VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

WEBOVÁ SLUŽBA PRO SPRÁVU VOUCHERŮ

WEB SERVICE FOR ELECTRONIC VOUCHERS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **NATÁLIA JURDOVÁ**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. RADEK KOČÍ, Ph.D.**
SUPERVISOR

BRNO 2017

Zadání bakalářské práce/20115/2016/xjurdo00

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav inteligentních systémů Akademický rok 2016/2017

Zadání bakalářské práce

Řešitel: **Jurdová Natália**
Obor: Informační technologie
Téma: **Webová služba pro správu voucherů**
Web Service for Electronic Vouchers
Kategorie: Web

Pokyny:

1. Prostudujte problematiku tvorby webových služeb, které jsou postaveny nad technologií Java.
2. Analyzujte správu papírových slevových voucherů používanou ve firmě AIS Servis, s.r.o. Identifikujte procesy spojené s vydáváním, aplikací a ověřením voucherů a slabá místa stávajícího řešení.
3. Navrhněte koncept správy elektronických voucherů. Správa bude realizována jako webová služba poskytující všechny potřebné operace včetně statistik využití voucherů.
4. Navržený systém implementujte s využitím Java technologií.
5. Připravte vhodnou testovací sadu a vyhodnoťte splnění požadavků na systém.

Literatura:

- World Wide Web Consortium. Web Services Architecture. https://www.w3.org/TR/ws-arch, dostupné říjen 2016.
- Oracle. The Java EE 6 Tutorial. https://docs.oracle.com/javaee/6/tutorial/doc/javaeetutorial6.pdf, dostupné říjen 2016.

Pro udělení zápočtu za první semestr je požadováno:

- První tři body zadání.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Kočí Radek, Ing., Ph.D.**, UITS FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 17. května 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav inteligentních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Petr Hanáček
vedoucí ústavu

Abstrakt

Táto bakalárska práca popisuje tvorbu webovej služby pre spracovávanie zľavových voucherov vo firme AIS Servis, s.r.o. Webová služba je implementovaná v jazyku Java a poskytuje metódy pre vytváranie, editáciu, mazanie, hľadanie a uplatňovanie zľavových voucherov. Taktiež umožňuje import a export do formátu CSV (z angl. Comma-separated values) a vytváranie štatistík o uplatňovaní zľavových voucherov.

Abstract

This bachelor thesis describes a creation of a web service for managing discount vouchers in the company AIS Servis Ltd. The web service is implemented in the programming language Java and provides methods for creating, editing, deleting, searching and applying discount vouchers. It also allows import and export to the CSV (Comma- separated values) file format and the creation of statistics on the application of discount vouchers.

Kľúčové slová

webová služba, WS, WSDL, SOAP, Java EE, Enterprise Java Bean, EJB, Java Persistence API, JPA, voucher

Keywords

web service, WS, WSDL, SOAP, Java EE, Enterprise Java Bean, EJB, Java Persistence API, JPA, voucher

Citácia

JURDOVÁ, Natália. *Webová služba pro správu voucherů*. Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Radek Kočí, Ph.D.

Webová služba pro správu voucherů

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracovala samostatne pod vedením pána Ing. Radka Kočího, Ph.D. Ďalšie informácie mi poskytli Vojtěch Žáček a Mgr. Martin Kunc z firmy AIS Servis, s.r.o. Uviedla som všetky literárne pramene a publikácie, z ktorých som čerpala.

.....
Natália Jurdová
17. mája 2017

Poďakovanie

Chcela by som sa poďakovať vedúcemu mojej bakalárskej práce Ing. Radkovi Kočímu, Ph.D za odborné rady pri písaní bakalárskej práce. Taktiež by som sa chcela poďakovať Vojtěchovi Žáčkovi a Mgr. Martinovi Kuncovi z firmy AIS Servis, s.r.o. za odbornú spoluprácu. V neposlednom rade ďakujem rodine za podporu popri celom štúdiu.

Obsah

Kapitola 1

Úvod

Ľudia v dnešnej dobe čoraz viac vyhľadávajú zľavy, akcie a kupóny alebo porovnávajú ceny produktov a služieb na rôznych internetových stránkach, aby ušetrili nemalé sumy peňazí. Predajcovia po celom svete sa musia snažiť nalákať klientov práve prostredníctvom týchto zliav. Rovnako je to aj v poisťovníctve.

Cieľom tejto práce je analyzovať a navrhnúť komplexné riešenie pre vytváranie a poskytovanie zliav zamestnancom, prípadne klientom poisťovne. Práca popisuje vytvorenie webovej služby pre správu zľavových voucherov ako náhradu za existujúce riešenia, ktoré nie sú dostačujúce a neposkytujú všetku potrebnú funkcionalitu. Nové riešenie má umožňovať uloženie voucherov, ich typov a vlastností do databázy, aby bolo možné ich ďalej spracovávať. Medzi požadované funkcie patrí vytváranie nových záznamov, editácia, mazanie a vyhľadávanie existujúcich záznamov. Ďalej má poskytovať metódy pre import dát z CSV formátu a export do rôznych požadovaných formátov, zisťovanie platnosti a uplatňovanie voucherov a generovanie štatistík o použitých voucheroch. Pridanou hodnotou je ukladanie všetkých prevádzaných akcií do databázy pre spätnú kontrolu.

Jednotlivé časti práce sú zamerané na vysvetlenie použitých technológií, analýzu existujúcich riešení, návrh nového riešenia a popis implementácie a testovania výslednej služby. Pre úplné porozumenie riešeného problému je v kapitole 2 vysvetlený význam webových služieb a možnosti implementácie týchto služieb, a v kapitole 3 je objasnená problematika tvorby webových služieb a podnikových aplikácií v programovacom jazyku Java. V kapitole 4 je podrobnejšie vysvetlená problematika aktuálneho riešenia a cieľ samotnej práce. Ďalej tiež kapitola popisuje návrh novej služby – výber technológií, databázovú štruktúru a metódy, ktoré poskytuje webová služba. V kapitole 5 je konkrétne popísaná implementácia jednotlivých metód webovej služby, štruktúra aplikácie a postupy a výsledky testovania výslednej služby.

Kapitola 2

Webová služba

Webová služba – WS (z angl. Web Service) je služba, ktorá poskytuje jednému elektronickému zariadeniu možnosť komunikovať s iným prostredníctvom internetu. Nejedná sa o komunikáciu človeka so strojom, ale o komunikáciu medzi dvoma strojmi navzájom. Webová služba pozostáva zo samotnej služby a jej popisu, ktorý zahrňuje informácie o rozhraní služby, implementácii vrátane dátových typov, metadát, kategórie a umiestnenia služby – kde je daná služba vystavená [9].

Ako popisuje World Wide Web Consortium (W3C), webová služba je softvér, ktorý poskytuje jedno zariadenie druhému prostredníctvom internetu, aby mohli vzájomne spolupracovať [1]. Umožňuje komunikovať aplikáciam, ktoré sú spustené na rôznych platformách, pričom rozdiely platforiem môžu byť napríklad v operačnom systéme, procesore, programovacom jazyku alebo použitých knižniciach.

Najviac používané implementácie webových služieb v dnešnej dobe sú založené na technológiách REST (z angl. Representational state transfer) a SOAP(z angl. Simple Object Access Protocol) protokolu. Každý z týchto prístupov má svoje výhody a nevýhody a pri výbere medzi nimi je treba dbať na rôzne aspekty.

Táto kapitola popisuje princípy tvorby webových služieb, dva najpoužívanejšie protokoly a rozdiely medzi nimi a výber technológie pre webovú službu, ktorú popisuje táto práca. V sekcii 2.1 je popísaný protokol REST, výhody a nevýhody jeho použitia, bezpečnosť a použitie. V sekcii 2.2 sú popísané rovnaké informácie o protokole SOAP. Sekcia 2.3 obsahuje rozdiely medzi týmito protokolmi a rady pre výber medzi nimi. V sekcii 2.4 je vysvetlené ktorý protokol a z akého dôvodu bol vybraný pre implementáciu webovej služby pre správu voucherov.

2.1 Representational state transfer

REST je komunikačný protokol, ktorý navrhol v roku 2000 Roy Fielding. Narozdiel od SOAP nie je štandardizovaný, sú preň len formálne popísané postupy používania. Pre prenos dát po sieti používa HTTP (z angl. Hypertext Transfer Protocol) protokol, pričom prenášané dáta sú tvorené pomocou XML (z angl. Extensible Markup Language), JSON (z angl. JavaScript Object Notation) alebo iných formátov. Služba môže byť identifikovaná pomocou URI (z angl. Uniform Resource Identifier) [9].

2.1.1 Výhody a nevýhody

Výhody tohto postupu pri vytváraní webových služieb sú:

- jednoduchá implementácia
- podporuje ACID transakcie
- odpovede môžu byť vrátené v rôznych formátoch (JSON, XML, atď.)

Nevýhody tohto postupu pri vytváraní webových služieb sú:

- použiteľný iba s HTTP protokolom
- nepodporuje autorizáciu a bezpečnosť

2.1.2 Bezpečnosť

Protokol REST zabezpečuje iba bezpečnosť na úrovni prenosu. To je zaistené použitím SSL (z angl. Secure Sockets Layer) pod HTTP, čiže použitím HTTPS protokolu (z angl. Hypertext Transfer Protocol Secure) pre bezpečnú komunikáciu s webovými servermi. Pomocou tohto prístupu je možné overiť klienta, podpísať alebo zašifrovať obsah správy, čo umožňuje zistiť, či sa správa počas prenosu nezmenila a zaisťuje to, že tretia strana nemôže zistiť, čo sa v správe prenáša. Komunikácia je šifrovaná a požaduje sa overenie komunikujúcich strán [7].

2.1.3 Použitie

Obvyklé použitie tohto protokolu je pri vytváraní webových služieb pre sociálne siete, mobilné a četovacie služby. Ide o služby, kde požadujeme malé náklady na implementáciu, nízku pamäťovú náročnosť programu, rýchlu odozvu a nekladieme veľký dôraz na chybovosť bezpečnosť programu.

2.2 Simple Object Access Protocol

SOAP je štandardizovaný protokol založený na XML. Bol navrhnutý v roku 1998 Microsoftom ako náhrada predtým používaného štandardu CORBA (z angl. Common Object Request Broker Architecture), ktorý prenášal dáta pomocou binárnych formátov. Pre prenos dát dokáže používať všetky protokoly. Rozhranie služby je popísané pomocou WSDL (z angl. Web Services Description Language) [9].

2.2.1 Web Services Description Language

WSDL je jazyk, pomocou ktorého je popísané rozhranie webovej služby, princíp volania webovej služby. Je popísaný pomocou XML formátu a poskytuje odpovede na otázky:

- Aké funkcie poskytuje daná webová služba?
- Ako sa má s webovou službou naviazať komunikácia?
- Kde a kým je webová služba poskytovaná?

WSDL súbor definuje formát správ, dátové typy, transportný protokol a je popísaný pomocou XML formátu. Špecifikácia WSDL dokumentu je podrobne popísaná v dokumentácii od W3C [2].

2.2.2 Výhody a nevýhody

Výhody tohto postupu pri vytváraní webových služieb sú:

- použiteľný so všetkými protokolmi (HTTP, FTP, SMTP, atď.)
- súčasťou je autorizácia a bezpečnosť
- rozhranie je kompletne popísané pomocou WSDL

Nevýhody tohto postupu pri vytváraní webových služieb sú:

- zložitejšia implementácia
- odpovede môžu byť vrátené iba vo formáte XML

2.2.3 Bezpečnosť

Protokol SOAP zaisťuje okrem použitia SSL aj tzv. bezpečnosť na úrovni XML nazývanú WS-Security alebo WSS (z angl. Web Services Security). Táto služba zabezpečuje šifrovanie a digitálne podpisovanie XML dokumentov prenášaných cez internet a overenie a autorizáciu prenášaných správ pomocou binárnych alebo XML znakov, čím robí SOAP službu bezpečnejšiu v porovnaní s REST [8].

2.2.4 Použitie

Obvyklé použitie tohto protokolu je pri vytváraní webových služieb pre finančné alebo telekomunikačné služby. Tieto služby vyžadujú väčšiu bezpečnosť a metódy poskytované službou by mali byť čo najmenej chybové. Na druhej strane sa pri implementácii nekladie veľký dôraz na náklady na implementáciu a pamäťovú náročnosť programu.

2.3 REST vs. SOAP

Pri výbere medzi týmito dvomi protokolmi treba brať do úvahy mnoho aspektov. Výber záleží na požiadavkoch klienta a samotnej aplikácie (pre aký účel je určená). Prehľadný popis výhod a nevýhod týchto protokolov môžeme vidieť na obrázku 2.1. Pre každú aplikáciu je výber protokolu iný, keďže niektoré aplikácie dbajú na rýchlosť, iné na bezpečnosť alebo v neposlednom rade na cenu a čas potrebný pre implementáciu webovej služby.

Prehľad prípadov použitia protokolov REST a SOAP môžeme vidieť na obrázku 2.2. Popisuje taktiež hlavné kritériá pre výber medzi protokolmi.

2.4 Výber protokolu

Webová služba pre správu zľavových voucherov je implementovaná pre firmu Kooperativa, a.s., ktorej softvér implementuje firma AIS Servis, s.r.o. Poisťovníctvo patrí medzi dve hlavné časti finančných služieb, pre implementáciu ktorých je lepšie použiť protokol SOAP. Pri výbere protokolu bolo ale nutné analyzovať všetky požiadavky, ktoré má služba spĺňať, aby bol výber správny. Po dôkladnej analýze bol vybraný protokol SOAP.

Webová služba slúži pre prenos súkromných údajov o zľavách, klientoch a zmluvách. Je dôležité aby bola spoľahlivá, bezpečná a obsahovala čo najmenšie množstvo chýb. Služba slúži pre komerčné účely a preto podmienky nízkej ceny, rýchlej implementácie a pamäťovej náročnosti neboli prvoradé pri rozhodovaní medzi protokolmi REST a SOAP.

OMPARISON SUMMARY

Metric	SOAP	REST
Cost (lower)		✓
Effort (lower)		
Server side		✓
Client side	✓	
Lines of code (fewer)	✓	
Execution speed (faster)		✓
Memory (low consumption)		✓
Errors (less)	✓	
Functionality		
Protocol independent	✓	
Complexity (lower)		✓
Efficiency		
Performance (better)		✓
Reliability	✓	
Maintainability		
Server side		✓
Client side	✓	

Obr. 2.1: Porovnanie výhod a nevýhod protokolov REST a SOAP. Zdroj: [9].

ELECTION OF SOAP AND REST PROTOCOLS

	Main criteria	Project example
REST	Greater scalability; Compatibility; Performance; Simplicity; Point-to-point communication model; Limited bandwidth.	Mobile app integration with the information systems; Simple client-server data exchange solutions.
SOAP	Higher security and reliability; Lower number of errors; Asynchronous requests; Distributed computing; Support from other standards (WSDL, WS).	Business information systems (B2B); Banking information systems; Payment systems.

Obr. 2.2: Porovnanie prípadov použitia protokolov REST a SOAP. Zdroj: [9].

Kapitola 3

Java Enterprise edition

Java je objektovo-orientovaný programovací jazyk, ktorý vyvinula firma Sun Microsystems v roku 1995. V súčasnej dobe je jedným z najpoužívanejších programovacích jazykov na svete [5].

Java Platforma je názov pre súbor programov od Sun Microsystems, ktoré umožňujú vývoj a spúšťanie programov v jazyku Java. Patria medzi ne:

- **Java Card** – aplikácie pre platobné a kreditné karty
- **Java ME (Micro Edition)** – aplikácie pre mobilné zariadenia
- **Java SE (Standard Edition)** – aplikácie pre desktopové počítače
- **Java EE (Enterprise Edition)** – podnikové aplikácie a informačné systémy

Všetky Java Platformy sa skladajú z JVM (z angl. Java Virtual Machine) a API (angl. application programming interface – aplikačného programového rozhrania) [5]. JVM je program, ktorý interpretuje medzikód (bajt-kód), ktorý je vytvorený zo zdrojových kódov jazyka Java, na spustiteľný program. API je zbierka funkcií a tried, ktoré môže programátor použiť pre vytvorenie softvérových komponent alebo aplikácií [4]. Táto kapitola popisuje použité technológie pre implementáciu metód, ktoré poskytuje webová služba a samotnej webovej služby v jazyku Java. Sekcia 3.1 popisuje štruktúru aplikácie v Jave EE a rozdelenie aplikácie na jednotlivé vrstvy. Tieto vrstvy sú potom podrobnejšie popísané v ďalších sekciách – klientská vrstva v sekcii 3.2, webová vrstva v sekcii 3.3, biznis vrstva v sekcii 3.4 a databázová vrstva v sekcii 3.5. Posledná sekcia tejto kapitoly popisuje Java EE server a funkcionalitu, ktorú poskytuje 3.6.

3.1 Aplikácia v Java EE

Hlavným cieľom Java EE je poskytnúť vývojárom podnikových aplikácií sadu nástrojov, ktoré im skracujú čas, potrebný pre vývoj aplikácie, znižujú zložitosť a zlepšujú výkon aplikácie [6]. Základom je platforma Java SE, nad ktorou sú definované nástroje pre platformu Java EE. Pomocou XML anotácií vkladaných do zdrojového kódu sú definované špecifické informácie o aplikácii. Java EE poskytuje nástroje pre tvorbu webových služieb, webových aplikácií, biznis logiky a prístupu k databáze.

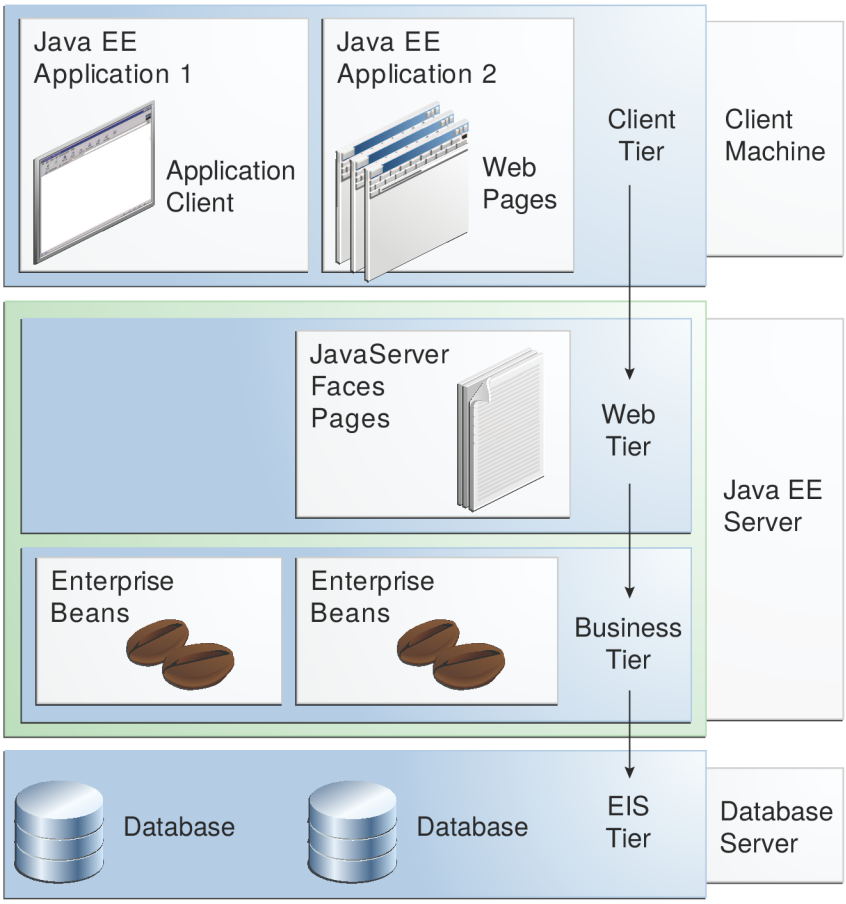
Anotácie sú forma metadát, ktorá poskytuje dodatočné údaje kompilátoru o programe a nemajú priamy vplyv na fungovanie kódu, ktorý anotujú. Anotácia sa skladá zo znaku @,

ktorý kompilátoru indikuje, že nasledujúce slovo je anotácia, a názvu anotácie. Môžu byť použité pri deklarácii tried, premenných, metód alebo iných častí programu [3].

Platforma Java EE používa viacvrstvový distribuovaný model aplikácie, kde je logika aplikácie rozdelená na jednotlivé časti podľa funkcie a komponenty aplikácie sú inštalované na rôznych strojoch podľa prislúchajúcej vrstvy [6]. Model aplikácie sa skladá z týchto častí:

- **klientská vrstva** – beží na klientskom stroji
- **webová vrstva** – beží na Java EE serveri
- **biznis vrstva** – beží na Java EE serveri
- **databázová vrstva** – beží na databázovom serveri

Toto rozdelenie môžeme vidieť na obrázku 3.1. Keďže webová a biznis vrstva bežia na jednom serveri hovoríme o tomto modele, že je trojvrstvový a túto vrstvu voláme aplikačná.



Obr. 3.1: Rozdelenie viacvrstvovej aplikácie v Jave EE na jednotlivé vrstvy. Zdroj: [6].

3.2 Klientská vrstva

Klientská vrstva pozostáva z klientských aplikácií – programov, ktoré bežia na rôznych platformách a ktorých hlavným cieľom je komunikovať s Java EE serverom, ktorý väčšinou beží

na inom stroji. Táto vrstva priamo spolupracuje iba s webovou vrstvou, ktorej posiela svoje požiadavky. Klientmi môžu byť napríklad webové prehliadače, samostatné aplikácie alebo servery, ktoré bežia na inom stroji ako Java EE server [6].

3.3 Webová vrstva

Webová vrstva slúži pre komunikáciu s klientskou vrstvou spracováva požiadavky klientskej vrstvy a pomocou svojich komponent vygeneruje odpoveď, ktorá je poslaná do webového prehliadača klienta. Medzi komponenty webovej vrstvy patria servlety a stránky vytvárané pomocou JavaServer Faces (JSF) alebo JavaServer Pages (JSP). Súčasťou webovej vrstvy sú tiež JavaBeans, ktoré slúžia ako dočasné úložisko dát a zaisťujú komunikáciu s biznis vrstvou. Súčasťou tejto vrstvy je aj technológia pre tvorbu webových služieb [6].

3.3.1 Webové služby

JAX-WS (z angl. Java API for XML Web Services) je technológia programovacieho jazyka Java EE pre vytváranie webových služieb s použitím protokolu SOAP. Špecifikácia definuje, akým spôsobom sú mapované metódy na WSDL, volanie metód a mapovanie SOAP správy na parametre metód. Taktiež popisuje ako je mapovaná návratová hodnota. JAX-WS používa anotácie, ktoré sú súčasťou JWSDP (z angl. Java Web Services Development Pack), čo je SDK (z angl. software development kit) pre vytváranie webových služieb a aplikácií v Jave [6].

3.4 Biznis vrstva

Biznis vrstva tvorí jadro celej aplikácie a zahrňuje všetku logiku a funkcionalitu aplikácie. Biznis vrstva slúži pre spracovávanie požiadavkov od webovej vrstvy, pričom môže pracovať s databázovou vrstvou, a následné posielanie odpovede späť webovej vrstve. Súčasťou tejto vrstvy sú technológie pre tvorbu Enterprise JavaBeans a Java Persistence API.

3.4.1 Enterprise JavaBeans

Enterprise JavaBeans (EJB) je komponent biznis vrstvy na strane Jave EE serveru, ktorá zapúzdruje biznis logiku aplikácie. Cieľom EJB je oddeliť logiku aplikácie od prezentačnej (webovej) vrstvy a od persistentnej (databázovej) vrstvy. EJB poskytuje služby na úrovní systému ako napríklad transakčné spracovanie a autorizáciu vrámci bezpečnosti. Enterprise Beany sa používajú v prípade, keď má byť aplikácia dostupná mnohým klientom alebo keď je potreba zabezpečiť integritu dát. Zahrňujú biznis logiku, ktorá je nadeployovaná (z angl. deploy – nasadiť) na Java EE serveri. Existujú dva druhy Enterprise Beanov – Session a Message-driven. Všetky poznatky v tejto sekcii vychádzajú zo zdroja [6].

Session Bean

Session Bean slúži pre priamu spoluprácu medzi klientom a biznis vrstvou. Poznáme tri druhy Session Beanov – Stateful, Stateless a Singleton. Stateless a Singleton Session Bean môžu implementovať webové služby.

Stateful Session Bean (stavový bean) je naviazaný na konkrétneho klienta. Keď klient vytvorí inštanciu tohto beanu, všetky metódy volané z tohto beanu budú vykonávané vrámci

jednej a tej istej inštancie a stavy premenných sú uchovávané medzi jednotlivými volaniami. Trieda tohto beanu musí byť označená pomocou anotácie `@Stateful`.

Stateless Session Bean (bezstavový bean) nie je viazaný na určitého klienta, ale inštancie tohto beanu sú vytvorené v tzv. poole (z angl. pool – bazén) na Java EE serveri a pri volaní rôznych metód jedným klientom môžu byť použité rôzne inštancie tohto beanu. Nie je možné uchovávať stav premenných medzi jednotlivými volaniami metód, keďže klient nemá zaručené, že pri volaní ďalšej metódy bude použitá tá istá inštancia. Trieda tohto beanu musí byť označená pomocou anotácie `@Stateless`.

Singleton Session Bean (jedináčik) vytvára iba jednu inštanciu vrámci celej aplikácie a slúži pre globálne zdieľanie stavu premenných. Tento bean je zdieľaný medzi všetkých klientov pričom zabezpečuje súbežný prístup k dátam. Trieda tohto beanu musí byť označená pomocou anotácie `@Stateless`.

Message-driven Bean

Message-driven Bean má asynchrónne správanie a slúži k spracovaniu požiadavkov, na ktoré nie je požadovaná okamžitá odpoveď. Bean čaká na správy (obvykle Java Message Service – JMS správy), ktoré môžu byť posielané klientom, iným Enterprise Beanom, webovým komponentom alebo inou aplikáciou používajúcou JMS.

Prístup k Enterprise JavaBeans

Prístup k Enterprise JavaBeans je umožnený dvomi spôsobmi, ktoré závisia na tom, odkiaľ chceme k beanu pristupovať. Pokiaľ pristupujeme k beanu z klienta, ktorý beží na Java EE serveri, čiže je súčasťou webovej alebo biznis vrstvy (iný Enterprise Bean), môžeme použiť priamy prístup k beanu pomocou dependency injection (predávaní závislostí) použitím anotácie `@EJB`. Bean, ku ktorému sa má pristupovať pomocou dependency injection, musí mať implementované lokálne rozhranie (z angl. Local Interface). **Local Interface** je rozhranie zahrňujúce metódy, ktoré poskytuje bean lokálne vrámci aplikačnej vrstvy, teda webovej alebo biznis vrstvy. Trieda tohto rozhrania musí byť označená pomocou anotácie `@Local`.

Klienti, ktorí bežia mimo Java EE serveru, musia použiť pre prístup k beanu JNDI (z angl. Java Naming and Directory Interface) vyhľadávanie. Toto vyhľadávanie umožňuje klientom pomocou menných a adresárových služieb nájsť správnu inštanciu hľadaného beanu. Každý bean bežiaci na Java EE serveri má pridelené jednoznačné meno, podľa ktorého je identifikovaný. Bean, ku ktorému má pristupovať klient, ktorý beží mimo Java EE serveru musí implementovať vzdialené rozhranie (z angl. Remote Interface). **Remote Interface** je rozhranie zahrňujúce metódy, ktoré poskytuje bean vzdialene klientom aplikácie. Trieda tohto rozhrania musí byť označená pomocou anotácie `@Remote`.

3.4.2 Java Persistence API

Java Persistence API (JPA) je špecifikácia pre dočasnú reprezentáciu dát z databáze pomocou tried a metód. Slúži pre získavanie, ukladanie a prácu s dátami v databáze. Hlavným objektom je entita, ktorá reprezentuje tabuľku v databáze a jednotlivé inštancie tejto entity odpovedajú riadkom v tejto tabuľke. Pre prácu s entitami slúži Entity Manager. Pre dopytovanie nad entitami slúži Java Persistence query language (JPQL).

Entita

Entita je trieda, ktorá obsahuje anotáciu `@Entity`. V triede sú deklarované premenné reprezentujúce stĺpce tabuľky a metódy pre nastavovanie a získavanie hodnôt z týchto premenných (tzv. Getters and Setters). Pomocou anotácií sú v triede definované vzťahy s inými tabuľkami (entitami) a taktiež mená stĺpcov tabuľky, na ktoré sa majú naviazať premenné.

Ak je inštancia entity posielaná hodnotou ako samostatne stojaci objekt (napríklad prostredníctvom Remote rozhrania Session Beanu), tak táto entita musí implementovať rozhranie Serializable [6]. Trieda ktorá má byť serializovateľná musí implementovať rozhranie Serializable, obsahovať bezparametrický konštruktor a definovať `serialVersionUID` číslo. Ak je objekt serializovateľný je možné ho konvertovať na prúd bajtov a následne späť na kópiu tohto objektu. Tento prístup slúži hlavne pre správnu kompatibilitu aplikácie medzi rôznymi architektúrami. Objekty môžu byť posielané cez internet alebo ukladané do súborov na disk.

Entity Manager

Entity Manager umožňuje vytvárať a mazať inštancie entity, vyhľadávanie entít pomocou primárneho kľúča a vytváranie dopytov (angl. query) nad entitami. Entity Manager je potreba napojiť na databázu, v ktorej sú tabuľky s ktorými budeme pracovať, pomocou anotácie `@PersistenceContext`, pričom konfigurácia databázovej konekcie je v súbore persistence.xml. Entity Manager poskytuje transakčné spracovanie, kde pri použití Entity Manageru v nejakej metóde spôsobí, že celá táto metóda je v transakcii. Potvrdenie (angl. commit) transakcie je prevádzané automaticky po úspešnom ukončení metódy. V prípade, že metóda spôsobí výnimku, tak je automaticky volané zrušenie (angl. rollback) transakcie.

JPA umožňuje jednoduchšiu prácu s tabuľkami databázy. Pri vyhľadaní entity pomocou Entity Manageru, je táto inštancia entity priamo napojená na konkrétny riadok tabuľky, takže v prípade zmeny niektorého atribútu sa zmení automaticky aj obsah v databáze.

Java Persistence query language

Java Persistence query language (JPQL) je jednoduchý jazyk založený na princípoch SQL (z angl. Structured Query Language) pre dopytovanie nad entitami a vzťahov medzi nimi.

SQL je jazyk pre prácu s dátami v databáze. Rozdeľuje sa na časť pre manipuláciu dát DML (z angl. Data Manipulation Language) a časť pre definíciu dát DDL (angl. Data Definition Language). DML obsahuje príkazy pre výber (SELECT), vkladanie (INSERT), mazanie (DELETE) a editovanie (UPDATE) dát v databázových tabuľkách. DDL obsahuje príkazy pre vytváranie (CREATE), upravovanie (ALTER) a rušenie (DROP) tabuliek v databáze. Ide o jednoduchý jazyk, ktorý nie je zložitý na pochopenie, keďže cieľom jeho tvorcov bolo vytvoriť jazyk, ktorý by sa syntaxou príkazov čo najviac podobal bežnému jazyku (angličtine).

Pri JPQL jazyku je táto syntax veľmi podobná, ale v dopytoch používame mená entít a ich premenných, nie názvy tabuliek ako v SQL. Pri výbere záznamov z tabuľky pomocou jazyku JPQL dostaneme v Jave zoznam objektov Entity, ktorá reprezentuje danú tabuľku.

3.4.3 Data transfer object

Data transfer object (DTO) je jednoduchý objekt, ktorý slúži pre ukladanie dát vrámci aplikácie a neobsahuje žiadnu biznis logiku. V triede definujúcej DTO objekt sú deklarované premenné reprezentujúce dáta objektu a metódy pre nastavovanie a získavanie hodnôt

z týchto premenných (tzv. Getters and Setters). Môže obsahovať metódu pre výpis obsahu objektu (`toString()`). Pre správne používanie objektu medzi jednotlivými vrstvami aplikácie je nutné ho serializovať – implementovať rozhranie Serializable a bezparametrický konštruktor triedy.

3.5 Databázová vrstva

Databázová vrstva – EIS (z angl. Enterprise information system) predstavuje všetky externé systémy, ktoré pre svoj chod používa Java EE aplikácia. Môžu to byť databázové servery prípadne iné dátové zdroje, pričom sú umiestnené na inom stroji ako Java EE server. Komunikáciu s touto vrstvou zaisťuje biznis vrstva.

3.6 Java EE Server

Java EE Server je aplikačný server, ktorý poskytuje API (z angl. Application programming interface – rozhranie pre programovanie aplikácií) Java Platformy, teda prostredie pre spúšťanie štandardných služieb Javy EE. Java EE servery zastrešujú mnoho aplikačných komponentov, ktoré sú potrebné pre chod viacvrstvovej aplikácie a poskytujú ich vo forme kontajnerov [4]. Java EE aplikácia môže byť zložená z viacerých modulov, pričom pri preklade aplikácie sa moduly balia do archívov ktoré sa nasadzujú (z angl. deploy) na Java EE Server.

3.6.1 Java EE kontajner

Java EE kontajnery sú rozhrania medzi komponentmi a funkcionalitou poskytovanou platformou pre tieto komponenty. Funkcionalita kontajneru závisí na platforme a na vybranom komponente. Java EE server zastrešuje tieto kontajnery a umožňuje ich vzájomnú spoluprácu pre správny chod Java EE aplikácie.

Webový kontajner

Webový kontajner je rozhranie medzi webovými komponentmi (servlet, JSF alebo JSP) a webovým serverom. Webový modul obsahujúci webové komponenty je pri preklade zabalený do **WAR** archívu (z angl. web archive), čo je špeciálny JAR (z angl. java archive) s konfiguračnými súbormi pre webový modul. Tento archív je potom nasadený na Java EE server do webového kontajnera.

Klientský kontajner

Kontajnery klienta aplikácie je rozhranie medzi klientom Java EE aplikácie (napr. Java SE aplikácie) a Java EE serverom, ktorý je potrebný pre chod tejto aplikácie.

EJB kontajner

EJB kontajner je rozhranie medzi Enterprise Beanmi poskytujúcimi biznis logiku a Java EE serverom. EJB modul aplikácie obsahujúci Enterprise Beany je pri preklade zabalený do **JAR** archívu a tento archív je nasadený do EJB kontajnera na Java EE serveri.

3.6.2 Archívne súbory

Pri preklade Java EE aplikácie vzniká **EAR** archív (z angl. enterprise archive), ktorý môže obsahovať WAR alebo JAR archív, alebo obidva súčasne. Vytvorenie tohto archívu záleží na moduloch, z ktorých sa skladá Java EE aplikácia. Pokiaľ aplikácia obsahuje webový aj EJB modul, tak vzniknutý EAR archív obsahuje WAR aj JAR archívy. EAR archív obsahuje aj konfigurácie pre zabezpečenie programu, mapovanie EJB a url mapovanie webových modulov.

Kapitola 4

Analýza existujúcich riešení a návrh nového riešenia

Kapitola obsahuje analýzu problému, ktorý rieši táto bakalárska práca. V sekcii 4.1 je popísaný cieľ projektu a v sekcii 4.2 je analýza existujúcich riešení. Sekcia 4.3 charakterizuje navrhnuté databázové tabuľky pre uloženie voucherov a vzťahy medzi nimi a sekcia 4.4 popisuje aké funkcie poskytuje webová služba pre prácu s týmito tabuľkami.

4.1 Cieľ projektu

Cieľom projektu je vytvoriť webovú službu pre spracovávanie zľavových voucherov pre poisťovne Kooperativa, a.s., Vienna Insurance Group a Česká podnikateľská poisťovňa, a.s., Vienna Insurance Group, ktorých softvér vyvíja firma AIS Servis, s.r.o. Služba má nahradiť existujúce riešenia, ktoré nie sú dostačujúce a neposkytujú firme potrebnú funkčnosť.

Zľavové vouchery používajú získatelia poisťovne pre svojich klientov. Získateľ je zamestnanec alebo obchodný partner poisťovne, ktorý poskytuje poistenie klientom poisťovne a vytvára zmluvy o poistení. Klient poisťovne je fyzická alebo právnická osoba, ktorá žiada o poistenie.

4.2 Existujúce riešenia

Aktuálne riešenia poskytujú získateľom možnosť používať zľavové kódy, nie je však riešené ukladanie údajov o použití týchto kódov, ani údajov o samotných kódoch a ich parametroch. Nie je teda možné spätne dohľadať, ktorý získateľ koľko zliav a v akej výške poskytol, ani generovať štatistiky o použití voucherov za nejaké časové obdobie, prípadne umožniť zamestnancom správu zľavových kódov, ich typov a vlastností. Z tohto dôvodu bolo nutné implementovať webovú službu, ktorá by umožňovala ukladanie voucherov do databáze a zastrešovala by všetku potrebnú funkcionalitu pre prácu s nimi.

4.2.1 Zľava na uvážení

Získatelia mali možnosť poskytnúť zľavu svojím klientom, pričom výška tejto zľavy a komu bude poskytnutá bolo iba na uvážení jednotlivých získateľov. Týkalo sa to špeciálnych prípadov (klient mal napríklad preukaz ZŤP) a očakávalo sa, že získatelia budú tieto zľavy poskytovať len v určených prípadoch a nebudú tieto zľavy využívať pre iných – bežných

klientov. Ohľadne týchto zliav neboli prevádzané žiadne dodatočné kontroly, ani sa ich využívanie špeciálne neukladalo. Tým pádom sa nedalo dohľadať a skontrolovať, ktorý získateľ koľko týchto zliav poskytol. Poskytnutá zľava v percentách a v prepočítanej sume bola viditeľná iba na zmluvách klientov, čo bolo takmer nekontrolovateľné.

4.2.2 Špeciálne kódy

Vydávali sa špeciálne vouchery – kódy, ktoré boli programovo kontrolované, pričom výška zľavy pri použití týchto kódov bola na uvážení získateľa a nebola kontrolovaná (kontrolovaný bol iba kód). Tieto kódy vedeli a používali v podstate všetci získatelia. Boli určené pre nalákanie klientov jednotlivým získateľom.

4.2.3 Neštandardný obchodný prípad

Neštandardný obchodný prípad (ďalej len NOP) je prípad, kedy je nutné schváliť danú zmluvu vyššie postavenými pracovníkmi firmy. Tento prípad sa generuje pri použití nadlimitnej zľavy, čiže keď zľava prekročí určitú percentuálnu hranicu. To znamená, že v prípade, kedy získateľ poskytol klientovi neštandardne vysokú zľavu, bol vygenerovaný NOP. Vytváraná zmluva bola zablokovaná a muselo sa počkať, kým príslušný pracovník schváli tento NOP a povolí podpísanie zmluvy. Generovanie NOPov funguje aj naďalej pri používaní elektronickej správy voucherov, pričom sú jasne určené podmienky kedy sa musí generovať, pri akej výške zľavy.

4.3 Ukladanie dát v databáze

Dáta v databáze sú ukladané do tabuliek pre vouchery, uplatnenie voucherov, typy a vlastnosti voucherov a pre logovanie. Konkrétne tabuľky a vzťahy medzi nimi sú popísané v ďalšej časti textu. V prílohe B je ku každej databázovej tabuľke prehľadná tabuľka, popisujúca detaily o stĺpcoch – primárny kľúč, cudzie kľúče, dátové typy, povinné hodnoty a iné.

4.3.1 Tabuľky pre ukladanie dát

V tejto časti sú popísané jednotlivé tabuľky, ktoré je potrebné vytvoriť pre správny chod webovej služby. Ide o tabuľky pre uloženie voucherov, ich typov, vlastností a pre uloženie údajov o uplatnených voucheroch.

Tabuľka knz_voucher

Tabuľka pre reprezentáciu zľavových voucherov. V tabuľke B.1 sú prehľadne popísané stĺpce tabuľky `knz_voucher` a ich vlastnosti.

- **id** – jedinečný identifikátor tabuľky
- **id_knz_voucher_typ** – typ voucheru
- **kod** – kód voucheru, má tvar reťazca AABBBCCCCDDD, pričom AA je číslo 00-99 reprezentujúce zľavu voucheru v percentách (zodpovedá stĺpcu maxsle), BBB je náhodný reťazec písmen, CCCC je náhodný reťazec čísel a DDD je voliteľná časť kódu reprezentujúca špeciálnu vlastnosť voucheru (zodpovedá stĺpcu kod v tabuľke `knz_voucher_vlastnost`). Kód voucheru je unikátny iba pre určité časové obdobie,

ktoré je vymedzené hodnotami plod (začiatok platnosti voucheru) a pldo (koniec platnosti voucheru). Príklady kódov voucherov:

  - 90XXX0001001 – kód reprezentuje voucher so zľavou 90 % so špeciálnou vlastnosťou 001
  - 35XXX001 – kód reprezentuje voucher so zľavou 35 % bez špeciálnej vlastnosti

- **agentura** – agentúra získateľa
- **maxsle** – maximálna možná zľava voucheru v percentách
- **maxsle_kc** – maximálna možná zľava voucheru v českej korune
- **kodzi** – kód získateľa
- **id_uzi_uzivatel** – identifikátor užívateľa
- **plod** – začiatok platnosti voucheru
- **pldo** – koniec platnosti voucheru
- **ins_kdo** – kód operátora, ktorý vložil záznam do tabuľky
- **ins_kdy** – dátum a čas vloženia záznamu do tabuľky

Tabuľka knz_voucher_use

Tabuľka pre reprezentáciu uplatnených zľavových voucherov. V tabuľke B.2 sú prehľadne popísané stĺpce tabuľky `knz_voucher_use` a ich vlastnosti.

- **id** – jedinečný identifikátor tabuľky
- **id_knz_voucher** – voucher, ktorý bol uplatnený
- **use_kdy** – dátum a čas uplatnenia voucheru
- **klic_calc_flat** – zmluva, pre ktorú bol voucher uplatnený
- **sle** – použitá zľava voucheru v percentách
- **sle_kc** – použitá zľava voucheru v českej korune
- **id_uzi_uzivatel** – identifikátor užívateľa
- **vin** – VIN vozidla, pre poistenie ktorého bola uplatnená zľava

Tabuľka knz_voucher_typ

Tabuľka pre reprezentáciu typu voucheru. V tabuľke B.3 sú prehľadne popísané stĺpce tabuľky `knz_voucher_typ` a ich vlastnosti.

- **id** – jedinečný identifikátor tabuľky
- **kod** – kód typu voucheru
- **nazev** – názov typu voucheru

- **popis** – detailnejší popis typu voucheru
- **vlastnost** – kód vlastnosti voucheru, implicitná hodnota je 0
- **jednorazovy** – znak A (áno) alebo N (nie), či je voucher jednorazový

Tabuľka knz_voucher_vlastnost

Tabuľka pre reprezentáciu vlastnosti typu voucheru. V tabuľke B.4 sú prehľadne popísané stĺpce tabuľky `knz_voucher_vlastnost` a ich vlastnosti.

- **id** – jedinečný identifikátor tabuľky
- **kod** – kód vlastnosti voucheru
- **nazev** – názov vlastnosti voucheru

Tabuľka knz_voucher_log

Tabuľka pre logovanie prevádzaných akcií pri práci s vouchermi. V tabuľke B.5 sú prehľadne popísané stĺpce tabuľky `knz_voucher_log` a ich vlastnosti.

- **id** – jedinečný identifikátor tabuľky
- **id_knz_voucher_log_action** – prevádzaná akcia pri logovaní
- **ins_kdo** – kód operátora, ktorý vložil záznam do tabuľky
- **ins_kdy** – dátum a čas vloženia záznamu do tabuľky

Tabuľka knz_voucher_log_action

Tabuľka pre reprezentáciu prevádzaných akcií pri práci s vouchermi. V tabuľke B.6 sú prehľadne popísané stĺpce tabuľky `knz_voucher_log_action` a ich vlastnosti.

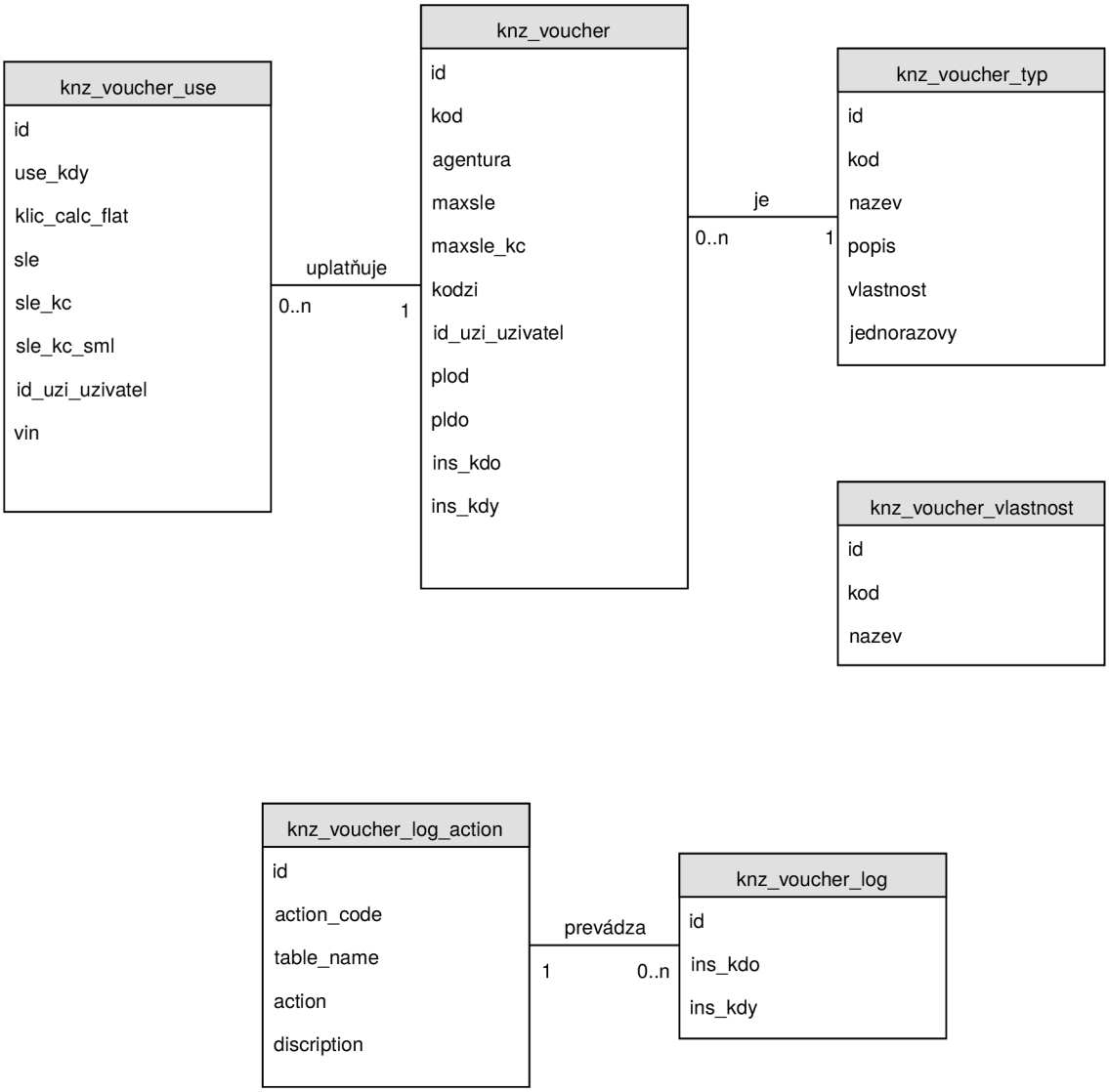
- **id** – jedinečný identifikátor tabuľky
- **action_code** – kód akcie pri logovaní
- **table_name** – meno tabuľky, pre ktorú logujeme akciu
- **action** – názov akcie, ktorá sa loguje
- **discription** – detailnejší popis akcie

4.3.2 Vzťahy medzi tabuľkami

Medzi tabuľkami existujú vzťahy, ktoré ich pomocou cudzích kľúčov spájajú do jedného celku. Každý zľavový voucher v tabuľke `knz_voucher` má priradený typ z tabuľky `knz_voucher_typ` pomocou cudzieho kľúča a jednotlivé typy majú priradenú vlastnosť z tabuľky `knz_voucher_vlastnost` podľa kódu vlastnosti. Pri uplatnení voucheru sa údaje zaznamenajú do tabuľky `knz_voucher_use`, pričom táto tabuľka obsahuje odkaz do tabuľky `knz_voucher`, ktorý reprezentuje konkrétny voucher, ktorý bol uplatnený. Pri každej prevádzanej akcii sa do tabuľky `knz_voucher_log` uloží aká akcia bola nad ktorou tabuľkou

prevedená, pričom v tejto tabuľke je odkaz do tabuľky `knz_voucher_log_action`, ktorá reprezentuje konkrétnu prevedenú akciu.

Na obrázku 4.1 je ER diagram popisujúci štruktúru tabuliek v databáze pre uloženie potrebných údajov o zľavových voucheroch a vzťahy medzi nimi.



Obr. 4.1: ER-diagram popisujúci databázovú štruktúru vytvorených tabuliek pre webovú službu pre správu voucherov.

4.4 Webová služba

Webová služba pre správu voucherov poskytuje metódy pre prácu s databázou (vkladanie, editovanie, mazanie a hľadanie), metódy pre zistenie validity voucheru a pre uplatnenie voucheru, pre import a export voucherov do CSV alebo iných formátov a vytváranie štatistík o uplatňovaní voucherov. Všetky prevádzané akcie sú logované do databázy pre prípadnú kontrolu.

Prehľad týchto metód poskytuje UML graf v prílohe C. Jednotlivé metódy sú podrobne popísané v ďalšej časti textu.

4.4.1 Práca s tabuľkami v databáze

Metódy popísané v tejto časti pracujú s tabuľkami `knz_voucher`, `knz_voucher_typ` a `knz_voucher_vlastnost`. Slúžia pre vkladanie, editáciu, mazanie, hľadanie a načítanie dát z tabuliek. Vkladanie do tabuliek je umožnené po zadaní všetkých povinných hodnôt do vstupného objektu. Pre editovanie záznamov je nutné poznať, ktorý záznam chcem meniť a hodnoty, ktoré majú byť zmenené. Mazanie je umožnené iba v tabuľkách pre typy a vlastnosti a je treba vedieť iba, ktorý záznam chceme zmazať. Hľadanie v tabuľkách je umožnené podľa kódu záznamu, prípadne pri tabuľke pre vouchery aj podľa dátumov, keďže voucherov s rovnakým kódom môže byť v databáze viac. Pre tabuľky s typmi a vlastnosťami sa načítajú vždy všetky hodnoty danej tabuľky. Pre vouchery je možnosť obmedziť toto načítanie podľa dátumu alebo aktuálne prihláseného užívateľa.

4.4.2 Zisťovanie platnosti voucheru a uplatňovanie voucheru

Metódy pre zisťovanie validity voucheru a uplatňovanie voucheru sú najdôležitejšími metódami celej webovej služby. Pracujú s tabuľkami `knz_voucher` a `knz_voucher_use`.

Z tabuľky `knz_voucher` je pomocou kódu voucheru a aktuálneho dátumu zisťované, či je voucher v danom období platný. Ďalšími voliteľnými atribútmi, podľa ktorých môžeme zisťovať validitu voucheru sú agentúra získateľa, kód získateľa alebo identifikačné číslo užívateľa. Pokiaľ tieto údaje nie sú zadané, tak tieto stĺpce v tabuľke neberieme do úvahy. Vrámci tejto metódy je tiež nutné zisťovať, či daný voucher nebol už uplatnený.

Volaniu tejto metódy predchádza volanie metódy pre zisťovanie validity voucheru. Pokiaľ je daný voucher platný, tak môžeme prejsť k jeho uplatneniu. Aplikácia teda počíta s tým, že pokiaľ je volaná táto metóda, tak daný voucher je určite platný a už ďalej jeho platnosť neoveruje. Pri uplatnení voucheru je nutné vyplniť všetky povinné údaje z tabuľky `knz_voucher_use`, čo sú zľava v percentách, zľava v českej korune, kód voucheru, ktorý uplatňujeme, identifikačný kľúč zmluvy, pre ktorú bol tento voucher uplatnený a užívateľ, ktorý túto zmluvu uzavrel.

4.4.3 Import a export voucherov

Import voucherov je umožnený iba z formátu CSV. Táto metóda následne využíva metódu pre vkladanie voucherov do databázy do tabuľky `knz_voucher`. Do iných tabuliek nie je import umožnený, keďže predpokladáme malé množstvo záznamov v týchto tabuľkách.

Pre export bola po zvážení všetkých dostupných prostriedkov využitá funkcionalita pri vytváraní webovej stránky. Vrámci nej je možné exportovať dáta do formátov PDF, CSV a XLS, čo bolo pre účely aplikácie dostačujúce a jednoduchšie, ako vytvárať vlastné riešenie.

4.4.4 Generovanie štatistík o uplatnených voucheroch

Pri generovaní štatistík o využitých voucheroch sa vychádza z tabuľky `knz_voucher_use`. Pre lepšie zobrazenie údajov pre koncového užívateľa sú dodatočne načítané údaje aj z iných tabuliek, napríklad namiesto identifikátora užívateľa, ktorý zľavu poskytol, je zobrazovaný jeho e-mail. Generovanie štatistík je možné ohraničiť pomocou dátumov – od kedy do kedy chceme vygenerovať štatistiku.

4.4.5 Logovanie prevádzaných akcií

Pri všetkých metódach, ktoré poskytuje webová služba, je potreba uložiť informáciu o tejto akcii do databázy pre spätnú kontrolu. Pri každej metóde je teda treba určiť akciu, ktorá bola prevedená a uložiť ju do databázy spolu s prihláseným užívateľom a aktuálnym dátumom.

Kapitola 5

Implementácia

Kapitola vysvetľuje výber technológií, samotnú implementáciu aplikácie a postupy pri testovaní výslednej webovej služby. V sekcii 5.1 je popísaný výber použitých technológií – databázy, programovacieho jazyka, verzovacieho systému a vývojového prostredia. Sekcia 5.2 popisuje postupy pri implementácie jednotlivých častí kódu a rozdelenie aplikácie na moduly. V poslednej sekcii 5.3 sú vysvetlené postupy testovania jednotlivých častí aplikácie.

5.1 Výber technológií

Výber technológií použitých pri implementácii aplikácie záležal v značnej miere na výbere firmy. Ide najmä o výber databázového serveru, programovacieho jazyka, aplikačného serveru, vývojového prostredia, verzovacieho systému a postupoch pri implementácii a testovaní výslednej aplikácie.

5.1.1 Databázový server

Ako databázový server je použitý server Informix od firmy IBM, ktorý je jedným z najrozšírenejších databázových serverov na svete. Vo firme je používaný dlhodobo a poskytuje všetku potrebnú funkcionalitu, ktorú firma od databázového serveru požaduje.

5.1.2 Programovací jazyk

Programovací jazyk, v ktorom je písaná aplikácia, je jazyk Java – konkrétne platforma Java EE pre programovanie podnikových aplikácií. Charakteristiku tohto jazyku a postup pri implementácii popisuje kapitola 3. Výber tohto jazyku bol súčasťou bodu 4. v zadaní bakalárskej práce, ale vyplýval predovšetkým z výberu programovacieho jazyka vo firme AIS Servis s.r.o, pre ktorú je táto aplikácia implementovaná. Hoci je webová služba implementovaná ako samostatný projekt a mohla by byť písaná v akomkoľvek inom jazyku, bol napriek tomu vybraný jazyk Java. Bolo to najmä preto, aby bolo v budúcnosti jednoduchšie vyvíjať ďalšie potrebné časti služby a udržovať webovú službu aktuálnu, keďže všetci programátori vo firme programujú v jazyku Java.

Pre písanie aplikácie bolo použité vývojové prostredie Netbeans a verzovací systém Subversion.

Architektúra aplikácie

Rozdelenie aplikácie na vrstvy a následné nasadzovanie vytvorených archívov na server sa mierne odlišuje od riešenia na obrázku 3.1 z kapitoly 3. Aplikácia je delená na štyri vrstvy ako je na obrázku, rozdiel je ale v tom, že webová a biznis vrstva nebežia na jednom spoločnom Java EE serveri. Pri implementácii a testovaní aplikácie bol ako aplikačný server použitý server Wildfly od firmy Red Hat, ktorý poskytuje implementácie všetkých potrebných štandardov Javy EE. Webová vrstva bola nasadzovaná na Tomcat od firmy Apache Software Foundation (ASF), ktorý nie je serverom, ale iba webovým kontajnerom. Tomcat však poskytoval všetku potrebnú funkcionalitu pre spúšťanie modulov webovej vrstvy aplikácie. Toto rozdelenie bolo vo firme vytvorené v minulosti a používa sa dodnes z dôvodu bezpečnosti a reálneho fyzického uloženia serverov vo firme.

5.2 Implementácia aplikácie

Vytvorená aplikácia je rozdelená na šesť modulov. Nasledujúce sekcie popisujú jednotlivé moduly, ich úlohu vrámci aplikácie, jednotlivé balíky a triedy s metódami, ktoré sa v nich nachádzajú. Modul `voucher-ear` slúži len pre vytvorenie archívu EAR pre nasadenie na aplikačný server a nie sú v ňom žiadne implementačné časti, preto nebude v ďalších sekciách popisovaný.

5.2.1 Modul voucher-cmd-client

Modul `voucher-cmd-client` je vytvorený pre testovanie metód v Enterprise Beanoch prostredníctvom vzdialeného rozhrania `VoucherBeanRemote`. Funkcia tohto modulu je popísaná v ďalšej sekcii 5.3.2.

5.2.2 Modul voucher-ejb

Modul `voucher-ejb` obsahuje implementácie Enterprise Benov – `VoucherBean` pre vzdialené rozhranie a `VoucherBeanDb` pre lokálne rozhranie a prácu s databázou, lokálne rozhranie `VoucherBeanDbLocal`, JPA objekty a pomocnú triedu `CsvLineParser` pre spracovanie jedného riadku CSV súboru. Nachádza sa tu tiež bean pre logovanie prevádzaných akcií `LogBean` s lokálnym rozhraním `LogBeanLocal`. Všetky beany, ktoré sú implementované v tejto aplikácii sú bezstavové (z angl. Stateless) beany (viď. 3.4.1). Ďalšie podsekcie podrobnejšie vysvetlia funkciu a postup implementácie týchto tried a rozhraní.

Trieda VoucherBean

Trieda `VoucherBean` reprezentuje Enterprise Bean pre vzdialené rozhranie `VoucherBean Remote`, ktoré sa nachádza v module `voucher-ejb-client`. Metódy v tejto triede slúžia ako akási medzivrstva pri volaní metód z lokálneho beanu `VoucherBeanDb`, prípadne z beanu `LogBean` alebo triedy `Validator`. Toto rozdelenie bolo vytvorené z dôvodu bezpečnosti, kde bola snaha oddeliť jednotlivé vrstvy a umožniť priamu spoluprácu len dvom susedným vrstvám, teda neumožniť priame napojenie tejto triedy na databázu.

Každá metóda v triede najprv zavolá potrebné validačné metódy z triedy `Validator` pre kontrolu vstupných parametrov. Následne volá metódu z triedy `VoucherBeanDb`, ktorá vykoná potrebné operácie, pričom každá metóda triedy `VoucherBean` má dvojicu v triede

`VoucherBeanDb`, ktorú vždy vo svojom tele volá. Na konci každej metódy sa zavolá metóda z triedy `LogBean`, ktorá zaznamená do databázy, aká akcia bola prevedená.

Trieda VoucherBeanDb a rozhranie VoucherBeanDbLocal

Trieda `VoucherBeanDb` je najdôležitejšou časťou celej aplikácie. Obsahuje implementácie všetkých metód, ktoré volá webová služba cez vzdialené rozhranie `VoucherBeanDbRemote` a vrámci ktorých pracuje s dátami v databáze.

Metódy pre vkladanie záznamov do tabuliek najprv premapujú DTO objekty na JPA pomocou metód v triede `BeanMapper`, následne skontrolujú, či už sa v tabuľke nenachádza záznam s daným kódom, aby nedošlo k duplicitnému vkladaniu a potom prevedú dodatočné kontroly, ktoré nemohli byť vykonané triedou `Validator`. Nakoniec doplnia do JPA objektu potrebné údaje ako je napríklad dátum a čas vkladania a uložia záznam do tabuľky pomocou EntityManageru.

Pri editácii záznamov v tabuľkách sa vyhľadá príslušný záznam podľa identifikátoru, pričom sa tento záznam uloží do JPA objektu. Následne sú menené údaje v tomto objekte podľa vyplnených dát v DTO. Na konci sú zmeny potvrdené EntityManagerom.

Mazanie záznamov je vykonávané pomocou EntityManageru, kde sa záznam vyhľadá podľa identifikátoru, namapuje na JPA a zmaže pomocou metódy `remove()`.

Pre vyhľadávanie a načítanie dát z databázy sú vytvárané dopyty v jazyku JPQL a tie sú potom vykonávané pomocou funkcie `createQuery()` EntityManageru. Táto vracia objekt alebo list JPA objektov, ktoré sa premapujú na DTO a sú posielané ako návratová hodnota metód.

Na princípe vytvárania SQL dopytov pracujú aj metódy pre zisťovanie platnosti, uplatňovanie voucherov a generovanie štatistík. Platnosť voucheru sa zisťuje podľa jeho kódu a aktuálneho dátumu. Pre uplatnenie voucheru je potreba pridať okrem kódu aj ďalšie údaje ako je výška zľavy alebo číslo zmluvy, pre ktorý bol uplatnený. Štatistiky sa generujú pre určité časové obdobie, ktoré je dané vstupnými parametrami.

Metóda na importovanie voucherov z CSV formátu využíva triedu `CsvLineParser` a metódy v vkladanie alebo editovanie voucherov, ktoré sú implementované v tejto triede. V návratovom objekte metódy sú zaznamenané údaje o počtoch úspešne importovaných a neimportovaných voucheroch, dôvody prečo neboli importované, prípadne ďalšie dodatočné informácie.

Trieda LogBean a rozhranie LogBeanLocal

Trieda `LogBean` implementuje lokálne rozhranie `LogBeanLocal` a obsahuje metódy pre uľahčenie logovania pomocou Log4j nástroju od spoločnosti Apache. Tieto metódy slúžia pre zaznamenávanie údajov o volaných metódach, vstupných parametroch a sql dopytoch do logovacieho súboru na serveri Wildfly. Trieda tiež obsahuje metódu, ktorá slúži na vytváranie záznamu o prevádzaných akciách, ktorý je ukladaný do databázy do tabuľky `knz_voucher_log`. Pri vytváraní záznamu je používaná trieda `KnzVoucherLogActionEnum`, z ktorej sa vyberá aká akcia bola vykonaná a v prípade, že táto akcia neexistuje v tabuľke `knz_voucher_log_action`, tak je do nej pridaná, pričom nové hodnoty stĺpcov tohto záznamu sú použité z daného enumu.

Trieda CsvLineParser

Trieda je implementáciou konečného automatu, ktorý bol vytvorený pre spracovanie jedného riadku CSV súboru. CSV je súborový formát, ktorý slúži pre výmenu tabuľkových dát. Jednotlivé záznamy z tabuliek sú reprezentované jedným riadkom a stĺpce sú oddelené špeciálnym znakom – oddeľovačom, ktorý môže byť napríklad čiarka, zvislica alebo bodkočiarka. Jednotlivé údaje môžu, ale nemusia, byť uzatvorené do úvodzoviek. Trieda počíta so všetkými spomenutými variantami a vracia zoznam údajov z jedného riadku CSV súboru.

Balík pre JPA objekty

Balík `cz.aisservis.knz.ejb.app.voucher.jpa` obsahuje triedy, v ktorých sú implementované JPA objekty reprezentujúce tabuľky v databáze. Triedy boli vygenerované z databázových tabuliek generátorom, ktorý bol vytvorený vo firme a ktorý firma na tento účel používa.

5.2.3 Modul voucher-ejb-client

V module `voucher-ejb-client` sa nachádza implementácia vzdialeného rozhrania `VoucherBeanRemote`, trieda `Validator` pre kontrolu vstupných dát a balíky pre DTO objekty, enumy, výnimky a pomocné triedy. V ďalších podsekciách budú tieto časti podrobnejšie popísané. V tomto module sa nachádza aj trieda s JUnit testami, ktorých implementácia je popísaná v ďalšej sekcii 5.3.1.

Trieda VoucherBeanRemote

Trieda `VoucherBeanRemote` popisuje vzdialené rozhranie, ktoré implementuje trieda `VoucherBean`. V triede sú metódy, ktoré pre svoj správny beh potrebuje webová služba a ktorých implementácia sa nachádza v module `voucher-ejb`. Konkrétny popis týchto metód je v sekcii 5.2.2.

Trieda Validator

Trieda `Validator` slúži pre kontrolu vstupných dát, ktoré sú posielané cez vzdialené rozhranie do Enterprise Beanov. Kontroly prebiehajú na všetkých DTO objektoch, ktoré sú používané ako vstupné parametre metód. Metódy zisťujú, či vstupné dáta nie sú prázdne (null) a v prípadne vyplnených dát, či nadobúdajú správne hodnoty. Ide teda napríklad o kontrolu kladnosti číselných hodnôt alebo prázdnosti zadaných reťazcov. Validátor tiež kontroluje niektoré logické chyby ako napríklad, že hodnota koncového dátumu musí byť vyššia ako začiatočného.

Balíky pre DTO objekty

V balíku `cz.aisservis.knz.ejb.app.voucher.dto` sa nachádzajú DTO objekty, ktoré boli vytvorené rovnakým generátorom ako JPA objekty a obsahujú zhodné premenné a typy ako JPA objekty. Slúžia na zobrazenie záznamov z databáze mimo biznis vrstvu aplikácie. Balík `cz.aisservis.knz.ejb.app.voucher.dto.ws` obsahuje pomocné DTO objekty, ktoré boli implementované ručne a slúžia pre posielanie hodnôt z biznis vrstvy do webovej služby. Tieto objekty boli vytvorené v prípadoch, kedy metóda vracia hodnoty v inom formáte, aký poskytujú generované objekty.

Balíky pre enumy

Balík `cz.aisservis.knz.ejb.app.voucher.enums` obsahuje triedu pre uloženie typov výnimiek `ExceptionEnum` a triedu pre uloženie akcií, ktoré sa logujú do databázy `KnzVoucher LogActionEnum`. V tomto enume sú definované všetky premenné, ktoré v databáze odpovedajú stĺpcom tabuľky `knz_voucher_log_action`. V prípade nenájdenia akcie v databáze podľa kódu, je táto akcia vložená do databázy podľa nastavených hodnôt v enume.

Balíky pre výnimky

Balík pre výnimky `cz.aisservis.knz.ejb.app.voucher.exceptions` obsahuje dve triedy pre výnimky, ktoré sú analogické. Výnimka `VoucherRollbackException` narozdiel od výnimky `VoucherException` umožňuje automatický rollback vrámci transakcie, v ktorej bola vyvolaná táto výnimka – vracia databázu do pôvodného stavu, v akom bola pred volaním príslušnej metódy.

Balíky pre pomocné triedy

Balík `cz.aisservis.knz.ejb.app.voucher.utils` obsahuje dve pomocné triedy, ktoré boli prebraté ako existujúce riešenia, ktoré sa používali vo firme. Trieda `BeanMapper` slúži pre mapovanie JPA objektov na DTO objekty, prípadne naopak. Premenné v týchto objektoch musia mať rovnaké názvy a typy. Pri mapovaní dochádza ku kopírovaniu hodnôt premenných z jedného objektu do premenných druhého objektu. Trieda tiež umožňuje mapovať celé zoznamy týchto objektov na zoznamy druhých objektov. Trieda `JNDIProvider` poskytuje metódy pre prístup k rozhraniam Enterprise Beanov v aplikácii.

5.2.4 Modul voucher-web

Modul `voucher-web` je pomocný modul pre vytvorenie webovej stránky pre reprezentáciu funkčnosti aplikácie. Tento modul vytvára WAR archív, ktorý je nasadzovaný na webový kontajner. Pri vytváraní stránky bol použitý framework PrimeFaces [1]. Hlavnými časťami ktoré boli implementované sú trieda `VoucherBean` a dokument home.xhtml pre tvorbu samotnej webovej stránky.

Stránka umožňuje zobrazovať dáta z tabuliek a prácu s týmito dátami pomocou volania metód z biznis vrstvy. Ide o metódy pre načítanie dát (samotné vytvorenie tabuliek), vkladanie, editovanie, mazanie a hľadanie v tabuľkách `knz_voucher`, `knz_voucher_typ` a `knz_voucher_vlastnost`. Taktiež poskytuje možnosť zistiť platnosť voucheru, uplatniť zadaný voucher a generovať štatistiku o uplatnených voucheroch.

Tento modul bol vytvorený hlavne z dôvodu vysvetlenia implementácie importu a exportu do tabuľky `knz_voucher`. Vďaka možnostiam, ktoré poskytoval framework Primefaces, bolo rozhodnuté, že export bude vytváraný na strane webových stránok zo zobrazovaných dátových tabuliek a webová služba bude poskytovať iba metódy pre načítanie dát do týchto tabuliek. Framework umožňuje exportovať do formátov PDF, CSV a XLS. Pre import ale framework neposkytoval dostačujúce riešenia, preto bol import implementovaný vrámci metód biznis vrstvy.

[1] https://www.primefaces.org/

5.2.5 Modul voucher-ws

Modul `voucher-ws` slúži pre implementáciu samotnej webovej služby a obsahuje všetky potrebné triedy pre jej chod. Modul vytvára WAR archív, ktorý je nasadzovaný na webový kontajner. Obsahuje triedu `VoucherService`, balíky pre DTO objekty webovej služby, enumy a výnimky.

Trieda VoucherService

Trieda `VoucherService` obsahuje implementácie metód webovej služby. V triede sú použité anotácie pre definovanie webovej služby (`@WebService`), metód služby (`@WebMethod`) a parametrov služby (`@WebParam`), ktoré umožňujú správne vytvorenie WSDL súboru, ktorý popisuje danú webovú službu.

Vrámci implementácie služby bola snaha o vytvorenie čo najmenšieho počtu metód, ktoré by poskytovala a ktoré by spĺňali požadovanú funkcionalitu. Preto boli niektoré metódy z biznis vrstvy spojené do jednej a vrámci webovej služby sa podľa pridaných parametrov rozhoduje, ktorá metóda z biznis vrstvy má byť volaná. Príkladom takéhoto spojenia sú metódy pre vkladanie a editovanie údajov v tabuľkách, kde sa vstupné objekty líšili iba v naplnení premennej id (identifikátor). Webová služba potom poskytuje iba jednu metódu a na základe vyplnenia tohto parametru je volaná príslušná metóda z biznis vrstvy. Ďalšími príkladmi spájania metód sú metódy pre načítanie a vyhľadávanie záznamov v tabuľkách, medzi ktorými sa rozlišuje podľa zadania kódu záznamu.

Balíky pre DTO objekty

Balík obsahuje DTO objekty, ktoré slúžia pre posielanie hodnôt z alebo do webovej služby. Tieto objekty boli implementované ručne.

Balíky pre enumy

Balík obsahuje triedu `ServiceExceptionEnum`, v ktorej sú uložené typy návratových hodnôt pri volaní metódy pre zisťovanie validity voucheru. Premennú tohto typu obsahuje objekt `ValidVoucherResponseDTO`.

Balíky pre výnimky

Balík obsahuje triedu `ServiceException`, v ktorej je implementovaná výnimka pre webová službu. Táto výnimka musí obsahovať anotáciu `@WebFault`.

5.3 Testovanie

Testovanie výslednej aplikácie prebiehalo viacerými spôsobmi a vo viacerých fázach. Počas vývoja aplikácie bolo nutné tieto testy niekoľkokrát opakovať, aby bola výsledná webová služba bezchybná a pracovala správne.

5.3.1 JUnit testy

Jedným zo spôsobov testovania aplikácie bolo využitie JUnit testov. JUnit je jednoduchý framework pre opakované automatické spúšťanie testov pre overovanie správneho fungovania metód programu. Ako bolo spomenuté v predchádzajúcej sekcii, pre overovanie správ-

nosti vstupných dát, ktoré boli posielané do metód webovej služby, bola implementovaná trieda `Validator` v module `voucher-ejb-client`. Funkciou JUnit testov v aplikácii bolo práve otestovať funkčnosť metód v tejto triede. JUnit testy boli implementované v triede `ValidatorTest`. Testy sú zamerané na kontrolu vstupných dát, pričom sú rozdelené na skupiny a každá skupina testov kontroluje jednu metódu v triede `Validator`. Vrámci skupiny jeden test vždy kontroluje možnosť, kedy sú všetky vstupné dáta správne zadané a ostatné kontrolujú všetky možné kombinácie vstupu s chybnými hodnotami.

5.3.2 Testovanie metód v Enterprise Beanoch

Trieda `VoucherBean` implementuje vzdialené rozhranie `VoucherBeanRemote` a vrámci metód tejto triedy sú volané metódy pre kontrolu vstupných dát z triedy `Validator` a metódy z triedy `VoucherBeanDb`, ktorý pracuje s databázou. Táto časť testovania sa týka triedy `VoucherBeanDb` v module `voucher-ejb`, ale tieto metódy musia byť volané cez vzdialené rozhranie `VoucherBeanRemote`.

Testy sú implementované v module `voucher-cmd-client` a kontrolujú správne fungovanie metód v triede `VoucherBeanDb`, ktoré manipulujú s dátami v databáze. Táto časť testovania nie je automatická a záleží na osobe, ktorá testy prevádza. Pred spúšťaním testov je nutné naplniť vstupné dáta správnymi údajmi a po volaní metód skontrolovať správnosť uloženia, zmeny alebo zmazania týchto údajov priamo v databáze. Nie je možné púšťať testy automaticky a bez spätnej kontroly, keďže nemôžeme zaručiť, že údaje v databázových tabuľkách budú rovnaké pri púšťaní testov v rozličných časoch.

Aj keď nie umožnené automatické spúšťanie testov, bola snaha o čo najjednoduchšie spúšťanie testov užívateľom. Modul obsahuje hlavnú triedu `Main`, ktorá slúži na spúšťanie testov z ostatných tried. Je využité spracovanie vstupných parametrov pomocou knižnice Apache Commons CLI, pričom každý parameter spúšťa naraz všetky testy jednej triedy. Medzi parametre ešte patrí parameter `help` pre vytlačenie nápovede a parameter `all` pre spustenie testov zo všetkých tried. Testy kontrolujú chybné vstupy, ktoré nebolo možné skontrolovať pomocou validátoru (napríklad zisťovanie duplicít v databáze – záznamy s rovnakým kódom) a niekoľkokrát volajú metódy s platnými údajmi pre dôkladnú kontrolu funkčnosti.

Testy v triede `Insert` slúžia pre kontrolu vkladania nových záznamov do tabuliek `knz_voucher`, `knz_voucher_typ` a `knz_voucher_vlastnsot` a trieda `Update` slúži pre kontrolu editácie údajov v týchto tabuľkách. Trieda `Select` načítava údaje z tabuliek podľa daných vstupných parametrov, výstup zaloguje a samotná kontrola, či metóda vrátila správne údaje, je na užívateľovi, ktorý testy volá. Trieda `Find` vyhľadáva údaje v tabuľkách podľa kódu alebo dátumov platnosti, pričom kontrola výstupu je opäť úlohou užívateľa.

5.3.3 Testovanie webovej služby

Po otestovaní všetkých vytvorených funkcií v beanoch je treba otestovať webovú službu, ktorá tieto metódy volá. Je teda nutné sa zamerať predovšetkým na správne fungovanie metód poskytovaných webovou službou, či metódy pracujú tak ako majú, či volajú správne metódy zo vzdialeného rozhrania a či pri chybách vracajú správne výnimky. Webová služba bola testovaná pomocou SOAP UI, čo je testovací nástroj pre testovanie REST a SOAP webových služieb.

Testovanie umožňuje volať metódy webovej služby, ale samotná kontrola a porovnanie, aký výstup má byť pre dané vstupné dáta, je na užívateľovi, ktorý tieto testy volá. Týmto

spôsobom boli otestované všetky metódy webovej služby, aby bola zaručená ich správna funkčnosť.

Kapitola 6

Záver

Práca popisuje návrh a implementáciu webovej služby pre správu zľavových voucherov. Služba poskytuje možnosť vytvorenia, editácie, mazania, vyhľadávania a načítania záznamov o voucheroch, ich typoch a vlastnostiach v databázových tabuľkách, zisťovanie platnosti voucheru a uplatňovanie voucheru, generovanie štatistík o použitých voucheroch a možnosť importu voucherov z formátu CSV. Výsledkom je fungujúca aplikácia, z ktorej väčšia časť sa už reálne používa v poisťovniach Kooperativa, a.s. a Česká podnikateľská poisťovňa, a.s. Ostatné časti by sa mali dostať do prevádzky v nadchádzajúcich mesiacoch. Aplikácia bola otestovaná na reálnych dátach a zatiaľ nevykazuje žiadne známky chybovosti.

Na začiatku tvorby práce bolo treba naštudovať potrebnú literatúru a analyzovať existujúce riešenia tohto problému. Následne sa navrhlo nové riešenie, pričom sa brali do úvahy predchádzajúce poznatky. Po vytvorení návrhu novej aplikácie sa mohlo prejsť do fáze implementácie a testovania aplikácie. Po dôkladnom otestovaní vytvorenej webovej služby, mohla byť táto služba nasadená do prevádzky pre dennodenné použitie.

Text tejto práce je užitočný pre ľudí, ktorí sa chcú naučiť programovať webové služby v jazyku Java, prípadne tvorbu podnikových aplikácií v Jave EE.

Možnými rozšíreniami aplikácie, ktoré sú naplánované v ďalšej fáze vývoja sú generovanie štatistík o prepadnutých voucheroch, teda voucheroch, ktoré nie sú platné a ani počas svojej platnosti neboli uplatnené, rozdeľovanie voucherov medzi koncových užívateľov podľa výšky rozpočtu daného užívateľa alebo generovanie grafov o použitých voucheroch.

Literatúra

[1] Booth, D.; Haas, H.; McCabe, F.; aj.: *Web Services Architecture*. [Online; cit. 12.02.2017].
URL https://www.w3.org/TR/ws-arch/

[2] Chinnici, R.; Moreau, J.-J.; Ryman, A.; aj.: *Web Services Description Language (WSDL)*. [Online; cit. 15.03.2017].
URL https://www.w3.org/TR/wsdl20/

[3] Evans, I.: *Learning the Java Language*. [Online; cit. 25.04.2017].
URL https://docs.oracle.com/javase/tutorial/java/annotations/index.html

[4] Evans, I.: *Your First Cup: An Introduction to the Java EE Platform*. [Online; cit. 19.03.2017].
URL https://docs.oracle.com/javaee/7/JEEFC.pdf

[5] Herout, P.: *Učebnice jazyka Java*. Kopp, 2013, iSBN: 9788072323982.

[6] Jendrock, E.; Cervera-Navarro, R.; Evans, I.; aj.: *Java Platform, Enterprise Edition The Java EE Tutorial*. [Online; cit. 18.02.2017].
URL https://docs.oracle.com/javaee/7/JEETT.pdf

[7] Serme, G.; de Oliveira, A. S.; Massiera, J.: Enabling Message Security for RESTful Services. *2012 IEEE 19th International Conference on Web Services*, 2012: s. 114–121.

[8] Serrano, N.; de Oliveira, J. H.; Gallardo, G.: Service-Oriented Architecture and Legacy Systems. *IEEE Software*, ročník 31, 2014: s. 15–19.

[9] Tihomirovs, J.; Grabis, J.: Comparison of SOAP and REST Based Web Services Using Software Evaluation Metrics. *Information Technology and Management Science*, ročník 19, 2016.

Prílohy

Príloha A

Obsah CD

Obsah priloženého CD:

- **Read.me** – súbor pre popis obsahu CD
- **manual.txt** – súbor pre popis spúšťania programu
- **xjurdo00_bp.pdf** – technická správa vo formáte PDF
- **latex** – adresár so súbormi k sádzaniu technickej správy
- **src** – adresár so zdrojovými kódmi programu

Príloha B

Databázové tabuľky

Prehľad štruktúry všetkých databázových tabuliek, s ktorými pracuje webová služba pre správu voucherov.

Názov stĺpca	Dátový typ	Dĺžka	Povinný	PK	FK	Index
id	SERIAL	10	Áno	*		*
id_knz_voucher_typ	INTEGER	10	Áno		*	*
kod	NVARCHAR	12,4	Áno			*
agentura	CHAR	3	Nie			*
maxsle	INTEGER	10	Áno			
maxsle_kc	INTEGER	10	Nie			
kodzi	CHAR	10	Nie			*
id_uzi_uzivatel	INTEGER	10	Nie		*	*
plod	DATE		Nie			
pldo	DATE		Nie			
ins_kdo	INTEGER	10	Áno		*	*
ins_kdy	DATETIME		Áno			

Tabuľka B.1: Tabuľka knz_voucher pre uloženie zľavového voucheru.

Názov stĺpca	Dátový typ	Dĺžka	Povinný	PK	FK	Index
id	SERIAL	10	Áno	*		*
id_knz_voucher	INTEGER	10	Áno		*	*
use_kdy	DATETIME		Áno			*
klic_calc_flat	INTEGER	10	Áno		*	*
sle	INTEGER	10	Áno			
sle_kc	INTEGER	10	Áno			
sle_kc_sml	INTEGER	10	Nie			
id_uzi_uzivatel	INTEGER	10	Áno		*	*
vin	CHAR	17	Nie			

Tabuľka B.2: Tabuľka knz_voucher_use pre uloženie uplatnenia zľavového voucheru.

Názov stĺpca	Dátový typ	Dĺžka	Povinný	PK	FK	Index
id	SERIAL	10	Áno	*		*
kod	VARCHAR	5	Áno			*
nazev	NVARCHAR	64	Áno			
popis	NVARCHAR	255	Áno			
vlastnost	INTEGER	10	Áno			
jednorazovy	CHAR	1	Áno			

Tabuľka B.3: Tabuľka knz_voucher_typ pre uloženie typu zľavového voucheru.

Názov stĺpca	Dátový typ	Dĺžka	Povinný	PK	FK	Index
id	SERIAL	10	Áno	*		*
kod	INTEGER	10	Áno			*
nazev	NVARCHAR	64	Áno			

Tabuľka B.4: Tabuľka knz_voucher_vlastnost pre uloženie vlastnosti typu zľavového voucheru.

Názov stĺpca	Dátový typ	Dĺžka	Povinný	PK	FK	Index
id	SERIAL	10	Áno	*		*
id_knz_voucher_log_action	INTEGER	10	Áno		*	*
ins_kdo	INTEGER	10	Áno			
ins_kdy	DATETIME		Áno			

Tabuľka B.5: Tabuľka knz_voucher_log pre uloženie prevádzaných akcií.

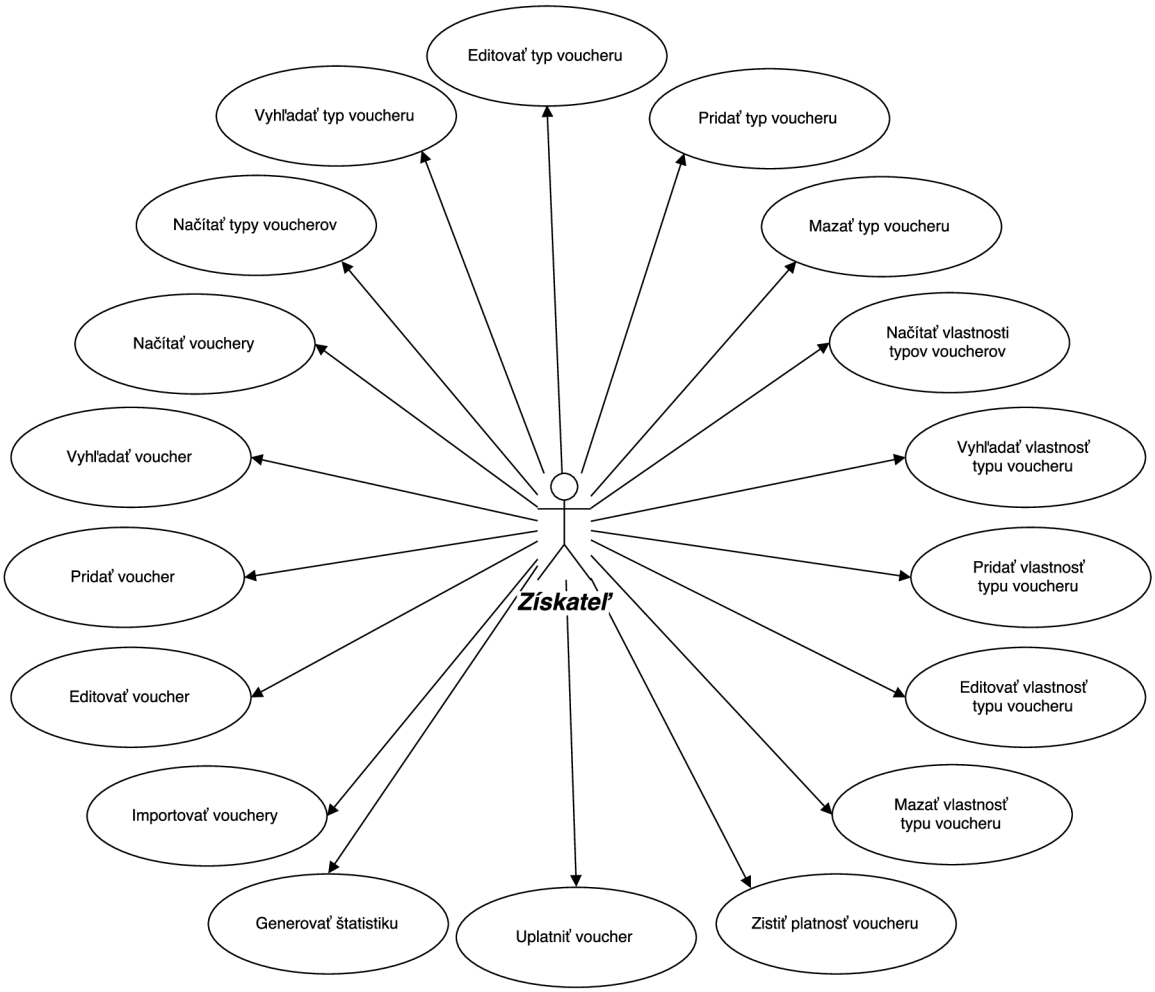
Názov stĺpca	Dátový typ	Dĺžka	Povinný	PK	FK	Index
id	SERIAL	10	Áno	*		*
action_code	INTEGER	10	Áno			
table_name	NVARCHAR	50	Áno			
action_name	NVARCHAR	20	Áno			
discription	LVARCHAR		Áno			

Tabuľka B.6: Tabuľka knz_voucher_log_action pre uloženie akcií pre logovanie.

Príloha C

UML graf

UML graf popisujúci akú funkcionalitu poskytuje webová služba pre správu voucherov.



Obr. C.1: UML graf webovej služby pre správu voucherov.