VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

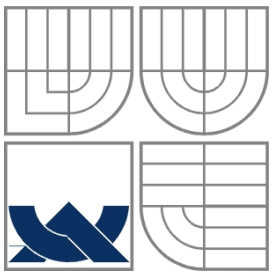
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

WEBOVÝ INFORMAČNÍ SYSTÉM PRO PODPORU AMBULANTNÍCH ORDINACÍ

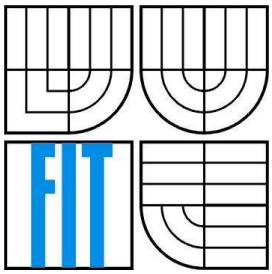
BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE MIROSLAV KOREŇ
AUTHOR

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

WEBOVÝ INFORMAČNÍ SYSTÉM PRO PODPORU AMBULANTNÍCH ORDINACÍ

WEB INFORMATION SYSTEMS FOR AMBULATORY CARE

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE
AUTHOR

MIROSLAV KOREŇ

VEDOUCÍ PRÁCE
SUPERVISOR

ING. PAVEL JURKA

BRNO 2008

Abstrakt

Cieľom bakalárskej práce “Webový informačný systém pre podporu ambulantných ordinácií” je zoznámiť sa s ambulantnými systémami a pochopiť ich princípy. Na základe získaných poznatkov potom vytvoriť takýto systém vlastný a zhodnotiť dosiahnuté výsledky. Vytvorený systém obsahuje základné funkcie bežne používané ambulantnými lekármi, je otvorený ďalším možnostiam vývoja. Technologicky je koncipovaný ako webový. Vznikol kombináciou prostriedkov pre vývoj webových informačných systémov a rozhraní v jazyku Java od firmy Sun. Konkrétne použitím JSF (Java Server Faces), JSP (Java Server Pages) a Servlets.

Kľúčové slova

J2EE, JSP, Servlets, JSF, MyFaces, ambulantný informačný systém, pacient, lekár, poisťovňa

Abstract

Aim of this bachelor's thesis “Web ambulatory care information system” is to gain information about ambulatory systems and comprehend their principles. On this knowledge base build up own ambulatory system and assume results. Created application involves basic functionality provided by obvious ambulatory systems and is opened for future update. System is conceived as web application. It was built up with the usage of facilities for development web information systems in Java language, specifically by using JSF (Java Server Faces), JSP (Java Server Pages) and Servlets.

Keywords

J2EE, JSP, Servlets, JSF, MyFaces, information system for ambulatory care, pacient, doctor, health-insurance company

Citácia

Koreň Miroslav: Webový informační systém pre podporu ambulantných ordinácií. Brno, 2008, bakalárska práca, FIT VUT v Brně.

Webový informační systém pre podporu ambulantných ordinácií

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením Ing. Pavla Jurky. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

……
Meno Priezvisko
Dátum

Poďakovanie

Ďakujem Ing. Pavlovi Jurkovi za cenné rady a pripomienky, ktoré mi poskytol pri návrhu, implementácii systému a písaní tejto práce. Moje poďakovanie ďalej patrí tým, ktorí mi pri práci pomohli.

Obsah

Úvod

Rozvoj informačných technológií v poslednom období zasahuje takmer všetky oblasti života, neobchádza ani takú oblasť ako je zdravotníctvo, konkrétne poskytovanie zdravotnej starostlivosti ambulantnými lekármi v ordináciách. Cieľom bakalárskej práce je zoznámiť sa s existujúcimi systémami pre podporu ambulantných ordinácií a na základe získaných poznatkov navrhnúť a implementovať systém vlastný. Ten má byť vystavaný na platforme Java Enterprise Edition.

Táto práca je súhrnom poznatkov a skúseností získaných pri vývoji informačného systému tohto druhu.

V prvej kapitole načrtáva princípy nasadených ambulantných systémov a popisuje ich vlastnosti. Poskytuje prehľad najpoužívanejších systémov.

Druhá kapitola definuje informačné systémy a informuje o postupoch ako má byť aplikácia tohto typu vystavaná, aby bola úspešná.

Práce na systéme ešte pred samotnou implementáciou zhrňuje kapitola číslo tri. Sumarizuje požiadavky na systém, možnosti vzniku problémov pri vývoji, analyzuje vytvorenú špecifikáciu systému.

Kapitola číslo štyri predstavuje vzniknutý návrh systému. Informuje o technológiách, ktoré boli pre tvorbu systému vybraté. Popisuje výhody použitia kombinácie týchto vývojových prvkov.

Implementácia systému je popísaná v štvrtej kapitole. Oboznamuje čitateľa o vývoji systému a popisuje jednotlivé fázy kódovania, ktoré vychádzajú z použitej technológie.

Piata kapitola naznačuje ako sa s vytvoreným systémom pracuje. Bližšie vysvetľuje jednotlivé úkony, ktoré lekár so systémom prevádza.

Posledná kapitola predstavuje súhrnný pohľad na systém, jeho zhodnotenie a návrhy jeho rozšírenia.

1 Ambulantné systémy

Skvalitnenie a zefektívnenie zdravotnej starostlivosti patrí medzi hlavné zámery mnohých rýchlo sa rozvíjajúcich krajín, nevynímajúc Českú alebo Slovenskú republiku. Informatizácia zdravotníctva je dôležitým podporným faktorom pre dosiahnutie tohto zámeru. Jednu z nosných častí systému zdravotníctva predstavuje činnosť ambulantných lekárov. Pre zrýchlenie a zefektívnenie ich práce sa v minulých rokoch začali zavádzať ambulantné systémy.

Ambulantný systém zabezpečuje základnú agendu ambulantného lekára v digitálnej forme. Zápis údajov v takejto podobe umožňuje rýchly a efektívny prístup k nim, možnosť vytvárania prehľadov a štatistík o zdravotnej starostlivosti, elektronické vykazovanie starostlivosti. Údaje v digitálnej podobe sú lepšie udržovateľné ako v podobe písomnej. Nasadenie takýchto systémov zaručuje integráciu a zjednotenie zdravotnej starostlivosti v rámci systému zdravotníctva krajiny. S príchodom moderných zobrazovacích technológii a výkonnejšej architektúry PC sa systémy stávajú užívateľsky prívetivejšie a výkonnejšie.

Moderné ambulantné systémy nasadzované v praxi by mali zabezpečovať vedenie kartotéky pacientov, záznamy vyšetrení, ordináciu liekov, agendu pracovných neschopností, vystavovanie potrebných zdravotníckych dokladov, možnosť syntetického pohľadu na jednotlivé elementy histórie ochorení (lieky, vyšetrenia, diagnózy), vykazovanie a vyúčtovanie zdravotnej starostlivosti, riešenie výberu poplatkov u lekára, množstvo prehľadov a štatistík, v rámci Českej republiky aj spoluprácu so systémom IZIP (elektronická knižka pacienta), prípadne možnosť vzdialenej správy a mnohé iné.

V rámci Českej republiky sa v praxi bežne používa hneď niekoľko informačných systémov pre ambulancie. Najpoužívanejší a najúspešnejší je zrejme systém PC Doctor vyrábaný spoločnosťou Dialog Mis, spol s.r.o.. Program poskytuje množstvo funkcií približne v rozsahu popísanom vyššie. Je elastický z pohľadu prispôsobenia požiadavkám na režim práce v ordinácii. Jeho obsluha je nenáročná a užívateľsky prívetivá. Je vybavený radou funkcií pre administráciu a údržbu aj pri prevoze v sieťovej verzii, ale nevyžaduje pritom od obsluhy žiadne špeciálne odborné znalosti. Rovnakou spoločnosťou je kontrolovaný vývoj systému Amicus. Amicus je program určený pre všetky typy ambulancií okrem stomatologických. Lekármi je cenený pre rýchle vyúčtovanie zdravotnej starostlivosti poisťovniam a pre rýchle dosiahnutú aktuálnu podporu pre prácu s programom. Spoločnosť Medisoft International, s.r.o. vyvíja produkt Medisoft. Ide o ambulantný systém určený pre prácu na jednom počítači poprípade v menších sieťach. V praxi sa používajú ešte mnohé ďalšie napr. Ais, Ambulance atď..

2 Informačné systémy a ich návrh

Informačný systém je otvorený systém, ktorý zhromažďuje, uchováva, spracováva a poskytuje informácie o spracovávanej realite. Architektúru informačných systémov vo väčšine prípadov predstavuje 3-vrstvová architektúra skladajúca sa z prezentačnej, funkčnej a dátovej časti. Prezentačná vrstva zabezpečuje komunikáciu s užívateľom v prívetivej podobe, funkčnú vrstvu tvorí vlastná aplikácia a dátovú vrstvu dáta uložené trvanlivo najčastejšie v databáze.

2.1 Životný cyklus

Pochopenie problematiky životného cyklu software[2] dáva základ pre vnesenie hierarchie a poriadku do vývoja aplikácií. Životný cyklus informačného systému je odstupňovaný do viacerých fáz, pričom sa ustálilo nasledujúce poradie:

- Analýza a špecifikácia požiadaviek
- Architektonický a podrobný návrh
- Implementácia
- Integrácia a testovanie
- Beh a údržba

2.2.1 Analýza a špecifikácia

Analýza a špecifikácia je prvou etapou vývoja, v ktorej sa rozoberajú požiadavky zákazníka na systém. Získajú sa, analyzujú, špecifikujú. Z nejasných požiadaviek sa stávajú požiadavky jasné, úplné, citované. Realizácia požiadaviek nie je v tejto etape uvažovaná. Vytvára sa štúdia vhodnosti, analýza možných rizík, či plán testovania.

2.2.2 Architektonický a podrobný návrh

Architektonický návrh nadväzuje na analýzu požiadaviek a slúži k ujasneniu celkovej koncepcie. Na základe nej môže byť prevedená dekompozícia systému a definovanie väzieb medzi jednotlivými komponentmi. Podrobný návrh sa sústreďuje na podrobnú špecifikáciu softwarových súčastí, výber algoritmov realizujúcich požadované funkcie, na stanovenie logickej a fyzickej štruktúry dát. V tejto fáze sa plánujú práce na implementácii systému, požiadavkách na ľudské, či systémové zdroje, odhad času a náklady spojené s realizáciou. Dobrý návrh je podmienkou úspešnej tvorby softwaru a minimalizuje náklady spojené s implementáciou a údržbou softwarového produktu.

Návrhy tejto fázy sú často prezentované vo forme rôznych modelov. Najpoužívanejším prostriedkom pre modelovanie dát je ER-diagram. Modeluje entity aplikácie spracovávané systémom

a statické vzťahy medzi nimi. Reprezentuje požiadavky na dáta uložené v databáze, vychádza z neho preto väčšinou návrh tabuliek relačnej databázy.

2.2.3 Implementácia

Obsahuje programovú realizáciu jednotlivých softwarových súčastí na základe predložených návrhov, vypracovávanie dokumentácie k jednotlivým súčastiam, testovanie vytvorených súčastí.

2.2.4 Integrácia a testovanie

Po fáze, kde boli vytvorené jednotlivé súčasti je nutné túto mozaiku spojiť do jediného celku. Testovanie je nutné aj v tejto etape bez ohľadu na testovanie vo fáze implementácie, pretože sa manipuláciou a spájaním súčastí môžu objaviť chyby nové.

2.2.5 Beh a údržba

Na základe akceptačných testov vytvorených vo fáze návrhu je prevedené testovanie systému cieľovým užívateľom, ktorý sa rozhodne systém prijať, alebo odložiť do doby, kým dodávateľ neodladí všetky viditeľné chyby systému. Systém väčšinou predstavuje dynamický celok, ktorý sa v čase mení. Aj po nasadení systému sa odhalia chyby doposiaľ neznáme. Preto dodávateľ zabezpečuje údržbu systému, prípadne dodanie nových modulov a zmeny systému na základe požiadaviek zákazníka.

3 Špecifikácia a analýza požiadaviek

Táto kapitola predstavuje popis úvodnej fázy tvorby informačného systému ambulantnej ordinácie. Snaží sa premietnuť získané poznatky zo sledovania existujúcich ambulantných systémov do vytvorenia požiadaviek na systém. Predstavuje analýzu týchto požiadaviek a zobrazuje model, ktorý popisuje budúce činnosti užívateľa pri práci so systémom.

3.1 Požiadavky na výsledný systém

Informačný systém ambulantnej ordinácie je značne rozsiahla téma. Zachytiť v ňom všetky úkony administratívy lekára tak, aby bol systém použiteľný v praxi si žiada zrejme väčší časový úsek, ako je poskytnutý k spracovaniu bakalárskej práce. Preto si práca kladie za vzor vytvorenie takého systému, ktorý sa bude približovať existujúcim vzorom. Mal by umožniť viac-užívateľský prístup do aplikácie, poskytnúť jednoduchú správu užívateľov v administračnom režime tak, aby bola možná editácia užívateľských účtov. Systém musí byť bezpečný, aby povoľoval prístup len povoleným užívateľom. Mal by vytvárať príjemné rozhranie pre prácu, zabezpečiť rýchle prevedenie úkonov v ňom, efektívny a intuitívny prístup k jednotlivým funkciám aplikácie. Po ukončení vývoja by mal systém zostať v stave, ktorý umožňuje jeho ďalšie rozšírenie.

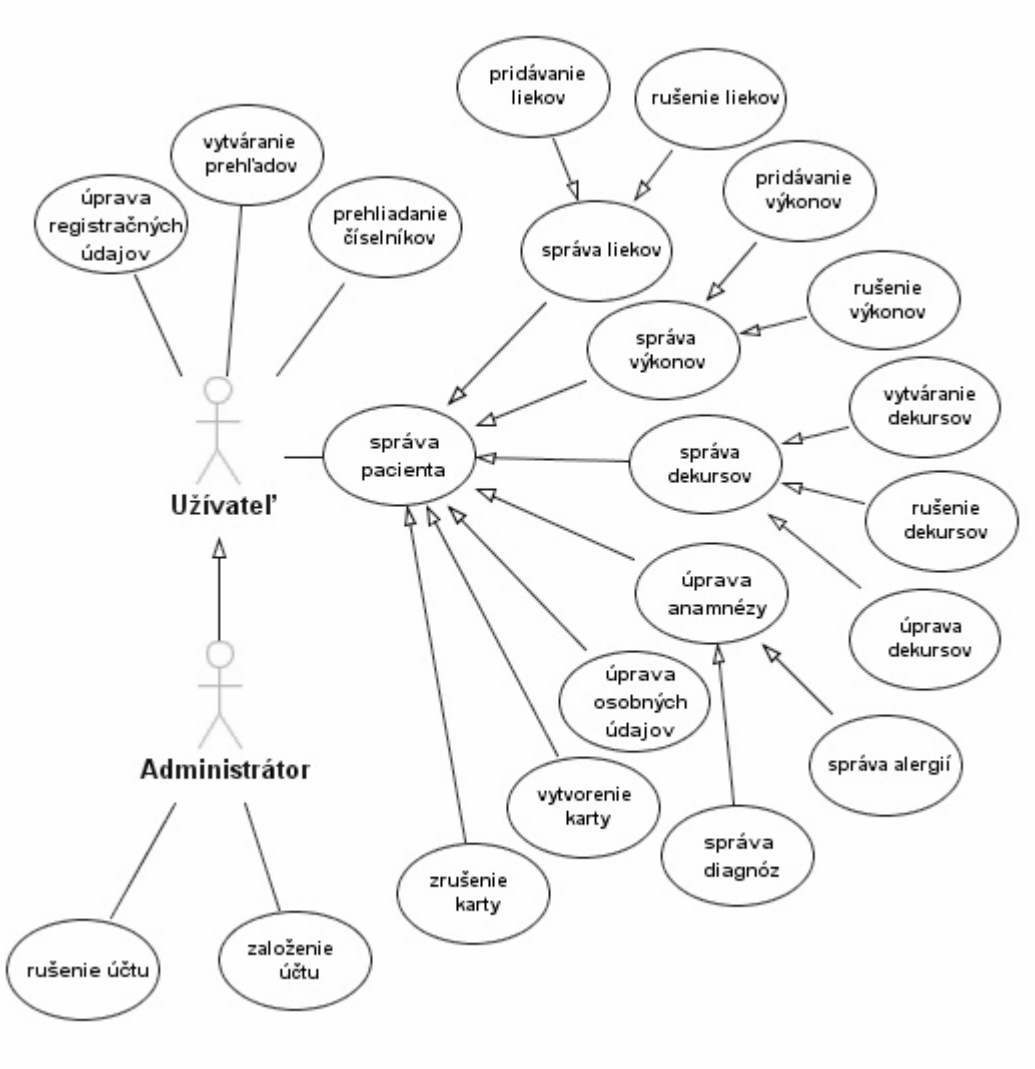
Aplikácia by mala poskytnúť možnosť práce s kartotékou pacientov, rýchle vyhľadanie pacienta a informácii o ňom v nej. Mala by poskytovať mechanizmy pre triedenie kartotéky a možnosti definovať výber pacientov podľa ich vlastností. Má umožniť prístup k pacientom jednotlivo na základe výberu z kartotéky. V karte pacienta potom možnosť editovať ako osobné tak zdravotné záznamy pacienta.

Systém má zobrazovať informácie o celkovej anamnéze pacienta, zachytiť jeho alergie, trvalé diagnózy, lieky, ktoré užíva. Umožní editáciu týchto informácií. Bude dokumentovať prevedenú zdravotnú starostlivosť prostredníctvom správy výkazov. Výkazy budú reprezentované v textovej podobe, systém umožní prácu s ich históriu a možnosti ich editáciu v čase. V praxi využívané ambulantné systémy umožňujú vykazovať zdravotnú starostlivosť prostredníctvom výkonov zmluvným poisťovniam. Vytvorený systém poskytne jednoduché rozhranie pre správu takýchto výkonov.

Cieľom je ďalej umožniť vytváranie prehľadov o poskytnutej zdravotnej starostlivosti. Výsledný systém by mal obsahovať správu číselníkových hodnôt a umožniť pracovať s týmito údajmi ako aj inými údajmi poskytovanými v rámci zdravotníckeho systému ČR.

3.2 Diagram prípadov užitia

Diagram špecifikuje užívateľské role, v akých je systém prístupný. Predstavuje celkový prehľad o úkonoch, ktoré bude ambulantný lekár vykonávať behom práce so systémom.



3.2.1 Špecifikácie rolí

Role v diagrame predstavujú užívateľov, ktorí budú so systémom pracovať. Aplikácia bude dostupná prostredníctvom dvoch módov ako užívateľ-lekár, alebo ako administrátor. Lekár môže spravovať iba vlastný užívateľský účet, administrátor bude mať neobmedzený prístup ku všetkým účtom. V systéme má byť pred jeho nasadením vytvorený účet typu administrátor. Po vstupe do systému prostredníctvom tohto účtu sa môžu registrovať ďalší užívatelia.

3.2.2 Špecifikácie prípadov užitia

Prípady užitia v diagrame prezentujú činnosti, ktoré bude lekár so systémom vykonávať. Nasledujúca časť popisuje prípady užitia slovne, aby naznačila ich špecifiká.

3.2.2.1 Správa užívateľských účtov

Administračný režim systému ponúkne možnosti správy užívateľských účtov lekárov. Umožní pridanie lekárov nových ako aj rušenie účtov viac nepotrebných. Pre lekára prihláseného v bežnom užívateľskom režime bude prístupná editácia vlastného účtu.

3.2.2.2 Správa pacienta

Súhrn informácií o pacientovi a jeho zdravotnom stave predstaví karta pacienta. Systém umožní vytváranie kariet nových, ich rušenie a editáciu. Prihlásenie sa k lekárovi zrejme vyústi do vytvorenia karty, ďalšie návštevy pacienta potom do jej editácie. Editovať bude možné osobné údaje pacienta zahrňujúce aj informácie o jeho zamestnaní či bydlisku. Systém umožní reprezentovať údaje o zdravotnom stave v podobe anamnézy pacienta a poskytne akcie pre prácu s ňou. Vytvorí vhodné nástroje a editory pre zaznamenanie každej návštevy pacienta u lekára v podobe dekursu (dekurs je textový záznam o priebehu choroby pacienta). Požiadavkou na správu dekursov je zobrazovanie dekursov minulých, ich možný výber podľa diagnózy, či času vyhotovenia, úprava dekursov. Aplikácia bude obsahovať vhodný dialóg pre reprezentáciu výkonov o zdravotnej starostlivosti pacienta. Bude umožnené vkladanie a mazanie záznamov o výkonoch. Systém poskytne možnosť definovať lieky pre pacienta.

3.2.2.3 Správa číselníkov

Významnou požiadavkou na činnosť aplikácie je sprístupňovať číselníky podľa potrieb v dialógoch výkonov, dekursov či liekov. Požaduje sa vytvoriť miesto v aplikácii, kde budú číselníky prístupné všetky dohromady.

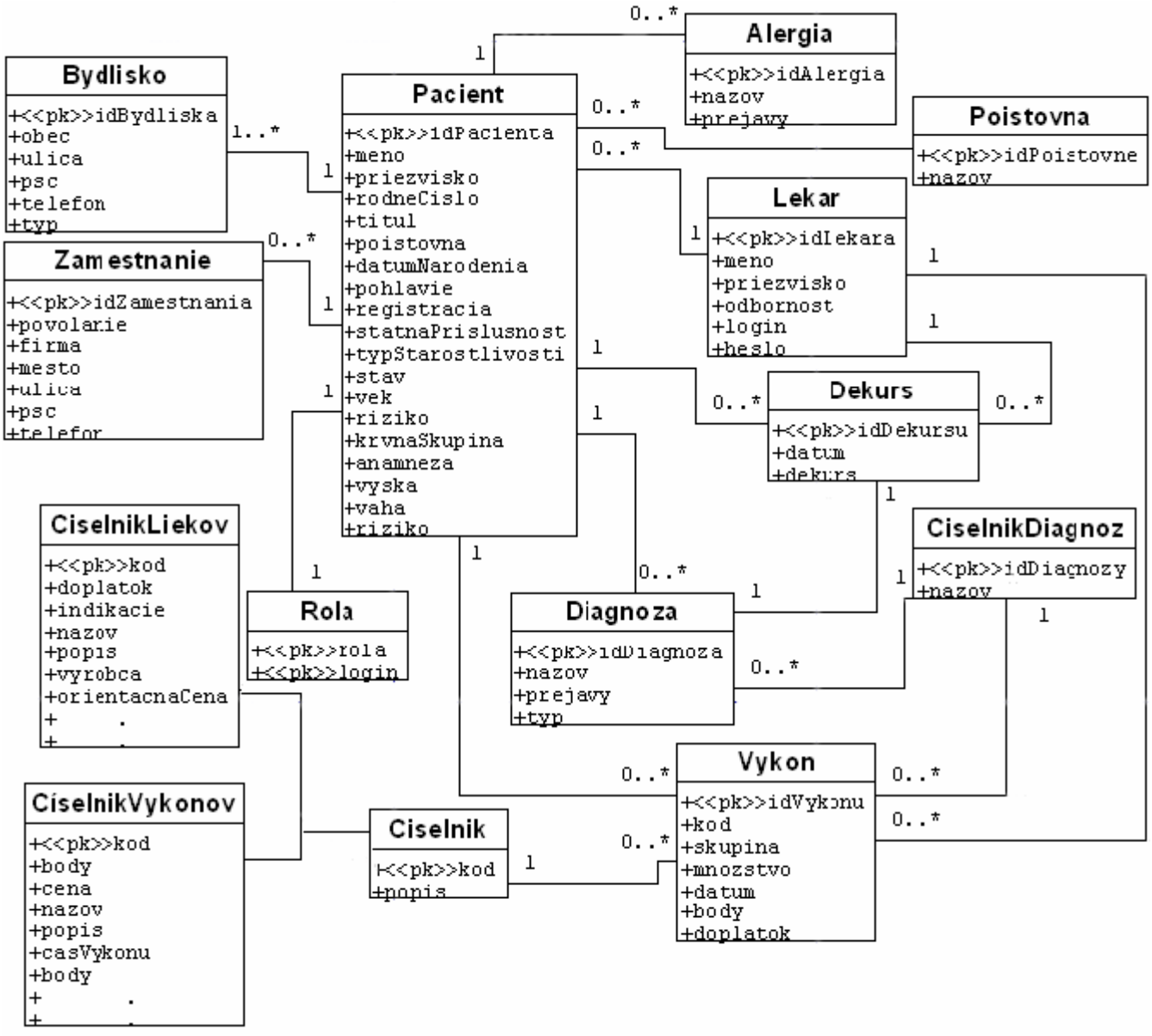
3.2.2.4 Vytváranie prehľadov

Systém umožní vytvárať lekárom definované prehľady. Rozhodovacím kritériom pri ich zobrazovaní bude čas. Prehľady budú tvorené výkonmi, dekursmi, liekmi a diagnózami.

4 Návrh systému

Po špecifikovaní požiadaviek na ambulantný systém boli vytvorené návrhy a modely systému. Z nich sa vychádzalo pri tvorbe databázovej schémy, či tvorbe modelovaných objektov. Táto kapitola prezentuje návrh systému v podobe entít a vzťahov medzi nimi. Informuje aj o použitej technológii a uvádza dôvody jej použitia.

4.1 ER Diagram



Bližší popis jednotlivých entít a údajov o nich, ktoré musia byť uchované v systéme predstavujú nasledujúce riadky.

4.1.1.1 Entita pacient

Pacient je entita kľúčová, systém bude pracovať prevažne s jeho údajmi a údajmi k nemu sa viažucimi. V systéme sa budú uchovávať osobné údaje o pacientovi, ktoré ho dostatočne popisujú z pohľadu potrieb lekára (meno, priezvisko, rodné číslo, titul, poisťovňa, dátum narodenia, pohlavie, štátna príslušnosť, stav, vek). K pacientovi sa ďalej viaže dátum jeho registrácie do systému a údaje o jeho zdravotnom stave (krvná skupina, výška, váha, krvný tlak).

4.1.1.2 Entita bydlisko

Táto entita zachytáva údaje o bydlisku pacienta, či už trvalom alebo prechodnom. Systém bude koncipovaný tak, aby mal každý pacient jedno trvalé bydlisko a voliteľne bydlisko prechodné. Údaje, o bydlisku, ktoré budú v systéme zachytené sú mesto, ulica, poštové smerové číslo, telefónny kontakt, a typ bydliska, teda či ide o bydlisko prechodné alebo trvalé.

4.1.1.3 Entita zamestnanie

Zamestnanie, ako sám názov napovedá zhŕňa údaje o tom ako a kde pacient pracuje. Atribútmi, ktoré budú uchovávané sú povolanie, teda čo daný pacient robí, firma- teda v akom subjekte pracuje, a popisné údaje o mieste pracoviska spolu s telefónnym kontaktom. Systém predpokladá existenciu jediného zamestnania pacienta.

4.1.1.4 Entita poisťovňa

Poisťovňa určuje, kde je pacient zdravotne poistený. V systéme bude určená svojim názvom, a identifikátorom, ktorý je zhodný s identifikáciou zdravotných poisťovní v Českej republike.

4.1.1.5 Entita rola

Entita rola vznikla ako dodatok systému v súvislosti z nasadením autentizácie do aplikácie. Je daná spôsobom akým server Apache chráni aplikáciu.

4.1.1.6 Entita lekár

Lekár v systéme podobne ako pacient zohráva úlohu kľúčovú, jednoznačne určuje celú hierarchiu pacientov, ktorí pod jeho starostlivosť spadajú a následne aj úkony, ktoré sa pacienta týkajú. Lekár bude v systéme reprezentovaný množinou atribútov meno, priezvisko, odbornosť a prihlasovacie údaje do systému.

4.1.1.7 Entita dekurs

Dekurs predstavuje záznam o chorobe pacienta. K textovému zápisu o priebehu choroby sa v systéme zachytí diagnóza danej choroby a dátum spracovania záznamu.

4.1.1.8 Entita výkon

Výkon lekára prezentuje činnosť jeho práce poisťovni. Výkony sú zaradené do skupín, pričom ambulantný lekár bežne vykazuje vykonané činnosti, hromadne vyrábané liečivé prípravky, prostriedky zdravotníckej techniky, individuálne vyrábané liečivé prípravky. Pre zjednodušenie budú v aplikácii uvažované len výkony a lieky. Kód výkonu je číselníková hodnota, ktorú spravuje VZP (Všeobecná zdravotná poisťovňa). VZP tieto číselníky neustále aktualizuje a jednotlivým výkonom priradzuje jednoznačný kód a ďalšie množstvo popisných údajov. Množstvo prevedeného výkonu signalizuje koľko krát bol daný výkon prevedený, poprípade koľko liekov bolo pri danom výkone vystavených. Každému výkonu je ďalej pripojený dátum jeho vykonania.

4.1.1.9 Entita číselník výkonov

Číselník výkonov predstavuje dátové rozhranie VZP pre potreby lekárov. VZP určuje u výkonov veľké množstvo hodnôt k výkonom priradených. Sú to napríklad čas potrebný na vykonanie výkonu, priame mzdové náklady na výkon, odbornosť lekára nutná k prevedeniu daného výkonu, kategória výkonu a mnoho ďalších. Pre potreby tejto aplikácie sa v číselníku výkonov bude výkon popisovať jednoznačne svojim kódom daným VZP. Ďalej sa bude uchovávať názov výkonu, jeho body a cena výkonu. Atribút body je obvykle vo vyúčtovaniach vyplnený ako súčet bodovej hodnoty výkonu a réžiou výkonu (vypočítaná ako súčin výkonu a minútovej režijnej sadzby podľa odbornosti výkonu). V aplikácii sa pre zjednodušenie bude považovať hodnota body ako jednoduché a nijak nevypočítavané číslo. Cena prezentuje objem peňazí v korunách za prevedený výkon.

4.1.1.10 Entita číselník liekov

Číselník liekov VZP opäť udáva veľké množstvo dát s liekom súvisiacich. Systém sprostredkuje názov a kód lieku, atribút DOP - doplnok bližšie charakterizujúci jeho názov: lieková forma, veľkosť balenia. Ďalej je to orientačná cena v korunách a úhrada, teda aké množstvo z ceny si pacient hradí sám.

4.1.1.11 Entita číselník diagnóz

Číselník diagnóz nepredstavuje oficiálny číselník VZP. Pre potreby systému bude zostavený z medzinárodnej klasifikácie chorôb, konkrétne z jej 10. revízie. Zachytí úplný názov choroby a jej kód. S kódu bude potom možné určiť, do ktorej kategórie choroba spadá.

4.2 Použité technológie

Súčasť návrhu predstavuje výber použitej technológie. Myšlienkou bolo pokúsiť sa o alternatívny prístup k tvorbe ambulantných systémov a koncipovať aplikáciu ako webovú. Pre vývoj systému boli vybrané technológie platformy J2EE, konkrétne kombinácia prostriedkov JSF, JSP a Servlety.

Obmedzená ponuka komponentov JSF framework prinútila do systému aplikovať v priebehu vývoja prostriedky projektu Apache Myfaces. Ako aplikačný server bol vybraný Apache Tomcat, ktorý je referenčným servrom technológie JSP + Servlety. Databázový server predstavuje Mysql. Nasledujúci text popisuje vybrané technológie podrobnejšie.

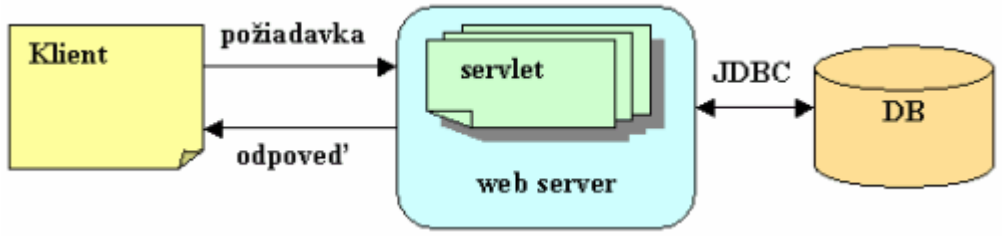
4.2.1 J2EE

Java Platform, Enterprise Edition (alebo **Java EE**, skôr označovaná ako *Java 2 Enterprise Edition* alebo *J2EE*) je súčasť platformy Java určená pre vývoj a beh podnikových aplikácií a informačných systémov. Základom pre platformu Java EE je platforma Java SE, nad ňou sú definované súčasti tvoriace Java EE[3].

Java EE je akýsi balík technológií, ktoré sa doplňujú a ktoré sú určené pre vývoj serverových systémov. Medzi základné, programátormi najviac využívané a mnou použité patria nasledujúce technológie:

4.2.1.1 Servlety

Servlety sú odpoveďou technológie Java na programovanie pomocou CGI – Common Gateway Interface. Ide o program napísaný v jazyku Java, ktorý beží na webovom serveri a pôsobí ako stredná vrstva medzi požiadavkou prichádzajúcou z webového prehliadača alebo od iného HTTP klienta a databázou, či aplikáciou na serveri HTTP. Servlety teda implementujú princíp požiadavka/odpoveď, typický pre architektúru klient-server.



Obrázok 3.1: ilustrácia procesov medzi klientom a serverom[4]

Úlohou servletov je:

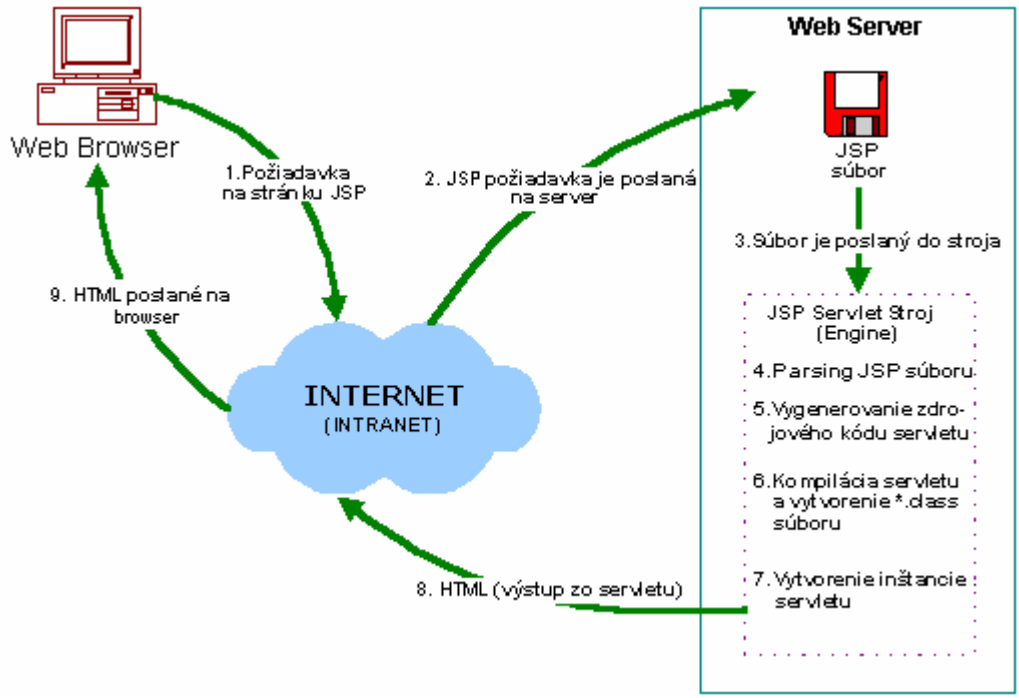
1. prečítať všetky dáta odoslané užívateľom
2. vyhľadať všetky ďalšie informácie o požiadavke, ktoré sú uložené v požiadavke HTTP
3. vytvoriť výsledky
4. sformátovať výsledky vo vnútri dokumentu
5. nastavenie vhodných parametrov odozvy HTTP
6. odoslanie dokumentu späť ku klientovi

Použitie servletov je výhodné vďaka ich účinnosti, pretože pri viacnásobnej požiadavke na servlet sa do pamäte nezavádza trieda servletu n-krát, ale vytvorí sa iba n-vlákien. Trieda servletu je však v pamäti umiestnená iba raz. Servlety majú rozsiahlu infraštruktúru pre analýzu a dekódovanie formulárových dát, dekódovanie a nastavovanie záhlaví HTTP, obsluhu cookies, či sledovanie sedení. Obsahujú špeciálne API pre komunikáciu so servrom, vďaka platforme Javy sú bez problémov prenositeľné [1][4].

4.2.1.2 JSP

Technológia Java Server Pages (ďalej len JSP) umožňuje kombinovať bežné, statické HTML stránky s dynamicky generovaným obsahom zo servletov. Väčšina stránky JSP sa skladá z normálneho HTML, ktorý sa predáva návštevníkovi bez zmeny. Časti, ktoré sú vytvárané dynamicky, sú označené špeciálnymi značkami podobnými HTML a sú vložené do stránky.

Zdrojový kód stránky je prekladaný na servlet a následne kompilovaný. Výsledok je servlet, ktorý generuje HTML. To je poslané klientovi ako odpoveď na jeho požiadavku.



Obrázok 3.2: životný cyklus JSP stránky[5]

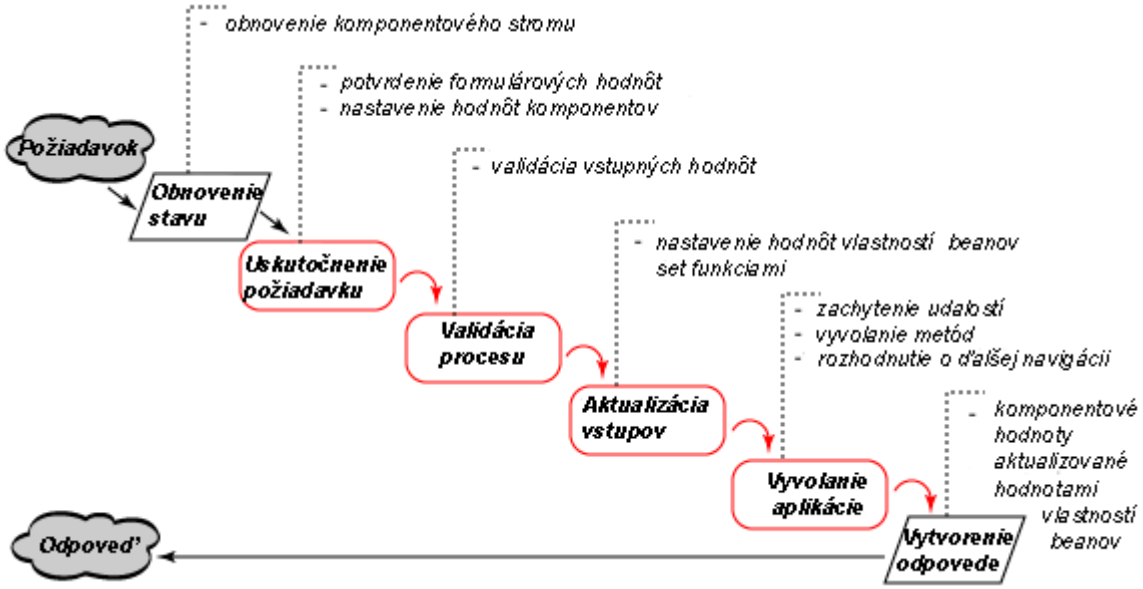
JSP má rad výhod oproti mnohým svojim alternatívam. Výhoda JSP spočíva napr. oproti PHP v tom, že dynamická časť je písaná v jazyku Java, ktorý má rozsiahle API pre prácu v sieti, prístup k databáze, distribuované objekty a podobne. Oproti prostým servletom umožňuje zapísať (upraviť) regulérne HTML, než mať milión príkazov `println`, ktoré vytvárajú HTML. Ďalej potom oddelením prezentácie od obsahu sa môžu zadať viacerým ľudom rôzne úlohy: experti na tvorbu webových stránok môžu vytvárať HTML pomocou známych nástrojov a ponechať miesto pre programátorov servletov pre vkladanie dynamického obsahu [1][5].

4.2.1.3 Java Server Faces

Java Server Faces (ďalej len JSF) je prostredie pre vývoj webových užívateľských rozhraní v jazyku Java od firmy Sun. Myšlienka stojaca za vznikom JSF bolo integrovať výhody JSP a servletov. Ďalším dôvodom pre vznik JSF bolo predovšetkým to, aby sa dizajnéri mohli venovať svojej práci a nezaťažovali sa špecifikami programovacieho jazyka a tiež, aby boli programátori zbavení povinnosti konzultovať svoj kód s týmito dizajnérmi.

JSF si je možné predstaviť ako kombináciu Struts (opensourcové prostredie pre vývoj webových aplikácií) a Swingu (prostredie pre vytváranie užívateľského rozhrania). Ako pri Struts, JSF zaisťuje riadenie životného cyklu aplikácií cez kontrolný servlet (FacesServlet), a ako pri Swingu zaisťuje bohatý komponentový model, sadu štandardných položiek užívateľského rozhrania, model pre vytváraní vlastných prvkov UI, spolu s riadením udalostí.

JSF aplikácia beží na serveri a je zabalená do štandardného servletového kontajneru ako Tomcat nebo WebLogic. Výstup posiela cez HTTP prostredníctvom rozhrania servletu a môže posielať v podstate akýkoľvek značkovací jazyk, bežne je to HTML. Kontajnerom JSF komponentov a i akousi zobrazovacou technológiou, sú obvykle JSP. Každá JSP stránka je reprezentovaná komponentovým stromom[7].



Obrázok 3.3: životný cyklus JSF[13]

Výhody tvorby webových aplikácií s JSF:

- Možnosť vytvárania užívateľských rozhraní zo zostavy štandardných komponentov.
- Poskytuje škálu JSP tagov k prístupu k týmto komponentom.
- Transparentne ukladá stavové informácie

- Poskytuje prostredie pre implementovanie vlastných komponentov.
- Zapuzdruje riadenie udalostí a predávanie komponentov, takže môžeme vytvárať štandardné JSF komponenty alebo vlastné pre podporu iných jazykov ako HTML.
- Umožňuje vývoj grafických vývojových prostredí pre webové aplikácie.
- Oddelenie ”bussiness logic” od samotnej prezentácie.
- JSF je pri vývoji webových aplikácií dostatočne silným protivníkom .NET platformy od Microsoftu

Nevýhody tvorby webových aplikácií s JSF:

- Samotný vývoj aplikácie ambulantného systému ako aj poznatky ostatných programátorov zverejnené na internete potvrdili chyby určitých komponent a logiky JSF.
- Model validátorov je navrhnutý obsluhovať len jeden komponent naraz, teda vzťah 1:N medzi komponentom a validátormi. Často sa ale stáva, že je potrebné mať väzbu N:N, napríklad keď niekomu overujeme jeho kreditnú kartu, platnosť môže byť daná viac než jedným poľom (typ karty, číslo, platnosť..).
- So štandardnými komponentmi nie je všeobecne možné pohybovať sa medzi stránkami bez vyvolávania validácie, to môže byť problém, ak do svojho formulára chcete pridať tlačidlo “predchádzajúci”.

Apache Myfaces

Apache Myfaces je projekt zahŕňajúci podprojekty týkajúce sa technológie Java Server Faces. Myfaces obsahuje konkrétnu implementáciu JSF, niekoľko knižníc komponentov obsahujúcich UI widgety pre vývoj webových aplikácii s JSF, integračné moduly pre spojenie s inými technológiami napr. s portálovými riešeniami (Java portlet). V mojej implementácii boli využité komponenty projektu Tomahawk[10].

4.2.1.4 JDBC

Jedná sa o API umožňujúci unifikovaný prístup k ľubovoľnému RDBMS, ktorý ponúka zodpovedajúci JDBC ovládač pre pripojenie. Dalo by sa povedať, že JDBC ponúka podobnú funkčnosť ako ODBC, ale s mnohými vylepšeniami, predovšetkým ide o objektový model prístupu k databáze. Aplikační programátori sa teda môžu naučiť jednotné príkazy JDBC API (ktoré sú dosť triviálne) a tie potom následne používať vo svojich riešeniach.

4.2.2 Apache Tomcat aplikačný server

Pre beh jazyku Java na serveri je potrebné použiť riešenie podporujúce štandardy firmy Sun. Server Apache Tomcat tieto štandardy podporuje.

Tomcat je relatívne jednoduchý Servlet/JSP Container, ktorý vo verzii 6.0 implementuje špecifikácie Servlet 2.4 a špecifikáciu Java Server Pages 2.0. Ako taký obsahuje ešte ďalšie nástroje, ktoré umožňujú vývoj a nasadenie webových aplikácií a webových služieb. Určitou zaujímavosťou môže byť skutočnosť, že Tomcat je referenčnou implementáciou J2EE Servlet/JSP, čo znamená, že ak budú JSP stránky fungovať pod Tomcatom, budú fungovať tiež vo všetkých J2EE certifikovaných serveroch na všetkých ich platformách.

4.2.3 Mysql Databázový server

MySQL je multiplatformná databáza. Komunikácia s ňou prebieha – ako už názov napovedá – pomocou jazyka SQL. Podobne ako u ostatných SQL databáz sa jedná o dialekt tohto jazyka s niektorými rozšíreniami. MySQL bolo od začiatku optimalizované predovšetkým na rýchlosť, a to i za cenu niektorých zjednodušení: má len jednoduché spôsoby zálohovania, a až donedávna nepodporovalo pohľady, triggery, a uložené procedúry.

4.2.4 NetBeans IDE

Jedná sa o vývojové prostredie poskytujúce prostriedky pre vývoj štandardných webových a aplikačných systémov. Zahŕňa grafické editory pre JSP stránky, Java kód servletov, javascript, kaskádové štýly atď.. Odtieňuje prístup k databázovému serveru pomocou špeciálneho rozhrania. Verzia 6.0 poskytuje vizuálny editor pre jednoduché pridávanie JSF component, túto možnosť som však pri vývoji nevyužil. Ďalej poskytuje zabudovaný aplikačný server Apache Tomcat a jednoduché prostriedky pre jeho nastavovanie, čo bolo pre mňa dôvodom pre výber práve tohto vývojového prostredia[12].

4.2.5 Tvorba webového rozhrania

4.2.5.1 Jazyk HTML

HTML je aplikáciou rozsiahleho značkovacieho jazyka SGML, kde je značkám priradená sémantika hypertextového dokumentu v prostredí Webu. Vývoj HTML bol ovplyvňovaný vývojom webových prehliadačov, ktoré spätne ovplyvňovali definíciu jazyka. Je charakterizovaný množinou značiek a ich atribútov definovaných pre danú verziu. Medzi značky sa uzavierajú časti textu dokumentu a tým sa určuje význam (sémantika) obsiahnutého textu.

4.2.5.2 Jazyk CSS

Jazyk pre popis spôsobu zobrazenia stránok napísaných v jazykoch HTML , XHTML alebo XML, ktorý do značnej miery zväčšuje možnosti formátovania stránok oproti formátovaniu v spomenutých jazykoch. Hlavným zmyslom je umožniť návrhárom oddeliť vzhľad dokumentu od jeho štruktúry a obsahu.

4.2.5.3 Jazyk JavaScript

JavaScript je interpretovaný programovací jazyk so základnou objektovo orientovanou koncepciou. Klientská verzia tohto jazyka (t.j. verzia pracujúca s DOM, pojmami ako udalosť prehliadača, dokument, okno a pod.) je súčasťou väčšiny všeobecne rozšírených prehliadačov ako sú napr. Internet Explorer a Mozilla Firefox. Vysokú použiteľnosť jazyka pri programovaní webových aplikácií zaručuje najmä jeho možnosť dynamického menenia obsahu, či vzhľadu dokumentu bez nutnosti komunikácie so serverom.

5 Implementácia

Táto kapitola predstavuje naznačenie procesu vývoja systému so všetkými jeho nástrahami a úskaliami. Pojednáva o tom, ako bola použitá kombinácia vývojových prvkov uvedených v prehľade použitej technológie, ako boli vytvorené modelované objekty navrhnuté v analýze, špecifikácii či návrhu a ako boli tieto objekty prevedené na grafický výstup tak aby s nimi bolo jednoducho manipulovateľné.

Projekt ambulantnej ordinácie bol vytvorený kombináciou technológii J2EE (4.2.1), hlavne servlety (4.2.1.1), JSP stránky (4.2.1.2), JSF framework (4.2.1.3) a JDBC (4.2.1.4). Využíva 3-vrstvovú architektúru s dátovou, funkčnou a prezentačnou vrstvou.

5.1 Dátová vrstva

Dátovú vrstvu prezentujú dáta uložené v databáze. Ako databázový server bol pre účely implementácie vybraný server mysql (4.2.3). Koncepcia relačných tabuliek systému vychádza z navrhnutého ER-diagramu (4.1) a je nasledujúca:

- alergia
- bydlisko
- ciselnik
- ciselnikDiagnoz
- ciselnikLiekov
- ciselnikVykonov
- dekurs
- diagnoza
- lekar
- pacient
- poistovna
- recept
- rolesx
- vykon
- zamestnanie

ER-diagram v návrhu obsahuje iba silné entitné množiny a vzťahy 1 ku mnoho. V dôsledku toho návrh tabuliek približne reprezentuje použité entity v ER. Jediný zložitejší vzťah predstavuje vzťah generalizácie, kde entita číselník vznikla zovšeobecnením číselníku liekov a číselníku výkonov. Vzťah generalizácie sa dá do relačnej schémy previesť mnohými spôsobmi. S oboma číselníkmi sa

v systéme pracuje relatívne samostatne, preto bola uvažovaná možnosť vytvoriť dve samostatné tabuľky číselníkov. Keďže však v systéme vznikla potreba práce s výkonmi, ktoré predstavujú ako výkony tak lieky, musela byť vytvorená samostatná tabuľka číselník a zároveň oba špeciálne číselníky. Tie sú previazané s generalizujúcou tabuľkou prostredníctvom cudzích kľúčov.

5.2 Funkčná vrstva

5.2.1 Pripojenie k databáze

Ako už bolo spomenuté, program je vytvorený kombináciou technológií J2EE. Jedna z nich, JDBC predstavuje rozhranie pre pripojenie programov v Jave k širokému spektru sql databáz. Pre prepojenie medzi dátovou a funkčnou vrstvou však nestačí iba samotné JDBC. Potrebný je aj driver Mysql Connector, ktorý konvertuje JDBC volania do sieťového protokolu Mysql databáze. Šíri sa ako Java archív a bolo potrebné ho nahrať do /lib adresára webového servru. Činnosti v systéme spojené so správou databázy riadia triedy v balíku dbase. Vytvorenie databázového spojenia zabezpečuje trieda DatabaseConnection. Spojenie je tak prístupné prostredníctvom vlastnosti connection objektu DatabaseConnection. Spojenie je možné uzatvoriť aplikovaním metódy `close()` na objekt. V prípade prechodu na iný databázový server sú v systéme pripravené aj drivre pre databáze Oracle či Sybase.

5.2.2 Zabezpečenie systému

Pre beh informačného systému bol použitý aplikačný server Apache Tomcat (4.2.2). Tomcat využíva súčasť nazývanú realm pre potreby zabezpečenia. Realm predstavuje skupinu informácií o užívateľoch a ich roliach. Pre potreby uloženia týchto údajov do databázy bol využitý typ realmu zvaný JDBCRealm[9][11]. Použitie tohto typu realmu si vyžiadalo vytvorenie databázovej tabuľky rolí v databáze. V tejto tabuľke sú umiestnení užívatelia systému a ich role, v ktorých v systéme vystupujú. Ambulantný systém si vyžiadal existenciu dvoch rolí. Rola administrátor umožňuje do systému pridávať a rušiť užívateľské účty. Túto rolu má pridelenú jediný lekár, ktorý je zároveň predvytvorený v systéme. Rola manager je rolou lekárov, ktorí chcú so systémom pracovať. Nemá však právo pridávať a rušiť cudzie účty, jedine editovať ten svoj. Takýmto spôsobom je v systéme ďalej vytváraný každý ďalší lekár. Pri použití JDBCRealm je zároveň nutná aj ďalšia databázová tabuľka, ktorá uchováva užívateľské mená a heslá prihlásených lekárov. V ambulantnom systéme je ňou databázová tabuľka lekár.

Po vytvorení JDBCRealm ako správcu uživateľských účtov bolo nutné definovať chránené oblasti webu. Pre prístup k nim je potrebná autentifikácia. V ambulantnom systéme je definovaná chránená oblasť pre lekára, ktorá je daná relatívnou cestou /secure/*.To znamená, že všetky súbory

v adresári secure si vyžiadajú lekárovu autentifikáciu. Pre ambulantný systém bola vybratá digest autentifikácia, ktorá je založená na súhrnom výbere. Spočíva v tom, že tento výber je vytvorený ako hash (použitím MD5 kryptovacieho algoritmu) mena, hesla, uri, http typu požiadavky (get, post, put, delete) a náhodne generovanej unikátnej hodnoty poskytnutej servrom.

Server Tomcat je nakonfigurovaný tak, že ak chceme pristúpiť do chránenej oblasti začne dáta z nej prenášať prostredníctvom protokolu https. Https je nadstavba http, kde sú dáta prenášané šifrovane pomocou ssl[14].

Ssl je protokol, ktorý poskytuje zabezpečenú komunikáciu šifrovaním a autentizáciu komunikujúcich strán. Používa klasickú kryptografiu s párom verejný kľúč súkromný kľúč. Klient pošle serveru požiadavku na ssl spojenie, server odpovie na túto požiadavku a zašle certifikát servru (certifikát predstavuje elektronicky podpísaný verejný kľúč). Podľa prijatého certifikátu si klient overí autentičnosť serveru. Na základe doposiaľ prijatých informácií klient vytvorí základ šifrovacieho kľúča, zašifruje ho kľúčom verejným a pošle serveru. Ten využije svoj súkromný kľúč na rozšifrovanie základu šifrovacieho kľúča. Z tohto základu ako server, tak klient vygenerujú hlavný šifrovací kľúč. Klient a server si navzájom potvrdia, že od tohto okamihu bude komunikácia šifrovaná vytvoreným kľúčom. Je tak ustanovené zabezpečené spojenie šifrované vytvoreným kľúčom.

5.2.3 Spôsob implementácie systému

Implementácia projektu vychádzala z postupov pri vytváraní JSF aplikácie. Integrácia JSF do systému do značnej miery mení spôsob fungovania celej aplikácie v porovnaní s projektom v JSP + Servlety.

5.2.3.1 Vytvorenie modelovaných objektov

V úvodnej fáze implementácie boli vytvorené triedy reprezentujúce modelované objekty s ich vlastnosťami. Takto vznikli v systéme triedy Pacient, Lekar, Vykon, Dekurs, Alergia, Diagnoza, CiselnikVykonov, CiselnikDiagnoz, CiselnikLiekov, ale aj mnohé iné. Ich vlastnosti sa zhodujú v prevažnej miere s atribútmi databázových tabuliek. Je tomu tak preto, že jednotlivé záznamy tabuliek databázy sa pri chode systému na vytvorené objekty podľa potreby mapujú.

5.2.3.2 Vytvorenie reakcií na udalosti

Celý chod JSF aplikácie je riadený udalosťami (ActionEvent - vyvolaná pri odoslaní formulára a ValueChangeEvent - vyvolaná pri zmene hodnoty nejakého prvku formulára).

Preto bolo nutné v ďalšej časti nadefinovať triedy listenerov, ktoré tieto udalosti spracovávajú. V ambulantnom systéme sú tieto triedy rozpoznateľné podľa prípony Bean v názve triedy. Tieto „beany“ reagujú na udalosti formulárov webovej aplikácie. V projekte ambulantného systému sú takto definované nasledujúce beany:

- pacientBean – reaguje na udalosti vyvolané správou kartotéky, teda na načítanie údajov s kartotéky, úpravu pacienta a jeho anamnézy, mazanie pacienta
- pacientCreateBean – reaguje na udalosti vyvolané pri vytvorení karty nového pacienta
- doctorBean – reaguje na udalosti vyvolané pri administrácii užívateľských účtov systému
- vykonyBean- reaguje na udalosti správy výkonov daného pacienta
- dekursBean- reaguje na udalosti správy dekursov daného pacienta
- beany pre správu číselníkov CiselnikDiagnozBean, CiselnikVykonovBean, CiselnikLiekovBean
- beany pre spracovávanie udalosti filtra tabuľky pacientov

5.2.3.3 Deklarácia riadenia beanov

Jednotlivé „beany“, teda triedy pre reakcie na udalosti formulárov boli v ďalšej časti deklarované v súbore faces-config.xml (5.4.1). Udialo sa tak preto, aby ich rozpoznali jednotlivé jsp stránky vyvolávajúce udalosti. Deklarácia beanov prebieha taktiež preto, aby sa určil obor platnosti daného beanu. Všetky beany v ambulantnom systéme majú platnosť session, to znamená, že sú prístupné v rámci všetkých požiadaviek registrovaného doktora.

5.3 Prezentačná vrstva

5.3.1 Komponentová štruktúra

Komponenty JSF predstavujú základné prvky formulárov ako sú textové polia, list-boxy, check-boxy, selecty, tabuľkové štruktúry a mnohé iné. JSF im dodáva pridanú funkcionalitu, tak aby bolo možné pohodlne pracovať s ich hodnotami. Dopĺňa tiež rozhodovacie procesy, či dané komponenty zobrazovať, možnosť zamedziť ich dopĺňaniu, všetko bez použitia javascriptu. V programe ambulantnej ordinácie bola použitá prakticky celá škála komponentov JSF.

Komponentová základňa JSF je však do značnej miery obmedzená. Preto vznikajú rôzne podporné projekty, ktoré si kladú za cieľ doplniť komponentovú štruktúru JSF o prvky tak, aby sa s nimi pracovalo podobne ako s komponentmi ostatnými. Pri vývoji systému bola preto aplikácia obohatená o komponentovú štruktúru projektu MyFaces (4.2.1.3). V ambulantnom systéme sú takto pridané myfaces komponenty inputCalendar a tabbedPane. InputCalendar sa používa pri práci s dátumom v systéme a tabbedPane pre prepínanie medzi kontextom bez nutnosti interakcie so serverom.

Ambulantný systém je zo základnej podstaty založený na vkladaní a spracovaní údajov, kde sa veľký dôraz kladie na ich validitu. Systém JSF je na tento spôsob spracovania vhodný, pretože umožňuje ku komponentom definovať vlastné validátory. Faces Servlet vykonáva pred spracovaním formulárových prvkov overenie ich hodnôt pomocou validátora komponenty. Validátor má

definovaný vzor pomocou regulárneho výrazu, ktorý porovná s hodnotou. Ak hodnota sedí na regulárny výraz môže byť uplatnené navigačné pravidlo (5.3.3). Ak nie, hodnoty sa nespracujú a systém zostáva na aktuálnej stránke, pričom sa vypíše chybové hlásenie. V ambulantnom systéme boli takto použité validátory rodného čísla, ktoré musí mať formu yyyyyy/xxx resp. yyyyyy/xxxx, validátor všeobecných názvov mien povoľujúci písmená abecedy a medzeru. Ďalej napríklad validátor psč so štruktúrou 5-miestneho čísla a mnohé iné. Nie vždy je však použitie týchto nástrojov možné a musia sa hľadať náhradné riešenia. Napríklad komponent kalendár obsahuje chybu, ktorá nedovoľuje použiť validátor vlastný, pretože sa pri chybe v dátume vyvolá validátor implicitný. Použitie kalendáru tiež znemožňuje použiť relatívne pozicovanie elementu, v ktorom je kalendár umiestnený. Chybovosť komponent zrejme zbrzdila napredovanie projektu MyFaces a preto sa v súčasnosti preferujú projekty iné. Do popredia sa dostáva projekt Woodstock, dokonca prostredie Netbeans 6 má v sebe grafický editor na prácu s JSF 1.2 + Woodstock.

Ambulantný lekár veľmi často spracováva rôzne dokumenty, spomeniem dekursy, recepty, preto bol do systému pridaný html textový editor. Nie je to však komponent Myfaces. Pretože editor tohto projektu sa neosvedčil, pre svoje chybné správanie, bol použitý javascriptový editor projektu TinyMCE.

TinyMCE predstavuje WYSIWYG editor, ktorý umožní definovať štýl dokumentu prostredníctvom nadpisov, zvýraznení, farieb atď.. Jeho výstupom je html kód. Keďže pracuje v režime WYSIWIG, uchováva si zobrazené formátovanie.

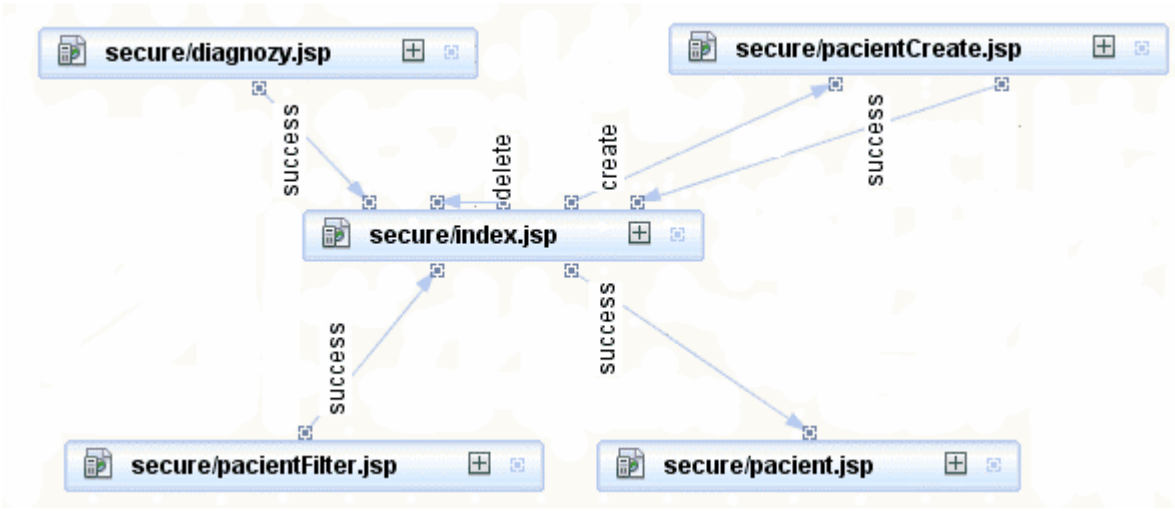
5.3.2 Vytvorenie jsp stránok

Jsp stránky vytvárajú prezentačnú vrstvu programu ambulantnej ordinácie. Obsahujú v sebe ako normálne html značky, tak aj značky komponent JSF. Väčšina týchto stránok, je umiestnená v chránenej oblasti, aby boli prístupné len po lekárovej autentizácii. Výnimku tvorí len prihlasovací formulár, ktorý musí byť umiestnený verejne. Vzhľad jednotlivých komponent je definovaný kaskádovým štýlom (4.2.5.2). Hoci je JSF pomerne bohatý nástroj, nezaobíde sa bez použitia klientského javascriptu (4.2.5.3). V projekte bol javascript použitý hlavne pri práci s tabuľkovými štruktúrami, aby každá udalosť tabuľky nevyvolala interakciu so servrom. Javascript dokáže efektívne pracovať aj s oknami prehliadača, preto bol využitá aj táto jeho vlastnosť.

5.3.3 Navigácia systému

Samotný systém využíva pre navigáciu medzi JSP stránkami systém hypertextových odkazov. Tak sa to deje v prípade, že neprenášajú medzi sebou žiadny kontext. Pretože však v systéme vznikla potreba prenášania kontextu a pomerne veľkého, najmä čo sa týka spracovania formulárových dát, ale aj nutnosť overovania formulárov, museli byť definované navigačné pravidlá. Navigačné pravidlá predstavujú systém navigácie poskytovaný JSF. Definujú zdroj udalosti, udalosť a cieľ udalosti.

Ukážka časti navigačných pravidiel systému ambulantnej ordinácie je zobrazená na obrázku 5.1. Obrázok ukazuje prechod zo stránky index.jsp na stránku pacientCreate.jsp, teda po vyvolaní akcie vytvor kartu pacienta (create) sa namiesto kartotéky pacientov (index.jsp) zobrazí stránka pre vytvorenie pacienta (pacientCreate.jsp). Naopak po vyvolaní udalosti success pri vytvorení karty pacienta sa v prehliadači vytvorí požiadavka na zobrazenie kartotéky. Napr. po vyvolaní akcie dalete na index.jsp sa kontext neprenáša a zostane v rámci rovnakej stránky, ktorú predstavuje kartotéka pacientov.



Obrázok 5.1: ukážka navigačných pravidiel

5.4 Konfiguračné súbory systému

Táto kapitola popisuje súbory, ktoré majú zásadný vplyv na spôsob fungovania aplikácie. Riadia činnosť objektov, pravidiel či jsp stránok systému.

5.4.1 Web.xml

V projekte ambulantnej ordinácie je vo web.xml umiestnené mapovanie servletov na url-adresy, pomocou ktorých sú servlety na webe dostupné. S pridaním frameworku JSF, však v projekte ustúpila nutnosť vytvárania servletov a skôr sa pristúpilo k vytváraniu JSP stránok pomocou komponentov JSF.

Súbor web.xml ďalej v projekte definuje používanie filtrov na súbory projektu. V projekte faces-redirect-filter presmerováva požiadavky na súbory s príponou .jsp na súbory s príponou .faces, aby sa tak zabránilo rozpoznaniu použitej technológie vývoja. Pridaním projektu MyFaces (4.2.1.3) do systému pre potreby rozšírenia použiteľných komponentov bolo nutné pridať ďalší filter nazvaný MyFacesExtension Filter. Ten mapuje všetky jsp stránky na Faces Servlet, aby tento servlet mohol vybudovať komponentový strom daných stránok.

Web.xml ďalej definuje pre server chránené oblasti aplikácie popísané v časti zabezpečenie systému (5.2.2). Pre prístup k nim sa musí lekár autentizovať. To nie je potrebné robiť stále pri dosahu na chránenú oblasť. Stačí to urobiť raz prostredníctvom prihlasovacieho dialógu, ktorý je definovaný vo web.xml v tagoch <login-config>. Prihlasovací dialóg sa zobrazí pri vstupe do systému, alebo pri odkaze na chránenú oblasť, ak práve užívateľ nie je prihlásený alebo uplynul časový limit prihlásenia. Časový limit je opäť definovaný vo web.xml v tagoch session-config a je určený na čas pol hodiny. Súbor ďalej udáva uzol, ktorým sa vstupuje do systému, je ním index.jsp- vytvárajúci kartotéku pacientov. Taktiež určuje parametre pre Myfaces komponenty.

5.4.2 Server.xml

Server Apache Tomcat konfiguruje xml súbor server.xml. Konfigurovať server bolo v projekte nutné kvôli zabezpečeniu systému (5.2.2). Server.xml definuje vo svojom tele typ realmu, teda akým spôsobom bola aplikácia zabezpečená. Ako už bolo spomenuté, pre potreby zabezpečenia bol vybratý JDBCRealm. V server.xml je teda ako referenčný typ vybratý tento realm spoločne s jeho prístupovými údajmi do databázy. Konkrétne meno a heslo do databáze, typ pripojenia pomocou JDBC, tabuľky rolí, a užívateľských účtov. Server.xml taktiež definuje protokol prístupu https k chráneným oblastiam a vytvorený pár verejný-súkromný kľúč.

5.4.3 Faces-config.xml

Faces-config je xml súbor popisujúci správanie JSF aplikácie. Obsahuje deklarácie jednotlivých beanov (5.2.3.3) a určuje navigačné pravidlá (5.3.3). V súbore sú tiež deklarované validátory a konvertery hodnôt, ktoré kontrolujú platnosť alebo konvertujú hodnoty formulárov.

6 Chod a fungovanie systému

Systém ambulantnej ordinácie predstavuje aplikáciu umožňujúcu lekárovi základnú administráciu úkonov, ktoré vykonáva v dennej praxi. Dáta, s ktorými pracuje je potrebné chrániť, preto je systém dostupný iba po prihlásení. Aby sa mohol lekár prihlásiť, je potrebné požiadať administrátora o založenie účtu. Po získaní prihlasovacích údajov je možné pristupovať k systému v roli lekár.

6.1 Kartotéka pacientov

Vstupným bodom aplikácie je kartotéka pacientov. Po prvom prihlásení do systému je prázdna. Správu kartotéky zaručujú úkony definované pod kartotékou. Tieto úkony zahrňujú pridávanie, rušenie a editáciu karty pacienta. Aby mohla byť karta existujúceho pacienta zrušená alebo editovaná, musí byť označený pacient v kartotéke, ktorého kartu chce lekár zrušiť resp. editovať. Kartotéku pacientov si môže lekár upravovať vlastným spôsobom tak, aby vyhovovala jeho predstavám o zobrazení dát o pacientoch. Celkový výber pacientov a údajov o nich je možné triediť a filtrovať. Triedenie pacientov zaručuje stlačenie popisu jednotlivých vlastností pacienta. Tým sa určí, podľa ktorého atribútu si želá pacientov triediť. Triediaci mechanizmus zoraďuje pacientov vzostupne podľa abecedy, resp. čísla ak je daná položka číselná. Výber vlastností zobrazovaných o pacientoch zaručuje záložka Stĺpce v menu karty Kartotéka pacientov. Definuje možné a zobrazené vlastnosti pacienta. Aby bol v kartotéke umožnený rýchly prístup k hľadanému pacientovi poprípade k skupine pacientov s rovnakou vlastnosťou, je na kartotéku možné uplatniť filter zobrazených údajov. Ten je prístupný pod záložkou Filter výberu v menu karty Kartotéka pacientov. Filter umožňuje selektovať pacientov podľa typu starostlivosti, pohlavia, miesta bydliska, dátumu narodenia či veku.

6.2 Správa pacientov

Správu pacientov predstavujú úkony spojené s definovaním a editovaním vlastností pacienta, anamnéz, výkonov, či záznamov o pacientovi.

6.2.1 Vytvorenie, úprava karty pacienta

Pri zadaní úkonu vytvorenie karty pacienta sa zobrazí karta, kde sú uvedené položky, ktoré sa o pacientovi uchovávajú . Niektoré z nich je potrebné vyplniť povinne, aby bola karta vytvorená. Vytvorený záznam obsahuje osobné údaje o pacientovi, o jeho zamestnaní, či bydlisku. Po vytvorení karty je možné pristupovať k lekárskym záznamom o danom pacientovi . Uložené údaje o pacientoch v karte je možné potom ľubovoľne upravovať. Väčšina položiek formulárov karty má preddefinovaný vzor, ktorý musia vložené údaje spĺňať. Jedná sa napríklad o formát rodného čísla či psč.

6.2.2 Anamnéza pacienta

Anamnéza pacienta predstavuje ďalšiu záložku v karte pacienta. Uschováva údaje o jeho zdravotnom stave. Je možné definovať celkovú anamnézu pacienta, teda či pacient má nejaké zdravotné problémy trvalého charakteru. Systém ďalej umožňuje zaznamenať alergie pacienta a jeho trvalé diagnózy.

6.2.3 Dekurs

Záznam o vyšetrení pacienta sa zapisuje v spodnej časti záložky dekurs karty pacienta do textového editoru. K záznamu je nutné priradiť diagnózu pacienta. Diagnózu je nutné zapísať číselne, pre potreby jej vyhľadania je možné zobraziť číselník diagnóz. Pravá časť záložky dekurs obsahuje históriu záznamov v čase spolu s priradenými diagnózami. Je možné vyhľadať ľubovoľný starý záznam a zmeniť jeho údaje.

6.2.4 Výkony

Záložka výkony v karte pacienta umožňuje lekárovi vykázať výkon o prevedenej zdravotnej starostlivosti. V systéme sa vykazujú ako výkony samotné, tak výkony vo forme predpísaných liekov. Nahrávanie výkonov úzko súvisí s číselníkom výkonov a číselníkom liekov. Lekár k danému výkonu vyberie s číselníka prevedený výkon resp. liek. Do výkonu sa tak automaticky vyplnia hodnoty o skupine výkonu a jeho kód, body a názov daného výkonu. Ďalej je potrebné uviesť diagnózu a dátum prevedeného výkonu. Prevedené výkony je možné v tabuľke výkonov ľubovoľne pridávať a rušiť.

6.2.5 Lieky

Systém umožňuje zaznamenávať predpísané lieky pacienta. Pri ich spracovávaní využíva hodnoty z číselníkov liekov. Konkrétne informácie o jeho názve, liekovej forme, výrobcovi, veľkosti úhrady za daný liek či maximálnu možnú cenu lieku. S predpísaným liekom previaže diagnózu pacienta a dátum predpísania lieku.

6.3 Správa lekára

Pod záložkou lekár v hlavnom menu je možné nájsť údaje spojené s prihláseným lekárom. Tie môže prihlásený lekár meniť podľa svojej potreby, prípadne tak aj svoje registračné údaje.
Záložka odhlásenie odhlási lekára so systému. V administračnom režime správa lekára obsahuje aj možnosti pridávania nových užívateľov systému.

6.4 Prehľady

Systém umožňuje zobraziť v záložke prehľady hlavného menu záznamy o prevedených výkonoch, liekoch či diagnózach komplexne, v rámci celého spektra pacientov. Hlavným obmedzujúcim faktorom týchto zobrazení je dátum prevedenia.

6.5 Číselníky

Lekár sa pri svojej činnosti veľmi často odkazuje na číselníky hodnôt. V záložke číselníky je možné zobrazovať tie v systéme definované. Sú to číselník liekov a číselník výkonov. Oba definujú oficiálne číselníky vydávané Všeobecnou zdravotnou poisťovňou(VZP). Číselník diagnóz nie je číselníkom VZP, ale je pre potreby systému vytvorený pomocou MKCH- medzinárodnej klasifikácie chorôb.

7 Zhodnotenie

Táto kapitola ako už názov hovorí hodnotí dosiahnuté výsledky práce, architektúru systému a použité technológie. Navrhuje možnosti rozšírenia systému.

7.1 Dosiahnuté výsledky

Vytvorená aplikácia sa približuje činnosti ambulantných systémov bežne používaných v praxi v zjednodušenej podobe. Obsahuje ich základné funkčné súčasti ako kartotéku pacientov, výkony, dekursy, správu pacientov či správu číselníkov. Snahou bolo, aby prístup k ním bol intuitívny a nezabral užívateľovi systému mnoho času na pochopenie.

Vytvorený ambulantný systém je koncipovaný ako webový, takže umožňuje viac-užívateľský prístup, čo je jednoznačne jeho výhodou. Systém sa po umiestnení na verejný aplikačný server stáva prístupný všetkým užívateľom siete internet. To predstavuje veľké pozitívum, pretože by doňho lekár mohol pristupovať z ľubovoľného miesta siete. Tak by v systéme mohol pracovať napríklad z pohodlia svojho domova. Väčšina súčasných ambulantných systémov však takúto koncepciu doposiaľ nevyužíva. Je tomu tak zrejme v dôsledku pomalej interakcie klienta so vzdialeným servrom pri prenose veľkého množstva dát. Objavujú sa však už aj nadstavby týchto systémov, ktoré vzdialený prístup umožňujú.

Systém bol vytvorený na platforme J2EE. Tá poskytuje z tejto osobnej skúsenosti s ňou dostatočné zázemie pre vytváranie rozsiahlych informačných systémov. Technológie J2EE použité v systéme umožňujú vytvárať na sebe relatívne nezávislé súčasti aplikácie. Takto vytvorený produkt je vo výsledku dobre spravovateľný a modifikovateľný. Objektovosť technológie umožňuje ľahko modelovať zobrazovanú realitu do výsledného systému.

7.2 Možnosti rozšírenia

Vytvoriť ambulantný systém pre praktického lekára je práca pomerne náročná. Lekár vo svojej každodennej praxi vykonáva množstvo úkonov, ktoré je potrebné v systéme zachytiť. Okrem toho by mal systém zabezpečovať aj interakciu s pacientmi, poisťovňami či systémami inými, vyhovovať systému zdravotníctva krajiny. Všetko to prináša veľké možnosti rozšírenia vytvoreného systému.

Všetky významné zdravotnícke systémy v rámci ČR splňujú v súčasnosti dátový štandard Ministerstva zdravotníctva nazvaný DASTA. Štandard slúži k predávaniu dát medzi zdravotníckymi systémami. Vytvára dátové bloky vo formáte xml, ktoré umožňujú prenos údajov o pacientoch, číselníkoch, laboratórnych príručkách atď. Vytvorenie ambulantného systému podľa tohto štandardu

by zrejme znamenalo možnosť prepojiť systém s množstvom zdravotníckych zariadení a systémov. Touto interakciou by sa napríklad zefektívnil prenos dát medzi ambulanciami a laboratóriami.

Systém by sa po implementovaní vhodných sieťových nástrojov mohol zúčastňovať na projekte IZIP. IZIP predstavuje vytváranie elektronických zdravotných knižiek pacienta. Pacientovi sú tak prístupné jeho zdravotné záznamy. Stáva sa nositeľom svojich zdravotníckych dát, ktoré môže v prípade potreby predať urýchlene ošetrujúcemu lekárovi.

V praxi užívané ambulantné systémy umožňujú vyúčtovanie zdravotníckej starostlivosti zmluvným poisťovniam. Vyúčtovanie sa riadi metodikou pre zadováženie a predávanie dokladov VZP ČR. Doklad predstavuje zápis pre uskutočnenie úhrady poskytnutej zdravotnej starostlivosti. Pre implementáciu systému vyúčtovania do aplikácie by bolo potrebné systémom vystavovať spomínané doklady o prevedených výkonoch.

Systém v súčasnej podobe pracuje s číselníkmi vo veľmi obmedzenej miere. Prakticky implementuje iba tri z veľkého množstva poskytovaných. Možné rozšírenie vzniknutého systému by teda mohlo obsahovať komplexnú správu číselníkov či automatický import číselníkov novo sprístupnených.

Funkčnosť systému je do značnej miery obmedzená. V budúcnosti je ho možno rozšíriť o moduly prehliadok, očkovaní, umožniť aby bol systém použiteľný aj špecializovanými lekármi.

Záver

Splnenie cieľa bakalárskej práce, ktorým je vytvorenie ambulantného systému si vyžiadalo pochopenie princípov fungovania týchto systémov. Implementačná technológia si zase vyžiadala pochopenie výstavby informačných systémov na platforme J2EE. Obe tieto úlohy boli náročné a zabrali značný čas pri vývoji. Vytvorený systém hlavne technologicky predstavuje zaujímavú alternatívu k reálnym ambulantným systémom. Pre potrebu vzdialenej správy, či viac-užívateľského prístupu s takýmto systémom je celkom možné nasadenie technológií platformy J2EE na vývoj ambulantných systémov v budúcnosti.

Poskytnutý časový priestor na vývoj aplikácie, ale aj malé skúsenosti s platformou vývoja spôsobili, že sa systém svojou funkčnosťou nemôže reálnym systémom zatiaľ rovnať. Táto práca sa preto skôr zamerala na zoznámenie s problematikou tvorby ambulantného systému pomocou technológii J2EE. Naznačila akým smerom vývoj smeroval a predostrela ďalšie možnosti rozšírenia systému.

Literatúra

[1] HALL, M.: *Java servlety a stránky JSP*, Praha, Neocortex, spol. s r.o., 2001

[2] KŘENA, B., KOČÍ, R.: *Úvod do softwarového inženýrství (IUS), 3. kapitola: Modely životníhocyklu softwaru*, Brno, 2006 [cit. 2008-07-20]

[3] *J2EE* [online]. 2008 [cit. 2008-07-10].
URL <http://interval.cz/clanky/j2ee-net-a-vyvoj-rozsahlych-systemu-1>

[4] *Java- Servlets* [online]. 2008 [cit. 2008-07-12].
URL <http://interval.cz/clanky/java-servlets-predstavenie-technologie>

[5] *Java Server Pages pro všechny* [online]. 2008 [cit. 2008-07-12].
URL <http://interval.cz/clanky/javaserver-pages-pro-vsechny>

[6] *J2EE tutoriál* [online]. 2008 [cit. 2008-01-12].
URL <http://java.sun.com/j2ee/1.4/docs/tutorial/doc>

[7] *JSF- Java Server Faces*[online]. 2008[cit.2008-07-12].
URL <http://nb.vse.cz/~zelenyj/it380/eseje/xkavm06/jsfxkavm.htm>

[8] *JSF , JSP, Servlets* [online]. 2008 [cit. 2008-07-12].
URL <http://www.coreservlets.com>

[9] *JDBCRealm* [online]. 2008 [cit. 2008-07-05].
URL: <http://interval.cz/clanky/autentizace-pomoci-filtru-ukazkova-aplikace>.

[10] *MyFaces Tomahawk*[online]. 2008 [cit. 2008-07-09].
URL <http://myfaces.apache.org/tomahawk>

[11] *Apache Tomcat security*[online]. 2008 [cit. 2008-07-05].
URL <http://tomcat.apache.org/tomcat-5.5-doc/realm-howto.html>

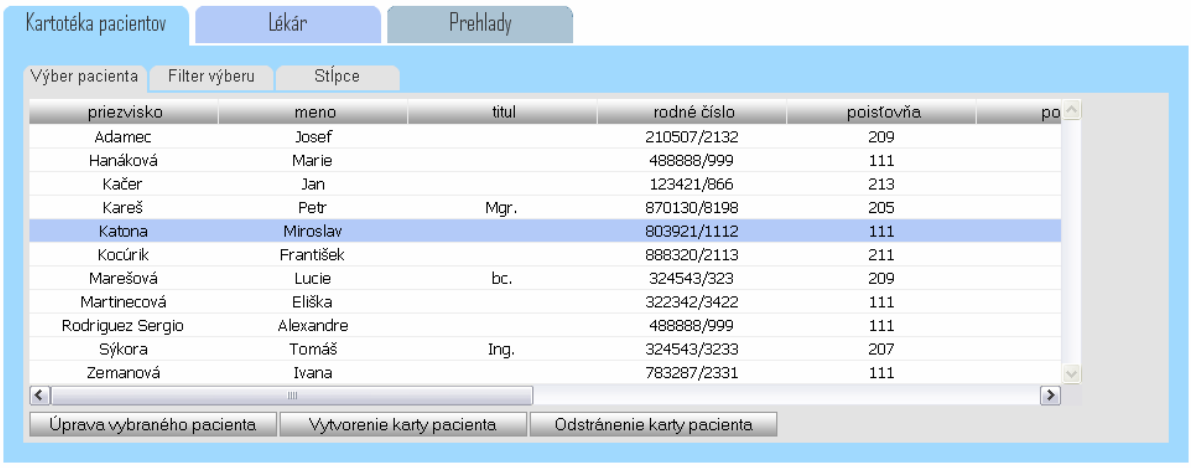
[12] *IDE Netbeans* [online]. 2008 [cit. 2008-07-01].
URL <http://www.netbeans.org/kb/articles/jAstrologer-intro.html>

[13] *JSF life cycle* [online]. 2008 [cit. 2008-07-20].
URL <http://www.ibm.com/developerworks/library/j-jsf2>

[14] *SSL* [online]. 2008 [cit. 2008-07-20].
URL <http://cs.wikipedia.org/wiki/Secure_Sockets_Layer>

Zoznam príloh

Obrazovky systému



Obrázok 1: kartotéka pacientov

Obrázok 2: výkony