heslo, změnu hesla a mnohé další). Je zde možnost modifikovat textovou šablonu, kterou portál zasílá při některém emailovém upozornění příklad viz obr. 5. Aby zasílání emailu bylo funkční je potřeba nastavit SMTP server, skrze který bude probíhat komunikace.



Obrázek 5: Nastavení šablony pro email

Další z možností je upozornění uživatelů v rámci portálu pomocí Liferay notifikací, které jsou již zakomponované jako doplněk (modul). Abychom mohli notifikace používat je nutnost je v nastavení doplňků povolit.

Portál zahrnuje i schopnost responzivního designu s využitím potenciálu HTML5. Při vývoji je možnost přepnout se do mobilního či tabletového emulátoru a vidět tak jak se naše stránka bude zobrazovat na těchto zařízeních. V základním nastavení podporuje 22 jazyků, mezi kterými nechybí samozřejmě ani čeština.

3.1.1 Portlety v Liferay

Myšlenkou využití portletů v rámci Liferay je tvořit portál z menších nezávislých částí, které by šly zabalit a používat v rámci dalších portálů. Portlety jsou tedy samostatné aplikace psané v Javě, které tvoří obsah a zajišťují nějakou konkrétní funkcionalitu. Například aplikace pro odesílaní SMS s využitím databáze uživatelů Liferay, e-learningové aplikace pro školení zaměstnanců ve firmě, skladové systémy a samozřejmě také monitorování webu.

Ihned po stažení a spuštění čisté verze Liferay je k dispozici více než 60 portletů různých kategorií, jako je správa obsahu, mapy Google, Wiki, Blogy, RSS zprávy a jiné. Má aplikace se skládá ze tří portletů: administrace uživatelů, administrace zákazníků a moje weby.

3.1.2 Role v Liferay

Role zde považujeme za sbírku oprávnění, která definují kupříkladu přístup k portletům. V rámci Liferay jsou definované některé základní role jako například: Administrátor, Vlastník či Host. Je zde samozřejmě možnost definovat i role vlastní. Role přiřazujeme jak uživatelům, tak samostatným portletům.

U mé aplikace jsou vytvořené dvě role „Administrátor“ a „Zákazník“. K administračním portletům je přiřazena role „Administrátor“ a k zákaznickým pak role „Zákazník“. Jakmile se uživatel přihlásí do portálu uvidí jen ty portlety, na které má přiřazenou příslušnou roli.

Spravovat role lze jak v rámci administrace Liferay, tak skrze kód prostřednictvím běžících portletů. Přehledný popis API, rozdělený pro jednotlivé verze Liferay, lze nalézt na https://docs.liferay.com/portal/.

3.2 Oracle Database

Oracle Database je moderním multiplatformním databázovým nástrojem, a jedná se dnes o jednu z nejrozšířenějších relačních databází. Databáze zajišťuje uchovávání, zpřístupnění a konzistenci dat. Také zde probíhá agregace či předzpracování uložených dat. Tento systém řízení báze dat je vyvíjen firmou Oracle Corporation, která se řadí mezi významné společnosti zabývající se tvorbou relačních databází a nástroji pro vývoj a správu databází.

V rámci svého projektu používám databázi ve verzi 12g a pro správu využívám program SQLDeveloper, taktéž od společnosti Oracle.

Aplikace s databází nekomunikuje na přímo, ale využívá jako mezivrstvu objektově-relační mapování skrze framework Hibernate. To mi umožňuje do budoucna snadnou migraci na případná jiná databázová řešení (PostgreSQL, MySQL). Informace jsou z čerpané z [4].

3.2.1 Hibernate

Hibernate je framework napsaný v jazyce Java. Je jednou z implementací JPA (Java Persistence API). Umožňuje objektově-relační mapování (ORM), tedy pod-

poruje ukládání dat objektově-orientovaných jazyků do relační databáze.

Třídy, které jsou mapovány do databáze, se nazývají „Entity“. Entita je standardní Java třída, která obsahuje anotaci „@Entity“. Datové typy v této třídy jsou pak automaticky konvertovány na datové typy SQL. Vztahy mezi objekty jsou reprezentované pomocí spojení tabulek a cizích klíčů.

Každé entitě odpovídá jedna tabulka v databázi, jejíž název je buď automaticky vygenerován z názvu třídy, nebo je přejat z anotace „@Table(name="Nazev")“, která je obsažena nad definici entity. Entitní třída musí obsahovat bezparametrický konstruktor. Proměnné jsou zpravidla definované jako private a jsou pro ně vytvořeny přístupové metody (gettery, settery). Mapování entity do databázové tabulky může být provedeno pomocí mapovacího souboru nebo anotací. U mého projektu jsem použil mapování pomocí anotací.

Ukázka použití JPA Anotace:

```
@Table(name = "BACKUP_LOG")
public class BackupLog {

    @Id
    @Column(name = "UUID", nullable = false)
    private String uuid;

    @Column(name = "BACKUP_TIME", nullable = false)
    private Date backupTime;

    @PrePersist
    protected void onCreate() {
        createdDate = new Date();
    }

    public BackupLog() {
        super();
    }

    public void setUuid(String uuid) {
        this.uuid = uuid;
    }

    public Long getBackupId() {
        return backupId;
    }

    public void setBackupTime(Date backupTime) {
        this.backupTime = backupTime;
    }
```

```
    public Date getBackupTime() {
        return backupTime;
    }}
```

Je zde možnost pomocí jazyku HQL (Hibernate Query Language), který je součástí Hibernate, psát dotazy přímo z kódu aplikace. Jazyk HQL je podobný jazyku SQL, není nutné se tedy učit něco nového a stačí znát základní syntax jazyka SQL. Jeho výhodou je však objektový přístup k databázi.
Jelikož jsem použil mapování entit pomoci JPA anotací, tak pro ukládání dat používám javax.persistence.EntityManager. Ukázka uložení entity do databáze:

```
    @PersistenceContext(unitName = "monitorovaniWeb")

    private EntityManager em;

    public void createCustomer(){
        Customer customer = new Customer();
        customer.setFirstName("Jan");
        customer.setSurname("Novák");
        em.persist(customer)
}
```

Výhodou Hibernate je podpora mnoha dialektů, jako je například: MySQL, PostgreSQL, Oracle, Microsoft SQL Server. Budu-li chtít přejít na jinou databázi, stačí jen v konfiguraci změnit dialekt, aby Hibernate používal správné datové typy a správnou syntaxi SQL. Hibernate taktéž umožňuje využití specifik jednotlivých databází, avšak pak musíme mít na paměti, že migrace na jinou databázi sebou může přinést potřebu zásahu do existujícího kódu.

3.3 JBoss

Aplikační servery jsou vrstvou mezi operačním systémem a aplikacemi. Pohání nejčastěji prostřední vrstvu architektury (u modelu MVC vrstvu C, Controller), ve které leží jádro aplikace, její logika a funkce. V této vrstvě probíhají výpočty a operace mezi vstupně-výstupními požadavky.

Jako aplikační server pro svou aplikaci používám JBoss 7.1 od společnosti Red Hat. Novější verze tohoto aplikačního serveru nesou jméno Wildfly. JBoss je tzv. "Java EE certified - Full", implementuje tak všechny hlavní funkcionality popsané v rámci standardu Java EE. Informace jsou čerpané z [5].

Pro názornou ukázku těchto technologií můžete nahlédnout do zdrojových kódů mé aplikace. Podrobnější informace o konkrétních použitých technologií naleznete v kapitole Java EE 3.4.

3.4 Java

Java je jedním z nejpoužívanějších programovacích jazyků. Označuje kromě programovacího jazyka také platformu. Platforma je složena z virtuálního stroje a příslušného API.

- Virtuálním strojem je myšleno prostředí kde naše aplikace běží.
- API je sada přeprogramovaných tříd, které nám umožňují přístup k virtuálnímu stroji.

Existují 3 Java platformy - Java SE (Standart Edition), Java ME (Micro Edition) a Java EE (Enterprise Edition). Java EE je nadstavbou Java SE. K provozování Javy EE na počítači či serveru, je nutné mít nainstalované také standardní API Java SE. V následujícím textu bude detailněji rozebrána především Java EE. Více informací dostupné z [3].

Pro příklad uvedu některé z mnoha API, které jsem použil ve své aplikaci. Několik z nich je pak popsáno v samostatných podkapitolách.
Příklad API Javy EE:

- javax.annotation,
- javax.ejb,
- javax.faces,
- javax.jms,
- javax.mail,
- javax.transaction a mnohé další.

3.4.1 EJB3

EJB3 (Enterprise Java Beans 3) jsou serverové komponenty pro tvorbu aplikací. Hlavní myšlenkou je oddělit bussines logiku od prezentační a databázové části a poskytnout programátorovi rozsáhlé API, které ho odstíní od problematiky, se kterou se většina business projektů potýká. Mezi takové problémy například patří:

- podpora souběžného zpracování – programátor je odstíněn od problematiky synchronizace souběžných přístupů ke sdíleným datům,
- bezpečnost – definice přístupů na úrovni tříd a metod,
- management transakcí – správa commitů a rollbacků nad databází,
- pooling – vytváření poolu instancí pro bezstavové beany a message-driven beany (MDB), kdy nám zvětšování a zmenšování poolu umožňuje snadnou škálovatelnost aplikace,
- asynchronní volání metod,
- podpora messagingu - odstiňuje programátora od komplikovaného API Java Messaging Service (JMS),
- zjednodušuje komunikaci se vzdáleným klientem skrze RMI (využívá TCP/IP protokolu).

Využití tohoto API dále zaručuje snadnou integraci s dalšími Java EE technologiemi jako jsou například: JPA, JNDI atd.

EJB mají na aplikačním serveru vyhrazen prostor, tzv. EJB kontejner, kam jsou nasazovány. Kontejner se stará o životní cyklus bean (vytvoření, inicializaci, destrukci). Beany jsou rozděleny na dva základní druhy - Session Beans a Message Driven Beans:

1. Stateless Session Beans (bezstavové beany) - neuchovávají stav mezi klientskými požadavky. Server každému požadavku přidělí novou instanci, která jej obslouží. Třída musí mít anotaci @Stateless. Naprostá většina implementovaných EJB spadá do této kategorie.
2. Stateful Session Beans (stavové beany) - pro každého klienta je vytvořena jedna instance a pro každé volání na server se použije stejná instance obsahující data z předchozího volání. Třída musí mít anotaci @Stateful.
3. Singleton Session Beans - existuje pouze jedna instance beany pro daný JVM. V případě, že aplikace běží v clusteru, je zapotřebí využít další technologii jako je například JBoss Clustering, která zajistí existenci jedné instance i napříč mnoha JVM. Singleton Beany se využívají například pro načítání statických hodnot z databáze, abychom se opakovaně nemuseli dotazovat pro stejné hodnoty.

4. Message Driven Beans - jsou vybudovány nad JMS technologií a slouží k asynchronnímu zpracování dat zaslaných do JMS. Typickým použitím je rozdělení dlouho běžícího úkolu do série mnoha kroků, které jsou postupně, nezávisle na sobě, vykonávány. Příkladem může být pravidelná kontrola konzistence u souborů uložených v souborovém systému, kdy jedna zpráva rovná se ID jednoho souboru. Takto můžeme do JMS zaslat kupříkladu deset tisíc souborů, které jsou postupně instancemi MDB zpracovávány. Velkou výhodou je, že každá zpráva je zpracovávána v samostatné transakci, tedy výjimka při zpracování jedné zprávy neovlivní zprávy ostatní.

Správný výběr scopu u Session Bean je velmi důležitý jak po stránce programátorské, tak optimalizační. Pokud bychom u velkého množství bean použili anotaci @Stateful a měli velkou zátěž ze strany uživatelů v kombinaci s málo výkonným serverem, tak se nám může snadno stát, že aplikaci začne velmi brzo docházet paměť. Stateful Session Beana si totiž udržuje svůj stav a ten musí být mezi jednotlivými požadavky uchováván.

Pro více informací o technologii EJB3 vám mohu doporučit knihu: EJB 3 in Action, Second Edition[2].

3.4.2 JMS

JMS (Java Message Service) je způsob komunikace mezi komponentami a aplikacemi. Umožňuje aplikacím vytvářet, odesílat, přijímat a číst zprávy. Zprávy jsou objekty držené ve frontě v paměti nebo databázi a jsou technologii uchovávány, dokud si je adresát nepřevezme. Na druhé konci fronty naslouchají v našem takzvané Message Driven Bean, tedy třídy reagující na příchod zprávy.

O fronty se stará samotný aplikační server, v mém případě JBoss 7.1.1 s využitím technologie HornetQ. Nakonfigurovat novou frontu znamená upravit soubor standalone.xml a přidat do něj následující řádek:

```
<jms-queue name="název">
 <entry name="queue/název"/>
 <entry name="java:jboss/exported/jms/queue/název"/>
</jms-queue>
```

Aplikace využívá dvě fronty, a to pro odesílání emailu a pro funkci Ping, která se dotazuje na uživatelem konfigurovaný server a kontroluje jeho dostupnost.

Díky využití JMS v kombinaci s MDB mohu v případě rozšířeného zájmu mou aplikaci snadno škálovat, a to pouze změnou několika řádků v konfiguračním souboru, kterými navýším maximální počet instancí, které mohou souběžně zpracovávat zprávy.

3.4.3 JSF

JSF (JavaSever Faces) technologie byla vyvinuta pro snadněji udržovatelný vývoj webových aplikací. Hlavní myšlenou JSF je oddělit logickou a prezentační část stránky.

Základem JSF je technologie JSP (JavaServer Pages), ve kterém figuruje systém knihoven tagů, díky kterým může vývojář vkládat na stránku komponenty uživatelského rozhraní (UI komponenty). Při tvorbě je možné využít vizuální vývojové prostředí nebo psát kód manuálně. Samotná data jednotlivých JSF stránek jsou uloženy v tzv. ManagedBeans, což jsou standardních Java třídy s anotací @ManagedBean(name = "názevBeany"). Každé managed beaně můžeme nastavit scope, tedy rozsah její platnosti, pomocí anotace.

Druhy scope:

- @RequestScoped - existuje pouze v rámci platnosti HTTP požadavku.
- @NoneScoped - existuje tak dlouho jako jedno vyhodnocení elementu.
- @ViewScoped - existuje po dobu interakce se stránkou. Pokud dojde k přepnutí na další stránku, tak je instance zrušena.
- @SessionScoped - existuje po dobu HTTP relace.
- @ApplicationScoped - vytvoří se s prvním požadavkem a existuje po celou dobu života aplikace.

Výchozí nastavení každé managed beany je @RequestScoped.

Ukázka:

```
@ManagedBean(name = "customerPublicPortletBean")
@ViewScoped
public class CustomerPublicBean implements Serializable {
```

Abychom zobrazili požadovanou hodnotu uloženou v ManagedBean, je nutné specifikovat název beany a atribut, který chceme zobrazit.

Příklad dotazování do Managed Beany

```
<p:inputText id="Jmeno" value="#{názevBeany.jmeno}" />
```

Příslušný atribut musí mít implementovány metody get() a set(). Technologii JSF jsem v kombinaci s porlety použil k vybudování své GUI části.

Pro více informací o technologii JSF vám mohu doporučit knihu: Mastering JavaServer Faces.

3.4.4 Webové služby

Webové služby umožňují komunikaci aplikací, které běží v různorodém prostředí, neboť komunikace je založena na společném obecném jazyce. Nejčastěji to bývá XML a protokol HTTP.

V mé aplikaci každou webovou službu popisuje jazyk WSDL (Web Services Description Language). WSDL popisuje, co webová služba poskytuje a jak si o to říci. Ke komunikaci mezi aplikacemi jsem použil protokol SOAP (Simple Object Access Protocol), který umožňuje zasílání zpráv v podobě XML mezi dvěma aplikacemi. SOAP požadavek je odeslán v těle HTTP metodou POST. Komunikace v každý jednotlivý okamžik probíhá pouze jedním směrem, kdy jedna aplikace pošle XML požadavek, druhá aplikace požadavek obslouží a vrátí zpět jako druhý požadavek. Abychom mohli identifikovat SOAP požadavek, musí hlavička HTTP obsahovat „SOAPAction“. SOAP požadavek je automaticky vygenerován z popisu webové služby, tedy z WSDL.

Je zde ještě další možnost, a to použití služby REST (Representational State Transfer), která využívá metody protokolu HTTP (GET, PUT, POST, DELETE) pro předání dat. Oproti technologii SOAP se jedná o režijně méně náročnou technologii, protože nevyžaduje parsování XML.

Technologii SOAP jsem zvolil z toho důvodu, že Java EE ve své aktuální sedmé verzi neobsahuje nativní podporu RESTu. V případě budoucího rozvoje, kdy by bylo zapotřebí mít klienta na platformě, která není tak výkonná jako běžný stolní PC, nebude problém podporu RESTu přidat.

Základní parametry WSDL:

- Types - definuje struktury používané ve zprávách.
- Operations - abstraktní definice operací, které služba podporuje.
- Binding - slouží pro navázání určitého typu portu na konkrétní protokol a formát přenosu zpráv.

Chceme-li se připojit na některou webovou službu stačí pouze vygenerovat klienta z WSDL. Díky tomu, že je na této technologii založena velká spousta aplikací, tak existuje generátor klienta z WSDL pro každý významný programovací jazyk. Generování klienta umožňuje převážná většina vývojových prostředí. Například: Eclipse, JDeveloper.

Nepodporuje-li naše vývojové prostředí generování klienta, nabízí se nám ještě možnost stáhnutí programu přímo pro tuto funkcionalitu (např. WebServiceStudio), pak už jen stačí zadat cestu k WSDL a klient je automaticky vygenerován.

Generátory vygenerují soubory s definicí metod, které zpřístupňuje WSDL a klienta obsahující kód pro vytvoření proxy objektu pro umožnění volání metod webové služby.

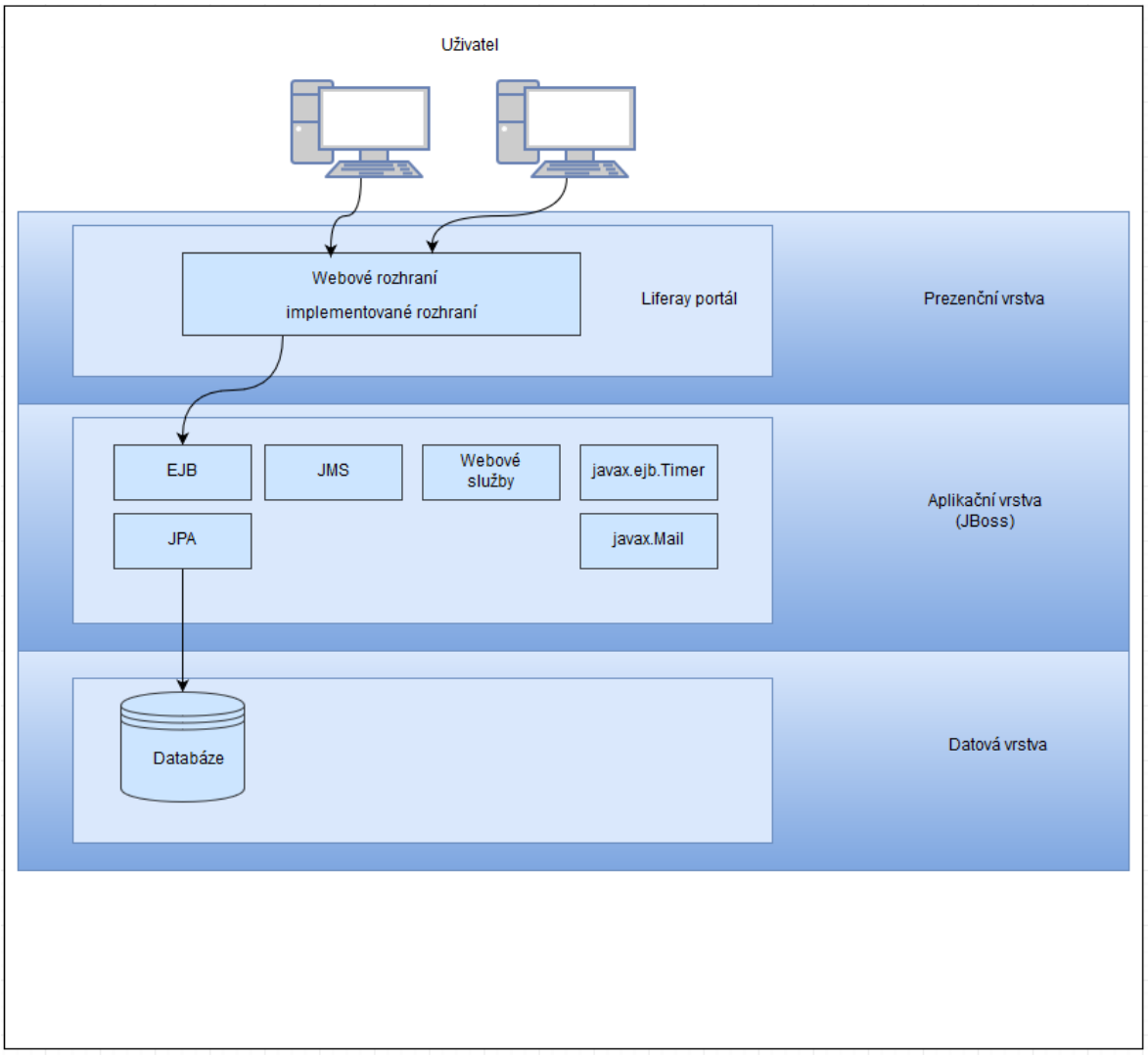
Má aplikace obsahuje dvě webové funkce vybudované na popsané technologii. První funkce slouží k ukládaní informací o pravidelných zálohách, druhá funkce pak slouží k ukládaní informací o provedených objednávkách.

4 Aplikace

Kapitola popisuje detailní návrh aplikace a příručku k použití. Příručka obsahuje náhled a popis jak s aplikací pracovat.

4.1 Diagram architektury

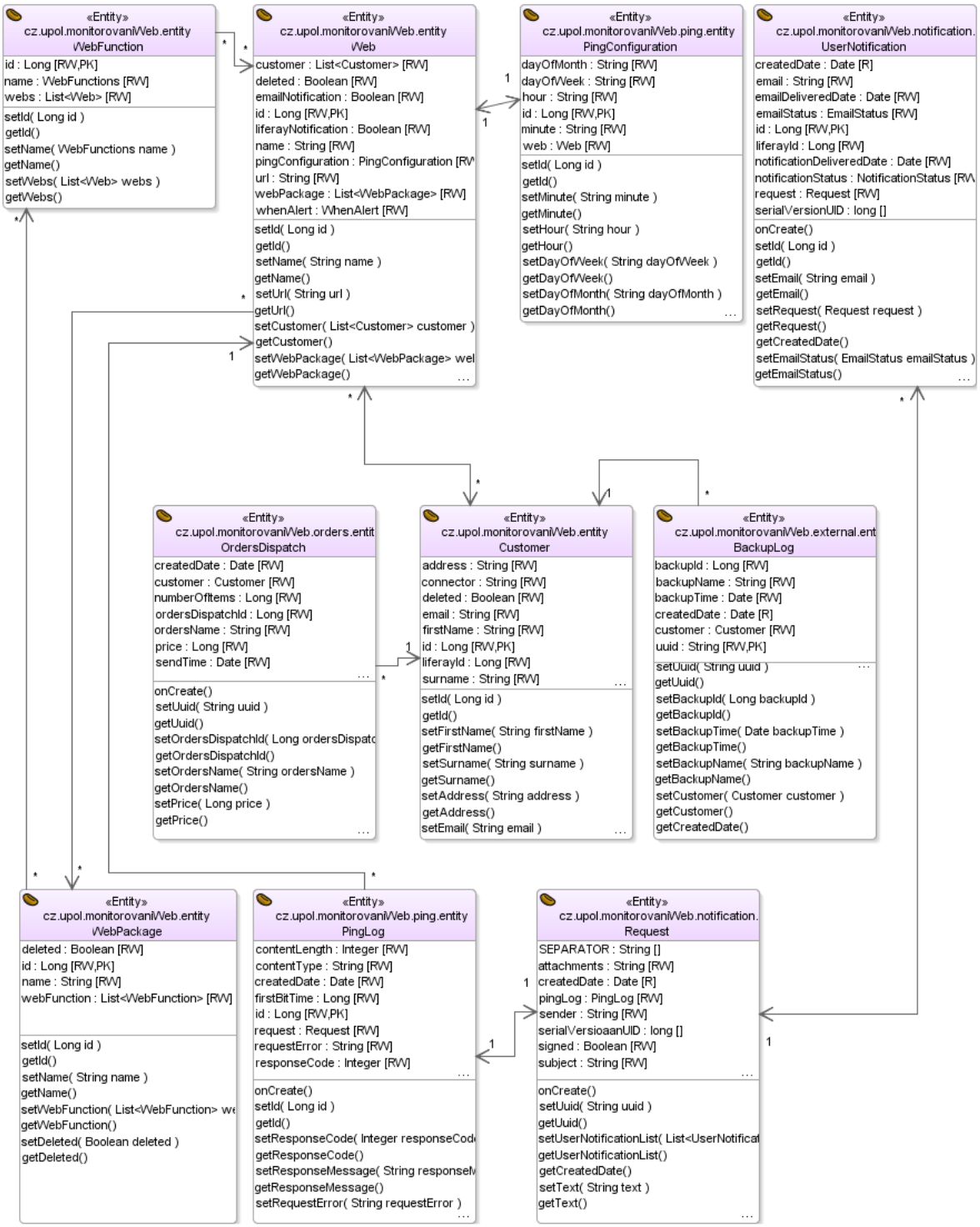
Diagram znázorňuje jednotlivé technologie, které jsou v aplikaci použity.



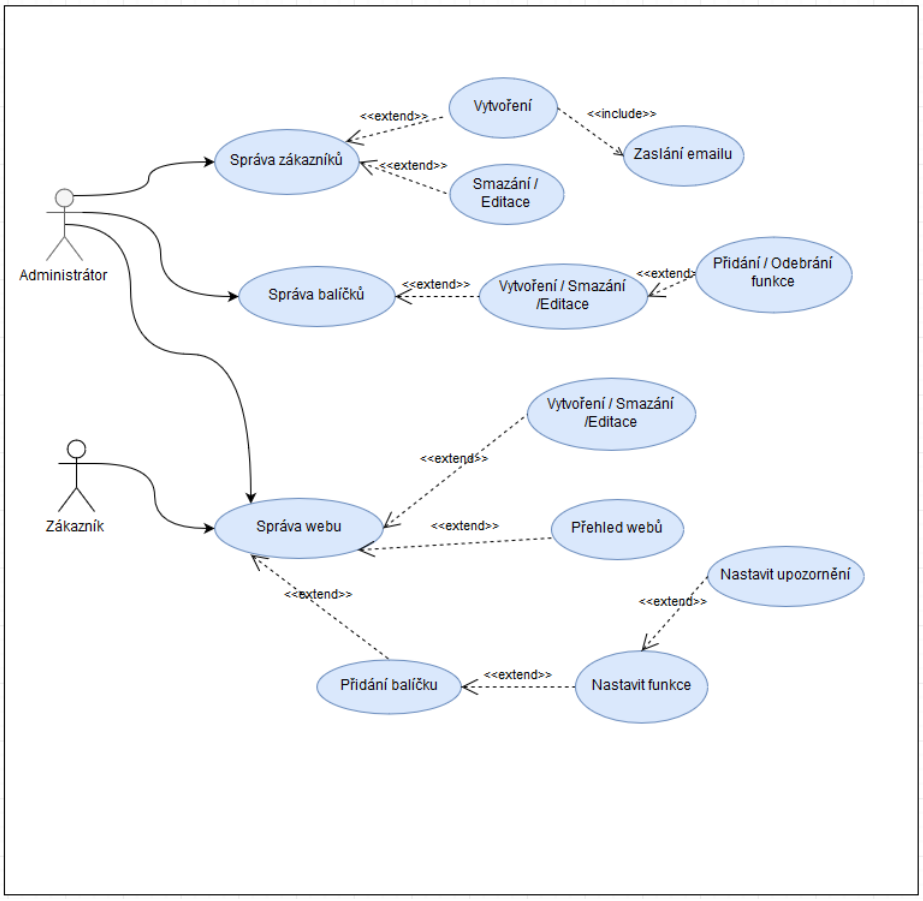
Obrázek 6: Diagram architektury

4.2 Datový model

Obrázek 7: Datový model aplikace



4.3 Případ užití



Obrázek 8: Diagram případu užití

4.4 O aplikaci

Aplikace je rozdělena na dvě hlavní části a to Administrativní a Zákaznickou. Přístupová oprávnění k dané části určuje role v Liferay. Zde jsou použité role Administrátor a Zákazník. Uživateli se po přihlášení zobrazí v menu část aplikace dle přiřazené role.

Administrativní část je pak rozdělena do dvou dalších částí. První část se stará o tvorbu balíčku z předdefinovaných funkcí. Druhá pak zahrnuje administraci uživatelů, přiřazování webů a samozřejmě i přiřazení balíčku k vybranému webu.

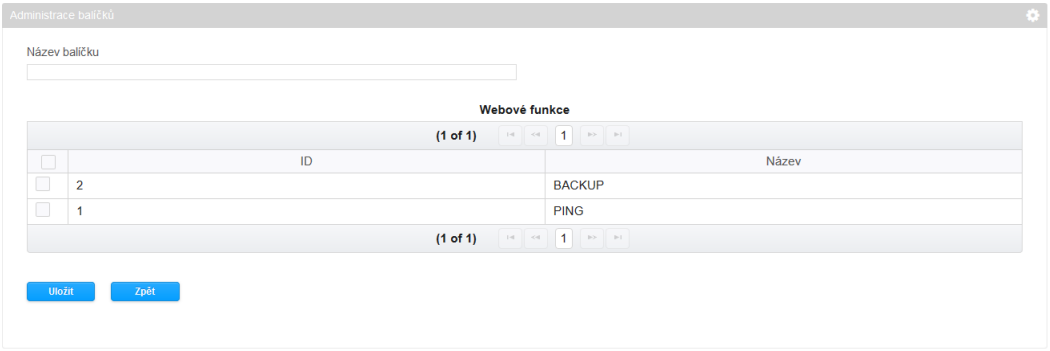
4.5 Administrativní část webových balíčků

Požadované role: Administrátor, Zákazník
Administrativní část webových balíčků slouží k nakombinování a seskupení jednotlivých webových funkcí do jednoho výsledného balíčku. Po vytvoření balíčku je možnost jeho přiřazení k jakémukoli webu. Náhled hlavní obrazovky je vidět na obrázku č. 9. Obrazovka obsahuje seznam již vytvořených balíčku a tlačítko pro vytvoření balíčku nového.

Obrázek 9: Hlavní stránka administrace balíčků

4.5.1 Přidání nového balíčku

V tabulce „Webové balíčky“ na obrázku č. 9 po kliknutí na tlačítko „Nový webový balíček“ je zobrazena stránka pro přidání nového balíčku, což můžeme vidět na obr. 10. Stránka nám umožňuje zadat název, vybrat funkce, které budou obsaženy v novém balíčku a po kliknutí na tlačítko „Uložit“ je nový balíček vytvořen a připraven k použití. Nově vytvořený balíček najdeme v tabulce „Webové balíčky“ viz obr.č. 9



Obrázek 10: Vytvoření nového balíčku

4.6 Administrativní část uživatelů

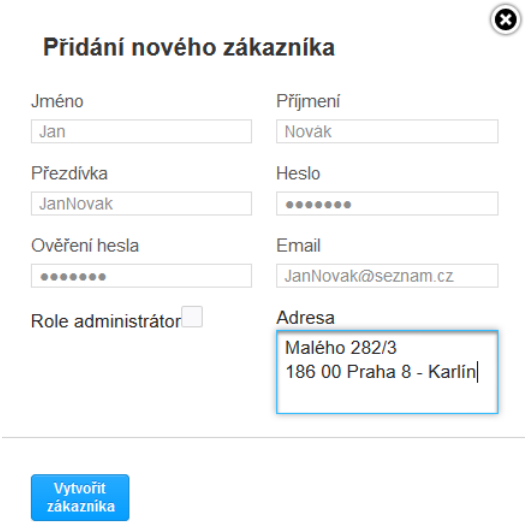
Požadované role: Administrátor

Tato část slouží k vytváření nových uživatelů jak už s rolí Administrátor tak s rolí Zákazník. Dále je zde možnost přidání/odebrání webu a webového balíčku.

Obrázek 11: Hlavní stránka administrace uživatelů

4.6.1 Přidání nového uživatele

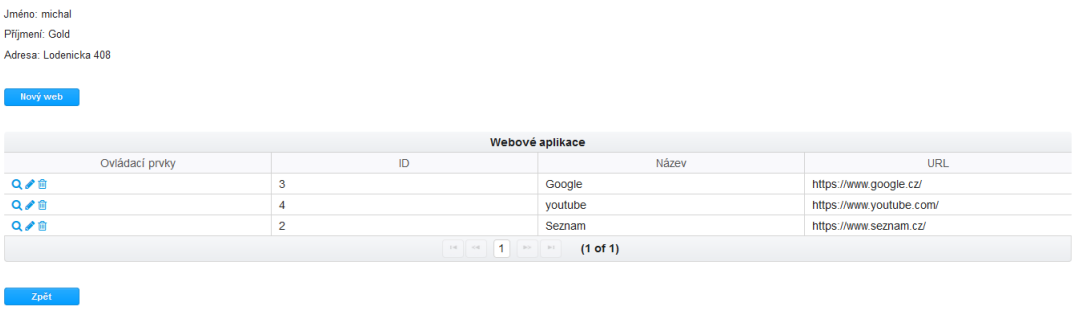
Administrátorovi se po kliknutí na tlačítko „Nový zákazník“ zobrazí dialogové okno (viz obr. 12) a je vyzván k vyplnění údajů o novém uživateli. Zde je možnost vybrat roli, kterou nový uživatel bude mít přiřazenu. Zůstane-li políčko „Administrátor“ odškrtnuté, nově vytvořenému uživateli zůstává role „Zákazník“. Po vyplnění všech údajů a kliknutí na tlačítko „Vytvořit zákazníka“ je nový zákazník vytvořen a přidán do tabulky „Zákazníci“ viz obr. 11



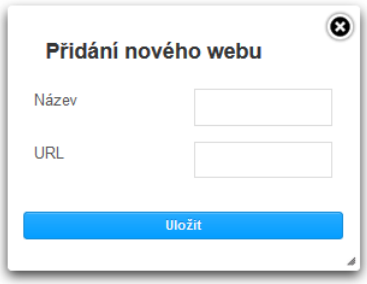
Obrázek 12: Dialog pro vytvoření nového uživatele

4.6.2 Přidání webové aplikace pro uživatele

Na obr. 11 po kliknutí na ikonu „lupy“ je zobrazena tabulka „Webové aplikace“, viz obr. 13, která obsahuje seznam všech aplikací patřící danému zákazníkovi. Nad tabulkou je zobrazeno tlačítko „Nový web“, které zobrazí dialog viz obr. 14. Po vyplnění hodnot a potvrzení kliknutím na tlačítko uložit je nový web vytvořen a přidán do tabulky „Webové aplikace“ na obr. 13.



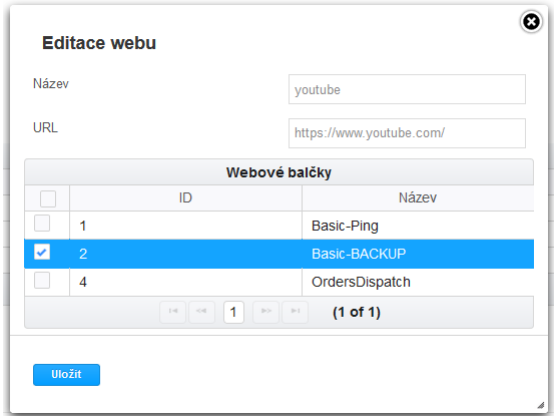
Obrázek 13: Seznam webových aplikací



Obrázek 14: Dialog přidání webu

4.6.3 Přidání webového balíčku k aplikaci

Pokud je u jakéhokoli záznamu v tabulce „Webové aplikace“ na obr. 13 zvolena ikona „tužky“, pak je zobrazen dialog „Editace webu“, viz obr. 15. V dialogu zaškrtneme balíčky, které chceme přidat k vybranému webu a stiskneme tlačítko „Uložit“. Nyní má náš web přiřazeny funkce, které jsou obsaženy ve vybraných balíčcích.



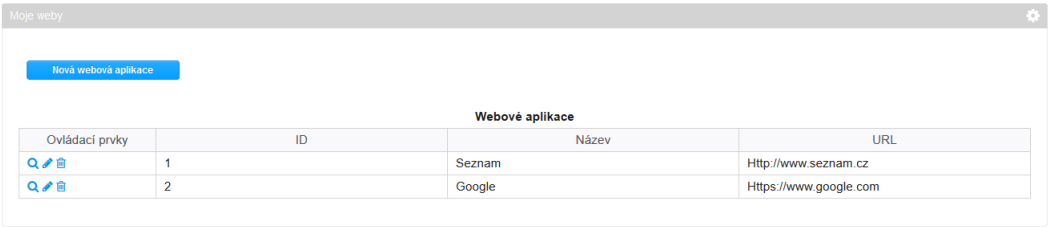
Obrázek 15: Editace webu pro administrátora

4.7 Zákaznická část (Moje weby)

Požadované role: Administrátor, Zákazník.

Zákaznická část slouží k přehledu funkčnosti webů, které má uživatel zaregistrován. Rozsah a podrobnost přehledu, který je danému uživateli dostupný, je závislý na počtu přiřazených funkcí, které obsahují jednotlivé balíčky. Tedy čím více balíčků uživatel využívá, tím rozsáhlejší funkcionalitu má dostupnou. Plně funkcionality balíčků plánuji využít především do budoucna, kdy bude množství monitorovacích funkcí o poznání bohatší a uživatel si tedy bude moci snadno přizpůsobovat výsledný monitoring svým potřebám.

Na hlavní obrazovce je zobrazena tabulka přiřazených webů a možnost přidat web nový, viz obr. 16



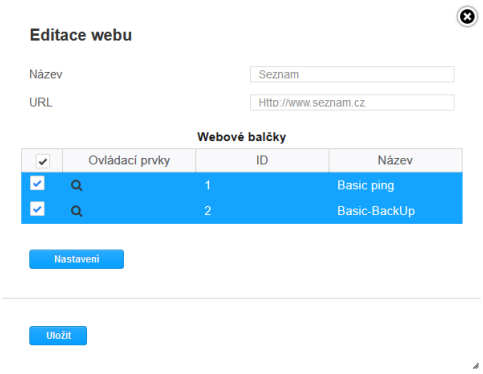
Obrázek 16: Hlavní stránka zákazníka (Moje weby)

4.7.1 Přidání webu

Uživatel zvolí tlačítko „Nový web“ na obrázku 16, následně je zobrazen dialog č. 14, který je identický jako v administrativní části.

4.7.2 Editace a nastavení webu

Editací je zde myšleno přidání webových balíčku, čehož dosáhneme kliknutím v tabulce „Webové aplikace“ na ikonu „tužky“. Po kliknutí je zobrazen dialog „Editace webu“, který je vidět na obrázku č. 16. Dialog vypadá podobně jako v administrativní části, jen navíc obsahuje tlačítko „Nastavení“. Pomocí tlačítka „Nastavení“ se dialog překreslí a zobrazí podrobnější možnosti nastavení k jednotlivým funkcím v daných balíčcích. Momentálně má podrobnější možnosti nastavení pouze funkce „Ping“. Není-li funkce „Ping“ v žádném vybraném balíčku, systém zobrazí informace o absenci funkce.



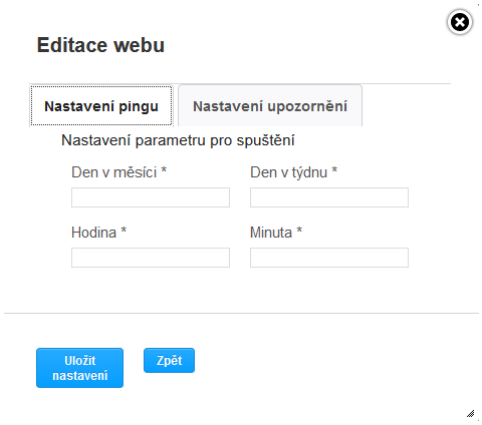
Obrázek 17: Editace webu pro zákazníka

4.7.3 Nastavení funkce Ping

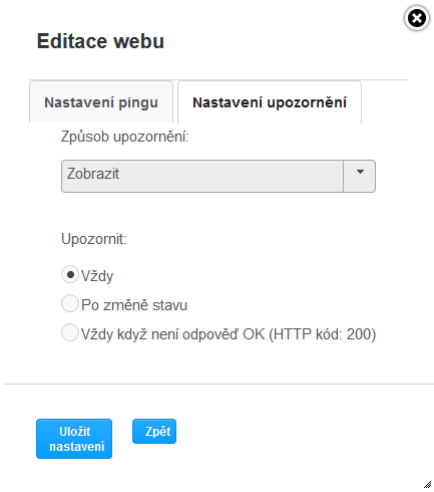
Pro nastavení funkce „Ping“ je jak již bylo uvedeno nutné, aby byla obsažena alespoň v jednom přiřazeném webovém balíčku. K nastavení se dostaneme kliknutím na tlačítko „Nastavení“ na obrázku č. 17. Nyní je zobrazen dialog č. 17 obsahující dvě záložky. Na první záložce „Nastavení pingu“ je možnost nastavit údaje kdy se bude funkce vykonávat (den v měsíci, den v týdnu, hodina a minuta). Ve druhé záložce, kterou vidíme na obrázku č. 17, je možnost nastavení způsobu upozornění (Email, Liferay), a dále nastavení kdy chce být uživatel upozorněn.

1. Vždy: uživatel je upozorněn při každém pingu webu.
2. Po změně stavu: uživatel je upozorněn tehdy, pokud se mezi posledními dvěma pingy změnil návratový kód.
3. Vždy, když není odpověď OK (HTTP kód: 200).

Při základním nastavení je kontrola webu nastavena na každou minutu v měsíci a upozornění je vypnuté.



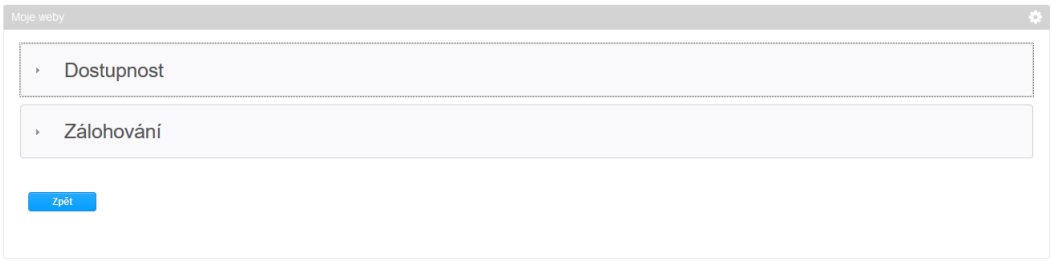
Obrázek 18: Nastavení funkce Ping



Obrázek 19: Nastavení upozornění

4.7.4 Detail webu

Po zvolení „lupy“ u libovolného webu v tabulce „Webové aplikace“ na obrázku č. 16 se zobrazí stránka detailu vybraného webu, kterou můžeme vidět na obrázku č. 20. Na stránce detailu je zobrazen počet záložek v závislosti na tom, kolik má vybraný web přiřazen funkcí. Každá záložka obsahuje jednu konkrétní funkci.

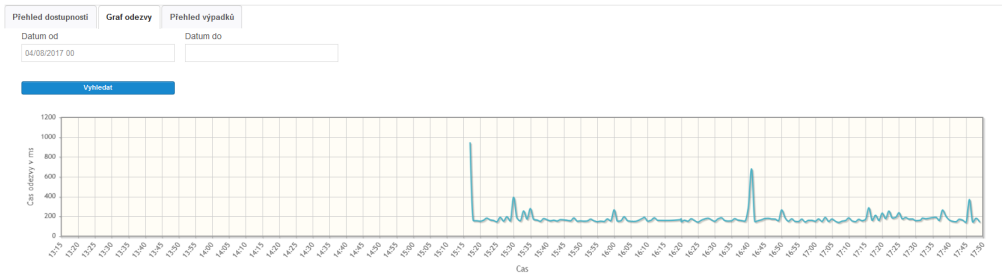


Obrázek 20: Detailní stránka webu

4.7.5 Detail funkce Ping

Má-li uživatel přiřazenou funkci ping, zobrazí se mu v detailu daného webu záložka „Dostupnost“. Záložka je umístěna první v pořadí, tudíž se zobrazí již rozbalená. Na obrázku č. 21 je vidět tabulka naměřených hodnot, které proběhly v pořádku (tedy záznamy, kdy byl web dostupný). Tabulka obsahuje časy dotazů na web, čas prvního příchozího bitu, celkový čas stažení obsahu stránky, typ obsahu a návratový kód. Další možností v tomto přehledu je zobrazení podzáložky „Graf dostupnosti“. Zde je z dostupných naměřených hodnot vykreslen graf. Graf je možno vykreslit pro požadovaný časový úsek zadáním „Datum od“/„Datum do“ a následným vyhledáním viz obrázek č. 22. Pro zjištění nedostupných naměřených hodnot slouží podzáložka „Přehled výpadků“, kde je zobrazena tabulka „Seznam výpadků“.

Obrázek 21: Tabulka přehledu dostupnosti

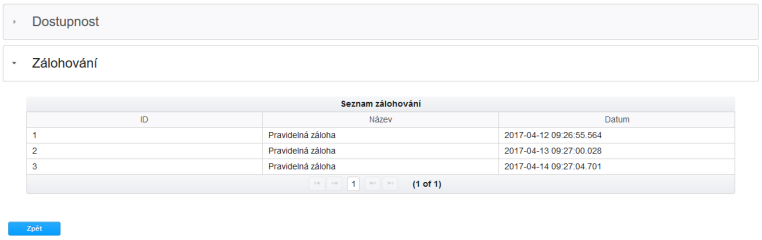


Obrázek 22: Graf přehledu dostupnosti

4.7.6 Detail zálohování

Na obrázku č. 23 je vidět tabulka s přehledem již proběhlých zálohování příslušného webu. Aby zákazník měl tuto tabulku k dispozici, musí jeho web obsahovat balíček s funkcí zálohování. Interval záloh si zvolí každý uživatel libovolně pro jakýkoli web. Libovolné je i zvolení textu ve sloupci „Název“. Vše

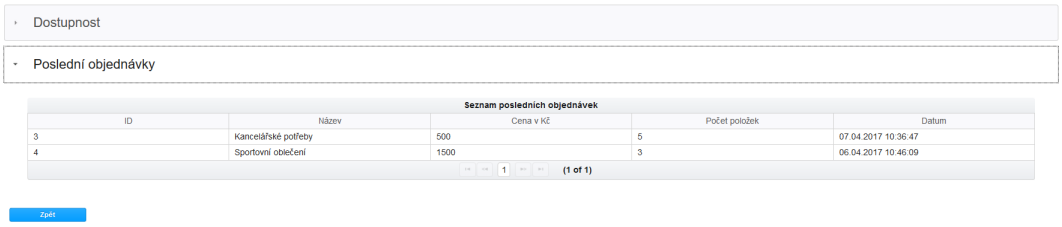
záleží pouze na tom, jak často bude volat webovou funkci, skrze kterou logování zálohování probíhá. Podrobnější popis nastavení a zakomponování popisuji v podkapitole 4.8.6.



Obrázek 23: Seznám proběhlých zálohování

4.7.7 Detail posledních objednávek

Na obrázku č. 24 můžeme vidět seznam posledních objednávek, které určitý web odeslal. Tuto funkcionalitu budou převážně používat jen ty weby, které se zabývají obchodem, například e-shopy. Uživatel bude mít jistotu, že jeho obchod je funkční a přijímá objednávky. Stejně jako u detailu zálohování, i zde komunikace probíhá skrze webovou funkci. Podrobnější popis nastavení a zakomponování popisuje v podkapitole 4.8.7.



Obrázek 24: Přehled posledních objednávek

4.8 Technické řešení

Kapitola je zaměřena na detailnější popis funkcí, které jsou implementovány a použity pro správné fungování aplikace. Z důvodu rozsáhlosti použitých technologií je popis spíše směřován na pochopení principů než na podrobný rozbor implementace.

4.8.1 Přidání nového balíčku

Při přidání nového balíčku je vytvořena nová entita „WebPackage“ a na ni navázané vybrané funkce, tedy entita „WebFunction“. Vztah entit je přidán do spojovací tabulky, kterou Hibernate vytvořil na základě anotace nad atributem „ListWebFunction“ v entitě „WebPackage“. Více informací o Hibernate je popsáno v podkapitole 3.2.1.

Ukázka anotace pro vytvoření spojovací tabulky:

```
@ManyToMany
@JoinTable(name = "WEB_PACKAGE_TO_WEB_FUNCTIONS",
    joinColumns = @JoinColumn(name = "WEB_PACKAGE_ID"),
    inverseJoinColumns = @JoinColumn(name = "WEB_FUNCTION_ID"))
private List<WebFunction> webFunction;
```

4.8.2 Přidání nového uživatele

U nového uživatele je nejprve vytvořen záznam v Liferay databázi a až poté v interní databázi aplikace. U vytváření záznamu v interní databázi je vygenerován konektor, který souvisí s autentizací pří dotazování se přes webovou službu. Propojení interní databáze aplikace s databází Liferay je implementováno přes „liferayId“. Jakmile jsou tyto záznamy vytvořeny, je uživateli zaslán email obsahující konektor a potřebné údaje k přihlášení do aplikace. K přihlášení slouží přezdívka a heslo, které bylo zasláno v emailu. Z důvodu bezpečnosti je vyžadováno při prvním přihlášení heslo změnit. Po přihlášení jsou pomocí „liferayId“ vybrány odpovídající záznamy v databázi aplikace, které jsou následně zobrazeny.

Tuto funkcionalitu zajišťují třídy „CommunicationToDatabaseSessionEJBBean“ a „LiferayCommunicationSessionEJBBean“.

4.8.3 Přidání webového balíčku k aplikaci

V první fázi jsou vytvořeny vazby balíčku na vybraný web. Poté se sjednotí všechny funkce obsaženy v balíčcích, zkontrolují se již existující vazby a pokud některá z přidávaných funkcí ještě vazbu nemá, přidá se. U přidání funkce „Ping“ je vytvořen nový timer, který je automaticky vykonáván každou minutu. Detailnější popis funkce „Ping“ v závislosti na timeru je rozebrán v podkapitole 4.8.5.

Tuto funkcionalitu zajišťuje třída „CommunicationToDatabaseSessionEJBBean“.

4.8.4 Přidání webu

Web může být přidán jak administrátorem, tak zákazníkem.

- Přidání zákazníkem - v prvním kroku je vytvořena nová entita „Web“, poté je nalezen přihlášený uživatel a nově vytvořený „Web“ je přidán do jeho seznamu webů (tedy vytvořena vazba entit „Customer“ a „Web“).
- Přidání administrátorem - administrátor vidí seznam zákazníku, a aby bylo možné přidat nový web, musí kliknout na detail jednoho z nich. Po kliknutí na detail je dočasně uloženo ID zákazníka, které je následně využito na dohledání entity „Customer“ z důvodu přidání vazby k nově vytvořenému webu.

Tuto funkcionalitu zajišťuje třída „CommunicationToDatabaseSessionEJBBean“.

4.8.5 Detail funkce Ping

Veškeré hodnoty jak pro obsahy tabulek, tak vykreslení grafu jsou načteny z databázové tabulky „PingLog“. Tabulka je plněna v závislosti na tom, kdy má jaký web nastaveny parametry pro kontrolu. Při přiřazení funkce „Ping“ k určitému času, je vytvořen javax.ejb.Timer (časovač) a uložena hodnota do databázové tabulky „PingConfiguration“, jejíž důvod existence zdůvodním v následujících větách. Dynamicky vytvářený timer existuje v aplikaci jen po dobu jednoho běhu aplikačního serveru. Jakmile je server vypnut, jsou všechny vytvořené timery z něj odstraněny. Při zapnutí serveru se vezmou záznamy reprezentující nastavení timeru z tabulky „PingConfiguration“ a v závislosti na hodnotách jsou znovu vytvořeny. Je tedy zaručeno, že vždy budou vytvořeny všechny timery, které existovaly při minulém běhu serveru. javax.ejb.Timer umožňuje vytvářet i persistentní timery, které se po vypnutí aplikace nesmažou. Jejich problémem však je, že si pamatují, jak dlouho byly vypnuty a po zapnutí aplikačního serveru zpětně vykonají všechny promeškané zpoždění. Což by i při krátkodobém výpadku serveru, na kterém běží má aplikace, způsobilo zbytečné zaznamenání mnoha hodnot v okamžiku těsně po spuštění aplikačního serveru.

Jakmile se timer aktivuje, začne se vykonávat metoda „regularWebPingCheck(Timer timer)“ která nese anotaci „@Timeout“. Ze vstupního parametru „timer“ je zjištěno ID konfigurace, pomocí kterého se vyhledá entita „PingConfiguration“, která má uloženu vazbu na web. V tomto okamžiku je vytvořena message s ID webu a poslána do JMS fronty, kterou zpracovává MDB. MDB vyzvedne z JMS fronty zaslanou message, zjistí z ní ID a následně vyhledá entitu „Web“, kterou použije jako vstupní parametr do funkce Ping. Samotná funkce Ping poté zkouší otevřít připojení k příslušné URL adrese, přičemž měří čas navázání prvního spojení, celkový čas stažení obsahu stránky, kód odpovědi, odpověď atd. Celá implementace funkce Ping je přiložena na CD.

Funkcionalitu pingu nalezneme v balíčku „cz.upol.monitorovaniWeb.ping“. Čtenáře bych rád odkázal na třídu "PingConfigurationSessionEJBBean", která

se stará o dynamické vytváření timerů z CRONů, které zadává uživatel skrze GUI, a jedná se o implementaci, kterou by mohl využít i na svých projektech.

4.8.6 Detail zálohování

Zálohování je jednou z webových funkcí, kterou aplikace podporuje. Aby zákazník mohl tuto funkci použít, musí jeho web mít jak přiřazený balíček obsahující funkcionalitu „Zálohování“, tak implementovaného klienta pro komunikaci skrze SOAP.
Uvedu příklad vytvoření komunikace s mou aplikací. Zákazník použije WSDL na adrese „Název serveru/monitorovani-web/ExternalSessionEJBBean?wsdl“, například:
http://localhost:8080/monitorovani-web/ExternalSessionEJBBean?wsdl pro vygenerování klienta a zakomponuje kód do své aplikace. Jakmile se zákazník dotáže do mé aplikace, je zkontrolován konektor a další povinné parametry (konektor zákazník obdrží při registraci) a pokud je vše správně zadáno, jsou jím zaslané údaje uloženy do tabulky „BackupLog“. V opačném případně je mu vrácen chybový kód s popisem problému.
Ukázka session beany, která generuje příslušné WSDL, je na obrázku č. 25. Podrobnější popis webových služeb čtenář nalezne v sekci 3.4.4 nebo prostudováním balíčku „cz.upol.monitorovaniWeb.external“.

```
@Stateless(name = "ExternalSessionEJB")
@WebService
@SOAPBinding(style = SOAPBinding.Style.DOCUMENT, parameterStyle = SOAPBinding.ParameterStyle.WRAPPED,
             use = SOAPBinding.Use.LITERAL)
// Cesta k WSDL na localhostu: http://localhost:8080/monitorovani-web/ExternalSessionEJBBean?wsdl
public class ExternalSessionEJBBean implements ExternalSessionEJB, ExternalSessionEJBLocal {

    private static Log Log = LogFactory.getLog(ExternalSessionEJBBean.class);

    @Resource
    SessionContext sessionContext;

    @PersistenceContext(unitName = "monitorovaniWeb")
    private EntityManager em;

    @WebMethod()
    @WebResult(name = "logBackup")
    public String logBackup(@WebParam(name = "connector", partName = "connector") @XmlElement(required = true)
                            String connector,
                            @WebParam(name = "backupId", partName = "backupId") @XmlElement(required = true)
                            Long backupId,
                            @WebParam(name = "webId", partName = "webId") @XmlElement(required = true)
                            Long webId,
                            @WebParam(name = "backupTime", partName = "backupTime") @XmlElement(required = true)
                            Date backupTime,
                            @WebParam(name = "backupName", partName = "backupName") @XmlElement(required = false)
                            String backupName) throws MissingParameters {


        Log.info("Method 'logBackup' called with parameters -> connector: " + connector + ", backupId: " + backupId +
                ", backupTime: " + backupTime + ", backupName: " + backupName);

        if ((connector == null || connector.equals("")) || backupId == null || backupTime == null || webId == null) {
            throw new MissingParameters("Connector, backupId and backupTime are mandatory");
        }
```

Obrázek 25: Náhled Session Beany generující WSDL

4.8.7 Detail posledních objednávek

Aby uživatel mohl tuto funkcionalitu použít, musí kontaktovat server, na kterém aplikace „Monitorování webu“ poběží a odeslat požadované údaje. Aby aplikaci mohl kontaktovat, musí si pomocí WSDL vygenerovat klienta pro komunikaci s mou aplikací. Jedná se o stejný postup jako u funkce zálohování.

5 Rozšiřitelnost

Při vývoji bylo počítáno s tím, že aplikace se bude do budoucna dále rozšiřovat. Ať už z mé vlastní iniciativy, nebo aby bylo vyhověno požadavkům zákazníků. Aplikace je navržena tak, aby byla možnost později vytvářet funkce na míru konkrétního přání zákazníka.
V následující části jsou uvedeny příklady pro přidání nové funkcionality:

Příklad přidání nové funkce ve směru dotazování zákazník -> aplikace. Jsou zapotřebí následující kroky:

1. Vytvořit tabulku pro uložení příchozích dat.
2. Implementovat logiku funkce (komunikace s nově vytvořenou tabulkou).
3. Přidat funkci do seznamu funkcí.
4. Přidat novou záložku do grafické části u detailu webu.
5. Zpřístupnit funkci pomocí WSDL.

Příklad přidání nové funkce ve směru aplikace -> zákazník. Jsou zapotřebí následující kroky:

1. Vytvořit tabulku pro uložení měřených dat.
2. Implementovat logiku funkce (komunikace s nově vytvořenou tabulkou).
3. Přidat funkci do seznamu funkcí.
4. Přidat novou záložku do grafické části u detailu webu.

U obou dvou výše demonstrovaných příkladů je samozřejmě možnost navázání na notifikace, ať už emailové, nebo v rámci portálu Liferay. Stačí pouze přidat volání na již existující metody. Pak už jen záleží na domluvě, zda se přidá možnost zákazníkovi měnit upozornění i v grafické části nebo ne.

Každé odeslané upozornění je uloženo v databázi, takže je možnost přidat přehledovou tabulku zaslaných upozornění.

6 Závěr

Výsledkem této práce je implementována webová aplikace, která konkuruje svou funkcionalitu již existujícím aplikacím zaměřeným na monitorování webů. Aplikace navíc umožňuje uživatelům ukládání svých dat pomocí webových služeb. Uživatel má na jednom místě jak přehled o funkčnosti webů, tak informace o posledních objednávkách nebo přehled proběhlých zálohování. Aplikace je navržena s ohledem na snadnou škálovatelnost a rozšiřitelnost, tudíž je možné vyhovět požadavkům zákazníků bez nutnosti velkého zásahu do zdrojového kódu. Aplikace je také díky využití responzivní technologie snadno ovladatelná i na mobilních zařízeních.

7 Obsah přiloženého CD/DVD

bin/
: Obsahuje soubory monitorovani-web.jar a monitorovani.war pro nasazení aplikace na Aplikační server JBoss.

doc/
: Text práce ve formátu PDF, vytvořený s použitím závazného stylu KI PřF UP v Olomouci pro závěrečné práce, včetně všech příloh, a všechny soubory potřebné pro bezproblémové vygenerování PDF dokumentu textu (v ZIP archivu), tj. zdrojový text textu, vložené obrázky, apod.

src/
: Kompletní zdrojové kódy aplikace.

readme.txt
: Instrukce pro nasazení webové aplikace Monitorování webů na webový server, včetně všech požadavků pro její bezproblémový provoz, a webová adresa, na které je aplikace nasazena pro účel testování při tvorbě posudků práce a pro účel obhajoby práce.

Navíc CD/DVD obsahuje:

client/
: Obsahuje soubory client.jar pro otestování webových služeb a k němu přiložený návodClient.txt s informacemi o použití.

Literatura

[1] WIKIPEDIA - otevřená encyklopedie www.wikipedia.org/wiki/Liferay

[2] PANDA D., RAHMAN R., CUPRAK R. a REMIJAN M. EJB 3 in action. Second edition. ISBN 9781935182993.

[3] Java Platform, Enterprise Edition 7 API Specification. [online]. [cit. 2017-04-15]. Dostupné z: https://docs.oracle.com/javaee/7/api/

[4] ORACLE CORPORATION. Oracle Database Online Documentation 12c Release 1. [online]. [cit. 2017-04-18]. Dostupné z: https://docs.oracle.com/database/121/ RED HAT.

[5] JBoss AS 7.1 Documentation. [online]. [cit. 2017-04-20]. Dostupné z: https://docs.jboss.org/author/display/AS71/Documentation