České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra počítačů

BAKALÁŘSKÁ PRÁCE

Informační systém ozdravných pobytů zdravotní pojišťovny

Radoslava Jandová

Vedoucí práce: Ing. Martin Bloch, CSc.

Studijní program: Softwarové technologie a management

Obor: Softwarové inženýrství

31. prosince 2012

Prohlášení:

Prohlašuji, že jsem bakalářskou práci ***Informační systém ozdravných pobytů zdravotní pojišťovny*** vypracovala samostatně pod vedením Ing. Martina Blocha, CSc. a v seznamu použité literatury jsem uvedla všechny použité literární a odborné zdroje.

Nemám závažný důvod proti užití tohoto školního díla ve smyslu §60 zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorských).

V Praze dne 31. prosince 2012 Radoslava Jandová

Abstrakt:

Předmětem bakalářské práce ***Informační systém ozdravných pobytů zdravotní pojišťovny*** je vytvoření informačního systému ozdravných pobytů dětí a mládeže, který nabízí sběr dat o účastnících pobytů a náhled podle zvolených kritérií.

Součástí práce je návrh architektury a implementace aplikace.

Klíčová slova:

Webový framework, aplikační server, Java EE, informace, databáze, informační systém, procesy.

Obsah

Příloha č. 1 – CD s elektronickou verzí bakalářské práce

Příloha č. 2 – Pokyny pro instalaci.

Příloha č. 3 – Framework Simpleton – uživatelský manuál

Příloha č. 4 – Uživatelský manuál

KAPITOLA A.

V první části mé bakalářské práce jsem se zabývala obecným významem informačních systémů a jejich vývojem.

1. Úvod do informačních systémů

V dnešní době se stále častěji setkáváme s pojmem *informační systém* (dále jen „IS“) a jeho využití nabývá stále většího významu. V praxi se sice ještě můžeme setkat s názorem *„Nikdy jsme žádný IS neměli a nějak jsme fungovali, takže žádný IS nepotřebujeme.“*, ale opravdu žádný IS neměli? IS nemusí být jen automatizovaný systém realizovaný pomocí informačních technologií. Za IS lze považovat stanovení jakýchkoli principů vedoucích k optimalizaci řízení firmy a produktivity její práce.

Informační systémy zajišťují zejména následující činnosti:

- sběr informací,
- uchování informací,
- aktualizace informací,
- prezentace informací,
- archivace informací.

Význam pojmu „informační systém“ [3]

Pokud se začneme zajímat o význam pojmu „informační systém“, zjistíme, že přesná definice v podstatě neexistuje a ani ji nelze jednoduše vytvořit. IS mají velmi širokou škálu použitelnosti a každý uživatel či tvůrce IS zdůrazňuje jiné aspekty a používá jinou terminologii. Musíme tedy hledat, co je pro IS společné a v tomto ohledu lze IS teoreticky chápat jako ***systém vzájemně propojených informací a procesů, které s těmito informacemi pracují.***

Pod pojmem ***informace*** rozumíme data, která slouží zejména pro rozhodování a řízení v rozsáhlejším systému. Souhrn těchto informací se nazývá ***databáze. Databázový systém*** pak tvoří jádro IS.

Pod pojmem ***procesy*** rozumíme posloupnost transformací, které zpracovávají informace do systému vstupující na informace ze systému vystupující. Velmi zjednodušeně můžeme říci, že procesy jsou činnosti zabezpečující sběr, uložení, zpracování a distribuci informací.

Nezanedbatelnou položkou IS je ***okolí***. Tím rozumíme jednak objekty, které změnou svých vlastností ovlivňují samotný systém, a dále objekty, které naopak mění své vlastnosti v závislosti na systému.

Pokud bychom teoretický popis IS vyjádřili praktickou terminologií, pak můžeme říci, že IS je softwarové vybavení firmy, které je schopno na základě zpracovávaných informací řídit vnitřní činnosti firmy nebo poskytovat informace potřebné pro tyto činnosti.

2. Informace

2.1. Výklad pojmu informace [5]

Slovo informace tvoří dnes běžnou součást naší slovní zásoby. Používáme je tak často, že už se ani nezamýšlíme nad jeho původem ani významem a spíše než exaktní definici jsme zpravidla schopni ukázat, že chápeme úlohu informace v každodenním životě na konkrétních příkladech. Pojem informace patří dnes k nejobecnějším kategoriím současné vědy a filozofie a podle toho, v jakém vědním oboru nebo oblasti lidské činnosti se používá, jsou aplikovány specifické přístupy ke zkoumání informace a jsou k dispozici různé způsoby jejího definování.

Slovo ***informace*** má původ v latinském výrazu ***in-formatio***, což znamená utváření, ztvárnění, vtištění formy či tvaru. Nejprve se toto slovo používalo pouze ve výrobě, postupně se ale začalo používat i ve smyslu „utváření mysli“. Odtud dalším významovým posunem se pojmem „informace“ začaly označovat zprávy nebo sdělení. Dnes jde o velmi široký a mnohoznačný pojem, který se užívá v různých významech.

Obecně je informace proces vnímání a poznávání vlastností a uspořádání objektů kolem nás. Podle fyzikální definice je informace schopnost organizovat, nebo v organizovaném stavu udržovat. Pro živé bytosti můžeme definici informace doplnit také tím, že informace je odpovědí na otázku. A samozřejmě lze najít i esoterické a filosofické výklady informace, které nejsou doloženy žádnými objektivními poznatky. Zde je informace definována jako součást kosmu, která by mohla, jako nějaká inteligentní substance samostatně existovat i mimo naše vědomí a mít nějaké schopnosti.

V užším slova smyslu (viz dále informace inženýrská) je informace chápána jako údaj o reálném prostředí, o jeho stavu a procesech v něm probíhajících. Informace snižuje nebo odstraňuje neuspořádanost (= entropii) systému. Množství informace je dáno rozdílem mezi stavem neurčitosti, kterou měl systém před a po přijetí informace. Jednotka informace je bit a nabývá hodnot 0 a 1.

Aby informace byla pro uživatele přínosem, měla by být
- pravdivá,
- srozumitelná,
- včasná,
- relevantní,
- etická, tzn. že uživatel je oprávněn tuto informaci obdržet.

Záměrně falešná informace se nazývá dezinformace nebo lež. Rozdíl mezi těmito dvěma pojmy spočívá v tom, že lež je používána pasivně, zatímco dezinformace je šířena aktivně. Nezáměrně špatná informace je chyba. Dezinformace, stejně jako chyba vede k nárůstu entropie, zatím co informace vede k poklesu entropie.

2.2. Informace v IS

IS zajišťují zejména následující činnosti:
- sběr informací,
- uchování informací,

- aktualizace informací,
- prezentace informací,
- archivace informací.

Každá informace musí být:
- pravdivá,
- srozumitelná,
- včasná,
- relevantní,
- etická, tzn. že uživatel je oprávněn tuto informaci obdržet.

V procesu řízení rozdělujeme informace podle následujících hledisek [3]:

2.2.1. Časové hledisko
- ***informace o minulosti*** – slouží jako výchozí podklad pro analýzy, pro zjišťování působících faktorů a konkrétního vlivu na řídící proces.
- ***informace o přítomnosti*** – jde zejména o kontrolní a rozhodovací informace, které slouží k přímému zasahování do běžících procesů.
- ***informace o budoucnosti*** – plány, cíle a kritéria, jejich plnění a posuzování.

2.2.2. Hledisko vztahu informace k řídícím činnostem
Informace obsahuje následující prvky:
- plány, cíle a požadovaný stav systému,
- pravidla pro řízení činností systému,
- data o průběhu činností,
- stav okolí systému.

2.2.3. Hledisko původu informace
- ***prvotní informace*** – vycházejí přímo z řídících procesů a vyjadřují stav jednotlivých prvků systému. Tyto informace jsou nutné přímo při průběhu procesů, ale nemají velkou strategickou hodnotu, protože jsou rozsáhlé, ale málo použitelné v obecném přehledu.
- ***druhotné informace*** – informují o probíhajících procesech zprostředkovaně, především spojováním nebo kombinováním prvotních informací. Tyto informace jsou důležité pro strategické rozhodování a řízení.

Uvedená klasifikace nám pomáhá určit, zda máme shromážděny všechny potřebné informace pro řídící činnosti.

3. Databáze

3.1. Výklad pojmu databáze

Databázový systém (datová základna) je uspořádaná množina informací (dat) uložená na paměťovém médiu. Je tvořen třemi částmi
- ***databáze*** = množina záznamů nebo poznatků,

- ***schéma*** = popisuje objekty, které jsou zastoupeny v databázi a vztahy mezi nimi. Existuje několik způsobů, jak reprezentovat schéma – to se nazývá modelování datové struktury.
- ***software*** = počítačový program, který slouží ke správě a dotazování nad databází a je znám jako „systém řízení báze dat“ (tzv. SŘBD, anglicky DBMS = database management system).

Běžně se databázový systém označuje zkráceně databáze, a ačkoli je toto označení nepřesné, používá jej nejen laická veřejnost, ale i odborníci.

Ke vzniku a rozvoji databázových systémů vedla programátory potřeba využívat libovolná firemní data právě pro rozsáhlejší IS. Potřebné informace ale byly většinou uloženy v dílčích souborech, zcela mylně nazývanými databáze, různých softwarových aplikací a pro společné využití byly v podstatě nepoužitelná. Důvodem byly např. tyto problémy hromadného zpracování dat [19]:

- ***programy a data byly navzájem závislé*** – při změně organizace dat, je nutno tyto změny promítnout do všech programů, které s daty pracují.
- ***redundance (nadbytečnost) dat*** - agendy byly pojímány jako relativně izolované části IS bez jakéhokoli sdílení dat. To vedlo k tomu, že stejné údaje o firmě a zaměstnancích mohly být uvedeny i v několika různých souborech.
- ***nekompatibilita dat*** – vznikala s ohledem na skutečnost, že se data v různých evidencích pořizovala v různých časových úsecích a odlišnými metodami.
- ***obtížná dosažitelnost dat*** – s ohledem na nekompatibilitu a nekonzistenci dat bylo pořízení výstupů náročné na zpracování jak z hlediska časového, tak z hlediska pracovní síly, protože odpovědi na různé dotazy byly zpracovávány ručně.
- ***izolovanost dat*** – vzhledem k tomu, že data byla uložena v různých souborech, byla různě organizována a měla různý formát. Tento fakt v podstatě znemožnil souhrnné použití těchto dat pro IS.
- ***problém současného přístupu více uživatelů*** – současné IS jsou určeny pro více uživatelských rolí a systém musí být připraven na koordinaci paralelních procesů, kdy jeden proces může data modifikovat, druhý je číst. Tento problém není možno vyřešit, pokud jsou data uložena izolovaně.
- ***problém ochrany dat před zneužitím*** – běžné SW programy měly a stále mají velmi nízkou ochranu vůči zneužití nebo poškození dat. IS již ale s tímto faktem počítají a ochrana dat je jednou z důležitých součástí IS.
- ***problém integrity dat*** – aby byl IS funkční, je nutno zajistit integritu (celistvost) dat, která na vstupu kontrolována. Pokud by data měla přicházet z více různých souborů, nebylo by možné tuto integritu v reálném čase a s bezchybným výsledkem provést a data by nebylo možno zkompilovat.

3.2. Historie [14]

Předchůdcem databází byly papírové kartotéky. Umožňovaly uspořádávání dat podle různých kritérií a zatřiďování nových položek. Veškeré operace s nimi prováděl přímo člověk. Správa takových kartoték byla v mnohém podobná správě dnešních databází.

Dalším krokem bylo převedení zpracování dat na stroje. Za první velké strojové zpracování dat lze považovat sčítání lidu ve Spojených státech v roce 1890. Paměťovým médiem byl

děrný štítek a zpracování sebraných informací probíhalo na elektromechanických strojích. Elektromechanické stroje se využívaly pro účely zpracování dat další půlstoletí.

Velkým impulsem pro další rozvoj databází byl vývoj počítačů v padesátých letech 20. století. Ukázalo se, že původně univerzální používání strojového kódu procesorů je (nejen) pro databázové úlohy neefektivní, a proto se objevil požadavek na vyšší jazyk pro zpracování dat.

V roce 1959 se konala konference zástupců firem, uživatelů a amerického ministerstva obrany, jejímž závěrem byl požadavek na univerzální databázový jazyk. Výsledkem byla o rok později na konferenci CODASYL publikovaná první verze jazyka COBOL, který byl po mnoho dalších let nejrozšířenějším jazykem pro hromadné zpracování dat.

V roce 1965 byl na konferenci CODASYL vytvořen výbor Database Task Group (DBTG), který měl za úkol vytvořit koncepci databázových systémů. Začaly vznikat první síťové SRBD na sálových počítačích. Jedním z prvních průkopníků databází byl Charles Bachman.

V roce 1971 vydal výbor zprávu The DBTG April 1971 Report, kde se objevily pojmy jako schéma databáze, jazyk pro definici schématu, subschéma a podobně. Byla zde popsána celá architektura síťového databázového systému. Ve stejné době byly vyvíjeny i hierarchické databáze. Jedním z prvních SŘBD byl IMS, který byl vyvinut firmou IBM pro program letu na Měsíc Program Apollo. Systém IMS patří stále k nejrozšířenějším na sálových počítačích.

V roce 1970 se začínají zveřejněním článku E. F. Codda používat také první relační databáze, které pohlížejí na data jako na tabulky. Kolem roku 1974 se vyvíjí první verze dotazovacího jazyka SQL. Vývoj této technologie po 10 letech přinesl výkonově použitelné systémy, srovnatelné se síťovými a hierarchickými databázemi.

V 90. letech 20. století se začínaly objevovat první objektově orientované databáze, jejichž filozofie byla přebírána z objektově orientovaných jazyků. Tyto databáze měly podle předpokladů vytlačit relační systémy. Původní předpoklady se však nenaplnily a vznikla kompromisní objektově-relační technologie.

3.3. Databázové modely

Existuje řada způsobů, jak reprezentovat schéma databáze (= modelovat datové struktury). Datový model je nástroj pro reprezentaci struktury a funkcionality databáze a umožňuje definovat [18]:

- schéma databáze, tj. organizaci dat,
- způsoby ochrany databáze,
- zajištění integrity dat,
- operace s daty.

Z hlediska způsobu ukládání dat a vazeb mezi nimi rozlišujeme následující základní datové modely.

3.3.1. Hierarchický model databáze

První z datových modelů, který byl v minulosti hojně využíván v praxi. Jde o datový model, ve kterém jsou data uspořádána ve stromové struktuře, která ale umožňuje pouze kardinalitu

vztahu 1:N mezi dvěma druhy dat. Každý záznam představuje uzel ve stromové struktuře a vztah mezi záznamy je typu rodič-potomek.

3.3.2. Síťový model databáze

Jde o zobecnění hierarchického modelu. Zatímco hierarchický databázový model strukturuje data do stromu záznamů, kde každý záznam má jednoho rodiče a až N potomků, síťový model umožňuje, aby záznam měl i více než jednoho rodiče. Struktura síťového modelu je popsána obecným grafem.

3.3.3. Relační model databáze

Relační databázový model sdružuje data do tzv. relací (tabulek), které obsahují n-tice (řádky). Tabulka je struktura záznamů s pevně stanovenými položkami (sloupci - atributy). Každý sloupec má definován jednoznačný název, typ a rozsah neboli doménu. Záznam se stává n-ticí (řádkem) tabulky. Pokud jsou v různých tabulkách sloupce stejného typu, pak tyto sloupce mohou vytvářet vazby mezi jednotlivými tabulkami. Tabulky se poté naplňují vlastním obsahem - konkrétními daty.

Kolekce více tabulek, jejich funkčních vztahů, indexů a dalších součástí tvoří relační databázi. Relační model klade velký důraz na zachování integrity dat. Zavádí pojmy referenční integrita, cizí klíč, primární klíč apod.

3.3.4. Objektový model databáze

Filozofie tohoto databázového systému je založena na objektově orientovaném programování. Data jsou reprezentována formou objektů použitých v objektově orientovaném programování.

3.3.5. Objektově-relační model databáze

Tento databázový systém vznikl propojením relačního a objektového modelu.

4. Procesy

4.1. Výklad pojmu proces [11]

Pojem ***proces*** pochází z latinského slova ***processus***. Jde o obecné označení pro postupné a nějak zaměřené děje či změny nebo pro posloupnost stavů nějakého systému. Procesy jsou činnosti obvykle zabezpečující sběr, přenos, uložení, zpracování a distribuci informací. Pro děje náhlé nebo zcela chaotické se pojem proces nepoužívá.

4.2. Pojem „proces“ v informatice [12]

V informatice je pojem ***proces*** (z anglického ***process***) užíván pro spuštěný počítačový program. Program je umístěn v operační paměti počítače v podobě sledu strojových instrukcí vykonávaných procesorem. Proces obsahuje nejen kód vykonávaného programu, ale i dynamicky měnící se data, která proces zpracovává. Jeden program může být v počítači sdílen více procesy s různými daty (například vícekrát spuštěný webový prohlížeč zobrazující různé stránky). Správu procesů vykonává operační systém, který zajišťuje jejich paralelní běh,

přiděluje jim systémové prostředky počítače a umožňuje uživateli procesy spravovat (spouštět, ukončovat atp.).

4.3. „Business“ procesy

V oblasti IS se také setkáváme s pojmem "*business proces*" (z anglického slova *business process*), který je důležitou součástí vývoje a tvorby celého IS. Business proces označuje reprezentaci procesů probíhajících v realitě dané firmy. Model IS se pak skládá z jednotlivých dílčích procesů.

Na rozdíl od standardních procesů operují business procesy nad daty uloženými v databázi a synchronizují se pomocí business událostí jako je např. potvrzení objednávky, odeslání objednávky apod. Algoritmy business procesů se popisují ve speciálních jazycích jako je např. BPEL (Business Process Execution Language).

Pro správný návrh business procesů je nezbytná kompletní analýza činností a vazeb uvnitř firmy, tj. zejména stanovení aktérů, zpracovávaných dat apod. Bez této analýzy by nemohl vzniknout správný funkční model IS ani správná implementace tohoto funkčního modelu. Čím je analýza podrobnější a důslednější, tím se zvyšuje úspěšnost a využitelnost konečné podoby IS.

Samotné modelování IS na základě bussines procesů je realizováno různými diagramy, jako jsou např. business process diagramy, data-flow diagramy, vývojové diagramy, use-case diagramy apod.

5. Informační systém

Pod pojmem IS rozumíme strukturovaný souhrn informací, které sbíráme, zpracováváme a využíváme k práci nebo k rozhodování. Základem IS je zpracování dat, jádrem je tedy databázový systém. Toto jádro je pak obaleno dalšími komponentami a podpůrnými systémy, jako je ekonomický systém firmy, systémy řízení a systémy automatizovaného řízení výroby, poštovní server, grafické systémy, skladové hospodářství a podobně.

Kvalitní IS tvoří celek, jehož realizace je velmi náročná a vyžaduje spolupráci stanoveného vývojového týmu. Ten je tvořen vybranými zástupci zadavatele i tvůrce, jako jsou zejména analytici, manažeři, vývojáři, architekti, testeři a administrátoři. Vývojem IS se v dnešní době zabývají specializované firmy, které jsou schopny zpracovat i velké IS „na míru“ podle požadavků zákazníka.

5.1. Výklad pojmu systém

Pokud hovoříme o IS, musíme se zabývat také otázkou, co rozumíme pod pojmem ***systém***. Obecně lze říci, že systém je účelově uspořádaná množina prvků a vazeb mezi nimi s dynamickým a účelovým chováním.

Obecný systém definujeme pomocí:

- prvků,
- vazeb mezi nimi,
- parametrů, tj. ohodnocení vazeb a prvků,

- účelových funkcí, tj. důvodů existence systému,
- cílových funkcí, tj. stavu, kterého chceme dosáhnout.

Definice systému může být buď výsledkem, nebo vstupem analytické činnosti. Pokud systém existuje, pak ***analýzou systému*** rozumíme jednoznačnou činnost, kdy na základě známé struktury systému je třeba určit jeho chování. Naproti tomu ***syntézou systému*** rozumíme nejednoznačnou činnost, kdy na základě požadovaného chování hledáme odpovídající strukturu systému, která by toto chování zajistila.

Systémy lze klasifikovat do různých tříd. Např.:
- podle vztahu k realitě na reálné a abstraktní,
- podle způsobu vzniku na přirozené a umělé,
- podle chování v čase na statické a dynamické,
- podle typu systémových veličin na spojité a diskrétní,
- podle tvaru statické charakteristiky na lineární a nelineární.

5.2. Životní cyklus IS [3]

Životní cyklus IS je tvořen etapami, jejichž posloupnost musí být zachována. Pro úspěšnou tvorbu kvalitního IS nelze jednotlivé etapy vynechat nebo zpracovávat v jiném pořadí, protože výstup jedné etapy je zpravidla podkladem pro zpracování další etapy. Moderní návrhové metodiky již připouští, že se etapy vývoje mohou iterativně opakovat. Rozlišujeme tyto etapy:
- specifikace cílů = předběžná analýza,
- specifikace požadavků = analýza systému,
- návrh = projektová studie,
- implementace,
- testování,
- zavádění systému,
- zkušební provoz,
- rutinní provoz a údržba,
- reengineering.

5.3. Řízení rizik při vývoji IS

Jedno české úsloví praví *„jen připraveným štěstí přeje“*. A toto platí i při vývoji IS. Proto je v průběhu životního cyklu IS kladen velký důraz na analýzu rizikových faktorů, určení pravděpodobnosti jejich výskytu a předběžný návrh opatření k odstranění jednotlivých rizik. Rizika dělíme do následujících kategorií:
- politická rizika,
- technologická rizika,
- riziko nedostatku zdrojů,
- znalostní a dovednostní riziko,
- riziko nových požadavků.

KAPITOLA B.

Ve druhé části mé bakalářské práce jsem realizovala praktickou ukázku vývoje informačního systému. Jako vzor jsem použila systém ozdravných pobytů, které pořádá Všeobecná zdravotní pojišťovna ČR pro své dětské klienty. Zajištění realizace těchto pobytů bylo náplní mé práce.

1. Deklarace záměru

Zadavatel projektu, zdravotní pojišťovna, každoročně organizuje a realizuje ozdravné pobyty pro své dětské klienty. Zadání projektu se týká vytvoření informačního systému, který eviduje klienty a doprovodné pracovníky přihlášené k účasti na pobytech. V informačním systému jsou vedeny veškeré údaje o klientech a doprovodu, které jsou pořizovány nejprve ve fázi přihlášení k pobytu a dále jsou doplňovány v rámci zpracování, kdy jsou klienti a doprovod rozdělováni do turnusů a skupin podle stanovených kritérií.

2. Vize projektu

2.1. Definice pojmů

V dokumentaci jsou používány tyto pojmy:

Pojem	Popis
zákazník	zadavatel projektu.
expozitura	přepážkové pracoviště zákazníka.
referent	pracovník zákazníka, který má oprávnění k určeným operacím nad databází, tj. editace a čtení dat, zadání přihlášky klienta nebo doprovodu.
koordinátor	pracovník zákazníka, který má neomezená oprávnění k operacím nad databází.
lékař	pracovník zákazníka, který má oprávnění k určeným operacím nad databází, tj. má oprávnění referenta a dále schvaluje přihlášky klientů k pobytu.
pobyt	ozdravný pobyt, který zákazník organizuje a realizuje pro své dětské klienty. Každý ročník pobytů je realizován jako samostatný celek.
klient	přihlášený zájemce o účast na ozdravném pobytu, tj. dítě ve věku 6 – 16 let, pojištěnec zákazníka.
doprovod	pracovník ve věku od 21 let, který pečuje o klienty v průběhu pobytu. Jeden doprovodný pracovník má na starosti jednu skupinu.
databáze klientů	zákazník v souvislosti s pobyty vytváří a vede seznam klientů, který obsahuje základní osobní a korespondenční údaje klienta.
databáze doprovodu	zákazník vede databázi doprovodu, kde jsou uvedeny základní osobní a korespondenční údaje doprovodu.
turnus	jednotka pobytů. Každý turnus má číslo, termín, kapacitu a cenu, které jsou pro každý ročník pobytů pevně stanoveny.
skupina	skupina maximálně 10 klientů, kteří jsou přibližně stejného věku, a vede ji jeden doprovodný pracovník.

přihláška	formulář, kterým se klient přihlašuje k aktuálnímu ročníku pobytů. Obsahuje pouze náležitosti týkající se pobytu, tzn. turnus a nepovinnou položku poznámky.
záznam	záznam klienta nebo doprovodu v databázi systému.

2.2. Současný způsob vedení a zpracování databáze

Zákazník je organizátorem pobytů pro své klienty ve věku 6 – 16 let. Každý rok je veden jako samostatný celek, pro který je určen počet turnusů, jejich kapacita, termín, místo konání a cena.

Klienti se k účasti hlásí prostřednictvím papírové přihlášky, která musí být podána na některé z expozitur. Přihláška podléhá schválení lékařem. Lékař kontroluje, zda je klient pojištěncem zákazníka a zda odpovídají zdravotní indikace, event. nejsou shledány kontraindikace. K dalšímu zpracování postupují pouze schválené přihlášky. Tyto jsou referenty zadány do vstupní databáze. Neschválené přihlášky jsou ihned vráceny klientům.

Doprovod se k účasti hlásí prostřednictvím papírové přihlášky, která musí být podána na některé z expozitur. Přihláška je referenty zadána do vstupní databáze. Pokud zájemce o doprovod nesplňuje podmínky (např. dolní věkovou hranici), je mu přihláška ihned vrácena jako zamítnutá.

Přihlášky jsou k dispozici ve formátu PDF na internetu zákazníka nebo v tištěné podobě na expoziturách. Přihlášku není možno podat elektronicky. Přihlášky se přijímají v období určeném zákazníkem.

Při pořízení záznamu jsou do databáze vloženy osobní údaje a dále požadavky na preferované turnusy. V dohodnutý termín předají referenti vstupní databáze (klientů i doprovodu) ke zpracování koordinátorovi. Koordinátor sestaví ze vstupních databází jednu ústřední databázi klientů a jednu ústřední databázi doprovodu, které dále zpracovává. Tato část zpracování obnáší následující kroky:

- kontrola duplicit – každý klient nebo doprovod se v daném roce může zúčastnit pobytů pouze jedenkrát,
- kontrola úplnosti a správnosti údajů – při nedostatcích je vyžádáno doplnění od expozitury, kde byla přihláška podána,
- rozdělení klientů a doprovodu do turnusů – podle možností je přihlíženo k preferenci turnusů uvedených v přihlášce,
- ve spolupráci s příslušným smluvním partnerem (tiskárnou) zajištění obeslání klientů a doprovodu. Klienti obdrží návratku a avízo k úhradě pobytu. Doprovod obdrží pouze návratku. Koordinátor poskytuje tiskárně vzor návratek a data.
- klienti a doprovod, které nebylo možno zařadit z důvodu naplněné kapacity, jsou vedeni v databázích jako náhradníci. Náhradníci obdrží dopis, ve kterém jsou o tomto stavu vyrozuměni. Dopisy rovněž rozesílá smluvní partner (tiskárna).

V další části zpracování jsou do databáze postupně zaznamenávány údaje o vrácených návratkách. Tím je aktualizována skutečná obsazenost turnusů. Vrácením návratky klient nebo doprovod:

- akceptuje zařazení = pobytu se zúčastní,
- pobyt stornoval = pobytu se nezúčastní,

- žádá o přeřazení do jiného turnusu = zúčastní se jen v případě změny turnusu,
- nereagoval, což může mít různé důvody, a proto je nutno tento stav prověřit telefonicky nebo jiným dostupným kontaktem.

Další fáze zpracování je zaměřena na kompletaci turnusů. Tato část zpracování obnáší tyto činnosti:

- přiřazení doprovodu ke skupinám,
- rozdělení klientů do skupin,
- vyřizování požadavků na přeřazení klientů do jiného turnusu/skupiny,
- obsazování uvolněných míst náhradníky a jejich obeslání.

Příjem přihlášek trvá do naplnění kapacity pobytů a hlavní databáze je průběžně doplňovaná. Výše uvedené kroky se tak periodicky opakují vždy po obdržení nových dat.

2.3. Zhodnocení současného stavu

Z výpisu činností je patrné, že hlavní náplní agendy je sumarizace a zpracování dat. Výstupem jsou seznamy klientů a doprovodu tříděné podle zadaného klíče (sestavy pro mailing, turnus, skupiny apod.). V současné době je agenda vedena a zpracovávána v programu MS Excel a je oddělena administrativní část od systémové. Vstupní data jsou pořizována více pracovníky na různých pozicích a v různých místech. Zpracování provádí jeden pracovník. Předávání dat probíhá elektronicky formou e-mailových zpráv. Všechny části zpracování jsou dotčeny lidským faktorem. Data nejsou chráněna proti ztrátě nebo poškození.

Při hodnocení současného stavu z hlediska obchodního, personálního a technického byly zjištěny následující nedostatky a rizika:

Nedostatek/riziko	***Popis***
Záznam dat do programu MS Excel	není zajištěno dostatečné zabezpečení dat před ztrátou nebo poškozením.
Decentralizované zpracování dat	není zajištěna integrita dat, existuje reálná možnost vzniku duplicit.
Vysoký podíl ruční práce při zařazování klientů a doprovodu do turnusů a skupin	zpracování dat je prováděno pouze ručně, což sebou nese vysoké riziko chybovosti lidského faktoru.
Elektronický přenos souborů mezi expoziturami	riziko poškození nebo ztráty dat při nezabezpečeném přenosu.
Data nejsou uložena a zpracovávána na serveru, ale na lokálních PC	není jedno centrální úložiště, chybí pravidelné zálohování dat.
Opakované zadávaní dat při účasti klienta nebo doprovodu ve více ročnících	nadbytečný nárůst práce z důvodu zadávání stejných dat každý rok.
Nelze se k pobytu přihlásit elektronicky	vzniká nadbytek agendy referentů, která je, s ohledem na rozvoj internetu, dnes již nadbytečná.
Zpracování centrální databáze jedním koordinátorem	chybí zastupitelnost na pozici koordinátora.

Zákazník může realizaci pobytů ukončit	realizace pobytů je závislá na mnoha faktorech a zákazník se může kdykoli rozhodnout o ukončení realizace pobytů.

2.4. Specifikace cílů

Zákazník požaduje zpracování informačního systému, který umožní vytvoření a zpracování informačního systémů pobytů s minimalizací rizika poškození nebo ztráty dat.

Cílem projektu je vytvoření informačního systému ozdravných pobytů s jednou centrální databází klientů a jednou databází doprovodu. Přihlášení k pobytu bude realizováno zejména elektronicky, ručním zadáním přes referenta pouze ojediněle. Při opakovaném pobytu bude možno pouze editovat data dle potřeby, nebude nutné vytvoření nového záznamu. Schvalování lékařem bude probíhat elektronicky. Klienti a doprovod, přihlášení k danému ročníku pobytů, budou moci on-line sledovat stav své přihlášky, tj. schválení, zařazení apod. Potvrzení pobytu umožní systém klientovi nebo doprovodu elektronicky. Ze strany zákazníka budou nastaveny uživatelské role referent – lékař – koordinátor s různým stupněm oprávnění k operacím nad databází.

3. Katalog požadavků

3.1. Funkční požadavky

Aplikace:

- umožní vložení klienta nebo doprovodu do databáze přes elektronický formulář,
- bude kontrolovat úplnost vstupních údajů, neumožní vložení klienta nebo doprovodu bez povinných údajů,
- umožní vložení klienta nebo doprovodu bez zadání doplňkových informací,
- bude kontrolovat duplicity záznamů,
- umožní editaci dat,
- bude vyžadovat elektronické schválení přihlášky klienta lékařem,
- umožní vyřazení klienta nebo doprovodu z pobytů – např. v případě překročení horní věkové hranice, změny zdravotní pojišťovny apod.,
- přiřadí klienta nebo doprovod do turnusu,
- přiřadí klienta nebo doprovod do skupiny – podmínkou bude souhlas klienta nebo doprovodu se zařazením do turnusu a kritériem bude věk,
- umožní přidání nově získaných dat i v průběhu zpracování,
- bude poskytovat informace o obsazenosti turnusů,
- umožní přesun klienta nebo doprovodu do jiného turnusu nebo skupiny,
- nebude evidovat nezařazené klienty nebo doprovod v pozici náhradníka,
- nebude archivovat historii účasti.

3.2. Nefunkční požadavky

Aplikace:

- bude provozována na aplikačním serveru GlassFish 3.1.2.,
- bude zpracována v programovacím jazyce Java, verze 7.0,
- bude vyvíjena ve vývojovém prostředí NetBeans IDE 7.2.,

- bude pro komunikaci s klienty a doprovodem používat webová rozhraní přístupná standardními webovými prohlížeči,
- bude pracovat s maximálním počtem klientů 5 000, maximální počet doprovodu pak bude 1 000,
- bude uchovávat osobní údaje klientů, kteří se k pobytům přihlásili,
- nevyžaduje lokalizaci uživatelského rozhraní do cizích jazyků,
- nebude řešit archivaci dat. Toto bude řešeno mimo rámec aplikaci, a to na systémové úrovni.

4. Plán testování

4.1. Verifikace

V rámci verifikačního testování budou vytvořeny testy jednotlivých tříd s využitím frameworku JUnit.

4.2. Validace

Validační testování bude probíhat prostřednictvím nezávislých testerů a pracovníků zákazníka. V rámci tohoto testování bude prověřena funkčnost systému z hlediska jednotlivých uživatelských rolí. Pro jednotlivá testování budou sestaveny testovací scénáře tak, aby bylo zajištěno testování všech funkcí aplikace. Testy budou provedeny v tomto minimálním rozsahu:

- testování funkcí povolených pro roli referenta,
- testování funkcí povolených pro roli lékaře,
- testování funkcí povolených pro roli koordinátora,
- průchod aplikace nově přihlášeným klientem,
- průchod aplikace evidovaným klientem,
- přihlášení doprovodu.

4.3. Akceptační test

V rámci předání aplikace zákazníkovi bude proveden akceptační test, při kterém budou ověřeny tyto vybrané funkcionality:

- přihlášení nového klienta k pobytu,
- přihlášení již existujícího klienta k pobytu,
- průběh zpracování přihlášky, tj. schválení a zařazení klienta,
- vyřazení klienta z pobytu,
- kontrola uživatelského manuálu - ověření shody metodiky a reality u vybraných operací.

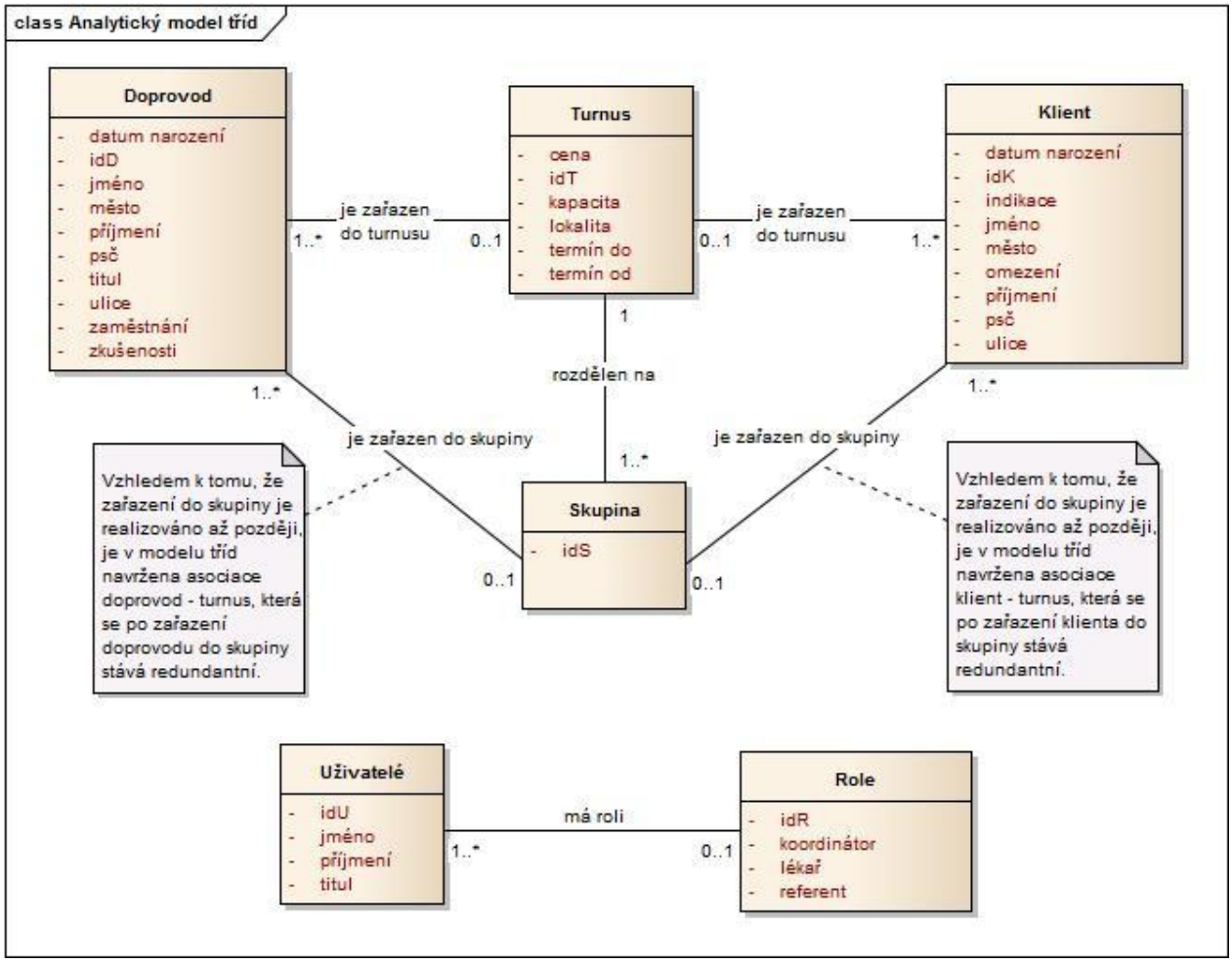
Informační systém bude akceptován, pokud výše uvedené testy skončí bez chyby.

5. Analytická dokumentace

5.1. Aktéři procesů

Aktér	Popis a činnosti
Klient	• pojištěnec zákazníka ve věku 6 – 16 let, který se přihlásí k pobytu a jehož záznam je schválen lékařem, tzn. splňuje podmínky pro pobyt stanovené zákazníkem, • pro přístup k záznamu je vyžadován ID kód, který si při prvním přihlášení volí klient sám a dále jej nelze měnit, • k pobytu se přihlásí elektronickou, event. papírovou přihláškou, • po přihlášení je zařazen do turnusu, • po schválení přihlášky je zařazen do skupiny, • může být přeřazen do jiného turnusu nebo skupiny, • může pobyt stornovat, • může být z pobytu vyřazen (např. zdravotní důvody), osobní údaje v databázi ale zůstávají k dispozici pro další ročníky pobytů.
Doprovod	• zájemce o funkci doprovodu starší 21 let, který se přihlásí k pobytu, • k pobytu se přihlásí papírovou přihláškou, • po schválení je doprovod zařazen do databáze a je zařazen do turnusu a skupiny, • může být přeřazen do jiného turnusu nebo skupiny, • může pobyt stornovat, • může být z pobytu vyřazen (např. zdravotní důvody), osobní údaje v databázi ale zůstávají k dispozici pro další ročníky pobytů.
Referent	• zaměstnanec zákazníka pověřený zpracováním agendy pobytů, • při práci s aplikací je vyžadován login a username, • může zadat záznam do systému, • může editovat data, • může prohlížet celou databázi klientů a doprovodu.
Lékař	• zaměstnanec zákazníka pověřený zpracováním agendy pobytů, • při práci s aplikací je vyžadován login a username, • může vykonávat činnosti referenta, • schvaluje/zamítá záznam klienta, tzn. kontroluje, zda klient - je pojištěncem zákazníka, - splňuje zdravotní indikace, - nemá omezení k pobytu, • může vyřadit klienta z pobytu, tzn. neschválí přihlášku.
Koordinátor	• zaměstnanec zákazníka pověřený zpracováním agendy pobytů, • při práci s aplikací není vyžadován, • má uživatelský a administrátorský přístup k funkcím aplikace, • zadává nové uživatele a přiděluje uživatelské role, • může vykonávat činnosti referenta a lékaře, • schvaluje přihlášku doprovodu, • provádí změnu zařazení klienta/doprovodu, • může vyřadit klienta nebo doprovod z pobytu.

5.2. Analytický model tříd



Obr. 01 Analytický model tříd.

V sekci atributů nejsou uvedeny atributy reprezentující asociace mezi entitními třídami.

5.2.1. Klient:

Klient je pojištěnec zákazníka ve věku 6 – 16 let, který se přihlásil k pobytu.

Atribut	***Popis***
idK	Unikátní identifikační číslo klienta.
jméno	Křestní jméno klienta.
příjmení	Příjmení klienta.
datum narození	Datum narození klienta.
ulice	Bydliště klienta – ulice + číslo domu.
město	Bydliště klienta – město.
psč	Bydliště klienta – PSČ.
indikace	Zdravotní indikace k pobytu.
omezení	Zdravotní omezení k pobytu.

5.2.2. Doprovod:

Doprovod jsou osoby ve věku od 21 let, které pečují o klienty v průběhu pobytu. Jeden doprovodný pracovník má na starosti vždy jednu skupinu klientů.

Atribut	*Popis*
idD	Unikátní identifikační číslo osoby.
jméno	Křestní jméno osoby.
příjmení	Příjmení osoby.
titul	Titul osoby.
datum narození	Datum narození osoby.
ulice	Bydliště osoby – ulice + číslo domu.
město	Bydliště osoby – město.
psč	Bydliště osoby – PSČ.
zaměstnání	Zaměstnání osoby.
zkušenosti	Zkušenosti s vedením dětí.

5.2.3. Turnus:

Entita specifikuje turnusy, do kterých je pobyt rozdělen.

Atribut	*Popis*
idT	Unikátní identifikační číslo turnusu.
termín od	Začátek turnusu.
termín do	Konec turnusu.
lokalita	Místo konání.
cena	Cena pobytu za jednoho klienta.
kapacita	Kapacita turnusu = maximální počet klientů pro jeden turnus.

5.2.4. Skupina:

Skupina je jednotka turnusu. Každá skupina má maximálně 10 klientů stejného věku.

Atribut	*Popis*
idS	Unikátní identifikační číslo skupiny.

5.2.5. Uživatel:

Uživatel je zaměstnanec zákazníka, který má přístup do informačního systému a je oprávněn vykonávat operace nad databází v rozsahu přidělené role.

Atribut	*Popis*
idU	Unikátní identifikační číslo uživatele.
jméno	Křestní jméno uživatele.
příjmení	Příjmení uživatele.
titul	Titul uživatele.

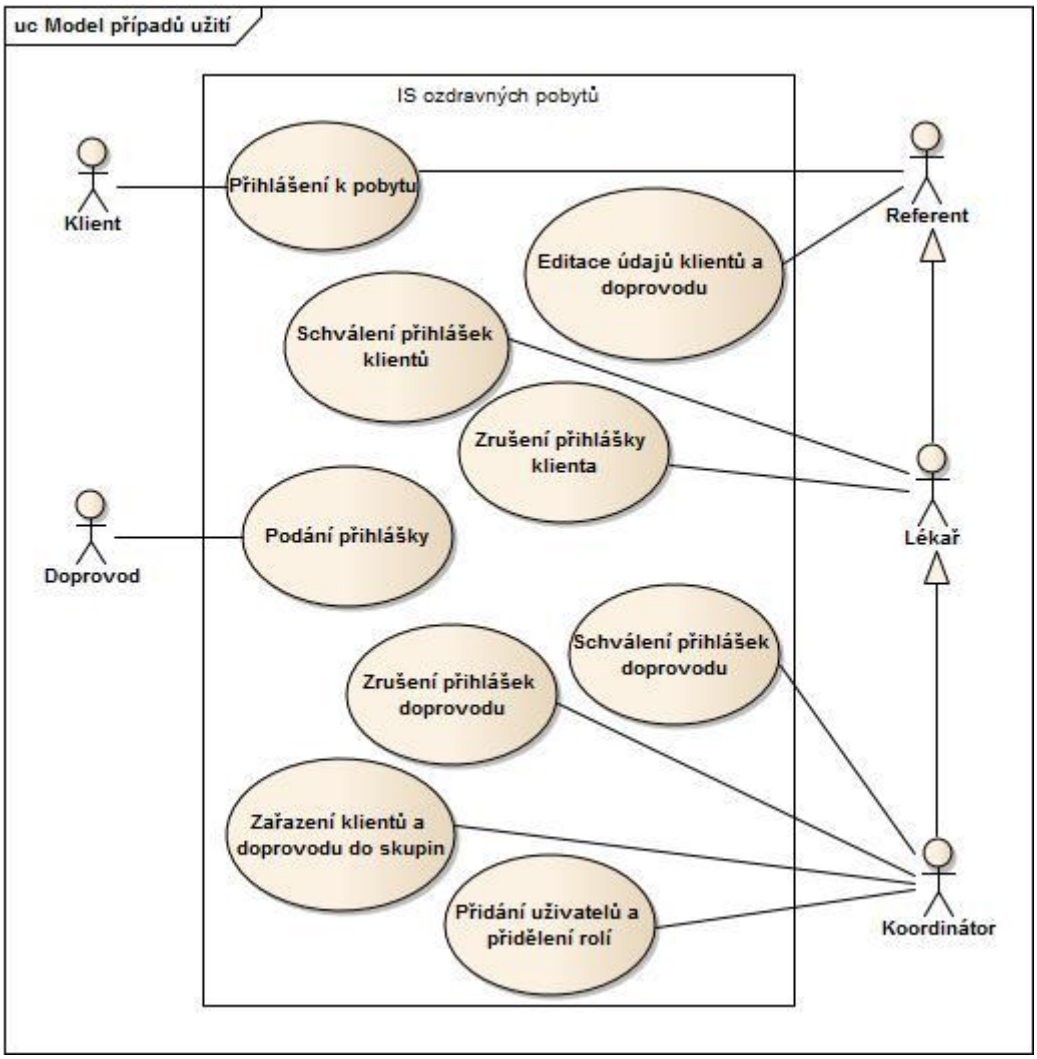
5.2.6. Role:

Entita specifikuje role pro uživatele.

Atribut	*Popis*
idR	Unikátní identifikační číslo role.

koordinátor	Role s nejvyšším uživatelským oprávněním.
lékař	Role pro lékaře, kteří schvalují přihlášky.
referent	Role s nejnižším uživatelským oprávněním.

5.3. Model případů užití



Obr. 02 Model případů užití.

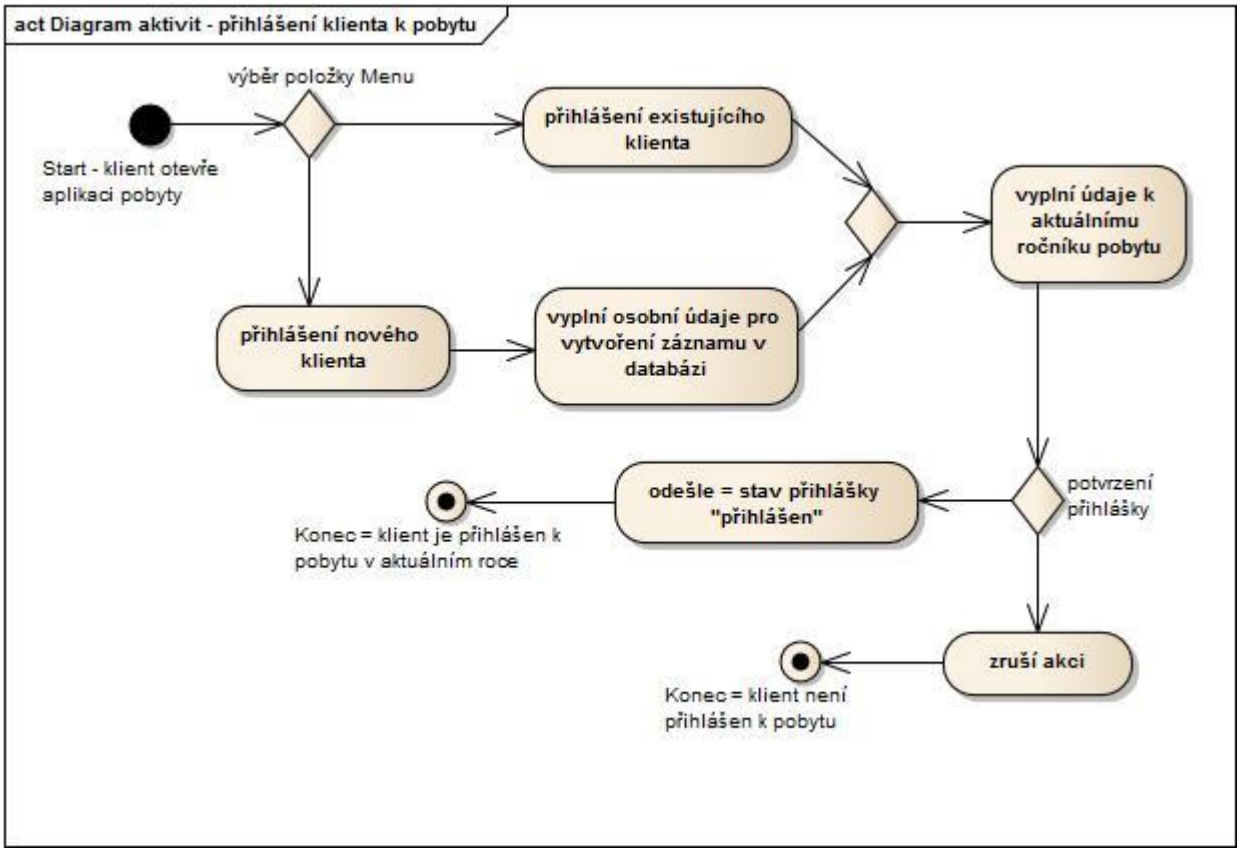
5.3.1. *Případ užití:* Vyplnění přihlášky

Aktér: Klient, Referent

Hlavní průchod případu užití:

a) Klient je veden v systému, tzn. pobytů se v minulosti již účastnil:
- otevře formulář pro vyhledání svého záznamu,
- vyhledá svůj záznam přes ID kód a příjmení,
- v případě potřeby edituje osobní údaje,
- doplní informace k aktuálnímu ročníku pobytů (volba turnusu),
- odešle záznam,
- obdrží informaci o ID kódu, který se nemění,
- stav přihlášky je ve stavu "přihlášen" a klient již nemůže údaje měnit.

b) Klient není v systému, tzn. pobytů se účastní poprvé:
- otevře elektronický formulář přihlášky,
- vyplní povinné, event. i nepovinné údaje,
- odešle záznam do systému,
- obdrží potvrzení o ID kódu,
- stav přihlášky je ve stavu "přihlášen" a klient již nemůže údaje měnit.



Obr. 03 Diagram aktivity – přihlášení klienta k pobytu.

5.3.2. *Případ užití:* Vyplnění přihlášky

Aktér: Doprovod

Hlavní průchod případu užití:

Doprovod
- z aplikace pobyty si vytiskne formulář přihlášky,
- po vyplnění jej doručí na některou z expozitur zákazníka,
- koordinátor přihlášku posoudí a schválí/neschválí přihlášku,
- po rozhodnutí o zařazení doprovodu je přihláška zadána do systému.

5.3.3. *Případ užití:* Editace údajů klienta a doprovodu

Aktér: Referent

Hlavní průchod případu užití:

Referent
- vyhledá klienta nebo doprovod v databázi podle ID nebo osobních údajů,
- provede editaci dat,
- editaci osobních údajů může provádět bez omezení podle potřeby.

5.3.4. *Případ užití:* Schválení přihlášek klientů

Aktér: Lékař

Hlavní průchod případu užití:

Lékař

- neobdrží avízo o novém záznamu, mezi povinnosti role "lékař" patří periodicky kontrolovat stav neschválených přihlášek,
- otevře aplikaci,
- vybere záznamy ke schválení,
- záznamy je nutno posoudit a schválit jednotlivě,
- v případě potřeby provede aktualizaci dat,
- provede schválení,
- změny uloží.

5.3.5. *Případ užití:* Zrušení přihlášky klienta

Aktér: Lékař

Hlavní průchod případu užití:

Lékař

- vyhledá v databázi přihlášek přihlášku, kterou chce zamítnout (tj. stav *„zrušen“*),
- vybere edit přihlášky,
- přihlášku nastaví do stavu *„zrušen“*.

5.3.6. *Případ užití:* Schválení přihlášky doprovodu

Aktér: Koordinátor

Hlavní průchod případu užití:

Koordinátor

- přihlášky doprovodu obdrží v papírové podobě,
- posoudí přihlášku a rozhodne o schválení/zamítnutí,
- schválené přihlášky jsou zavedeny do systému.

5.3.7. *Případ užití:* Zrušení přihlášek doprovodu

Aktér: Koordinátor

Hlavní průchod případu užití:

Koordinátor

- vyhledá přihlášku v databázi doprovodu,
- záznam smaže.

5.3.8. *Případ užití:* Zařazení klientů a doprovodu do skupin

Aktér: Koordinátor

Hlavní průchod případu užití:

Koordinátor

- vybere v menu položku k zařazení klientů, která poskytne seznam přihlášek ve stavu *„schválen“*,
- klienty postupně zařadí do skupiny,
- pokud skupina neexistuje, operativně ji vytvoří,
- doprovod je do skupiny zařazen při založení skupiny.

5.3.9. *Případ užití:* Přidání uživatelů a přidělení rolí

Aktér: Koordinátor

Hlavní průchod případu užití:

Koordinátor

- zadá nového referenta nebo lékaře do databáze uživatelů,
- přidělí uživateli uživatelské jméno (nelze nadále editovat),
- přidělí uživateli počáteční heslo (lze editovat),
- přidělí uživateli roli.

5.4. Analýza požadavků

5.4.1. Systémové zabezpečení

a) Vzhledem k tomu, že aplikace spravuje citlivá data, je přenos informace mezi klientem a serverem zabezpečen šifrováním (http protokol).

b) Přístup k datovým zdrojům je chráněn uživatelským username a heslem a rozsah přístupu je řešen uživatelskými rolemi referent, lékař, koordinátor.

5.4.2. Zabezpečení přístupu do systému pro klienty

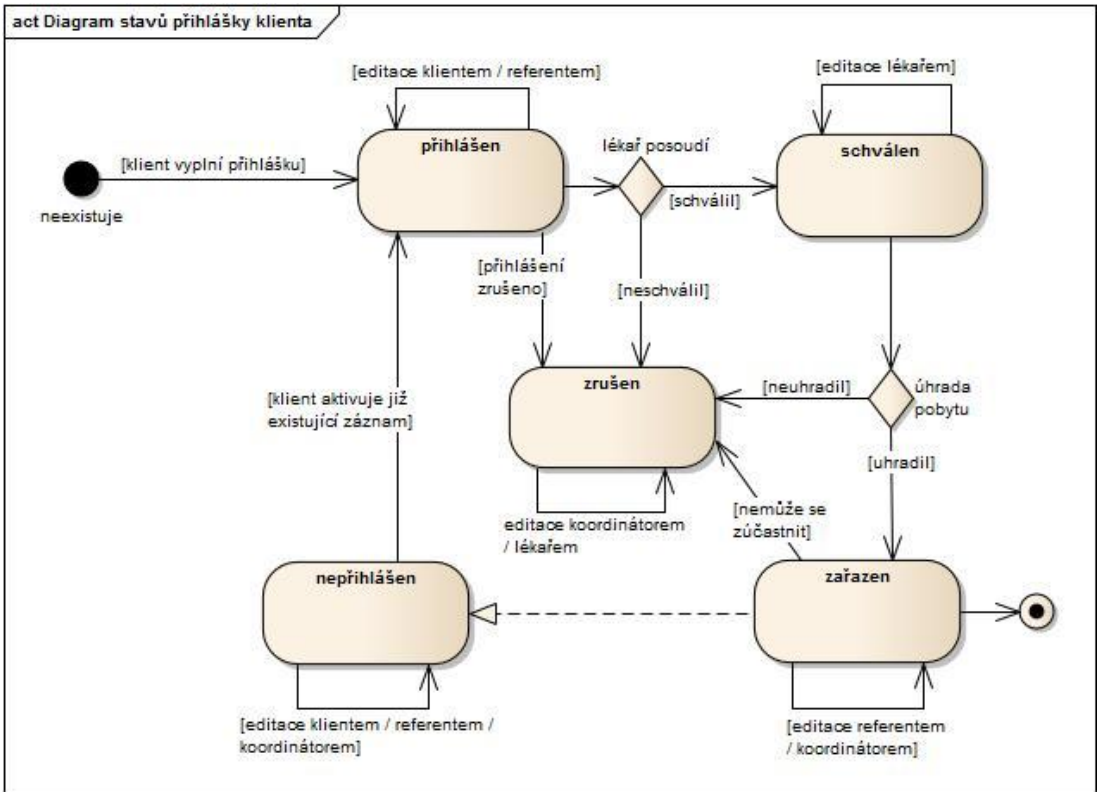
Oproti standardní situaci vyžadovat "username" a "heslo" byl zvolen přístup přes "ID záznamu" a "příjmení", a to z těchto důvodů:

- *ID kód* nahrazuje heslo*, příjmení* nahrazuje username,
- ID kód je řetěz minimálně šesti znaků,
- ID si volí klient sám při prvním přihlášení do databáze. Nadále již není ID kód možno editovat,
- k ID kódu je nutno uvést odpovídající příjmení klienta – tak je zajištěno, že se klient nedostane do záznamu jiného klienta,
- pod ID kódem lze svůj záznam v databázi kdykoli vyvolat a editovat data, přihlásit se k aktuálnímu ročníku pobytů nebo po přihlášení sledovat stav přihlášky,
- ID je použito zároveň jako identifikátor platby, tzn. jako variabilní symbol.

5.4.2. Diagram stavů přihlášky klienta

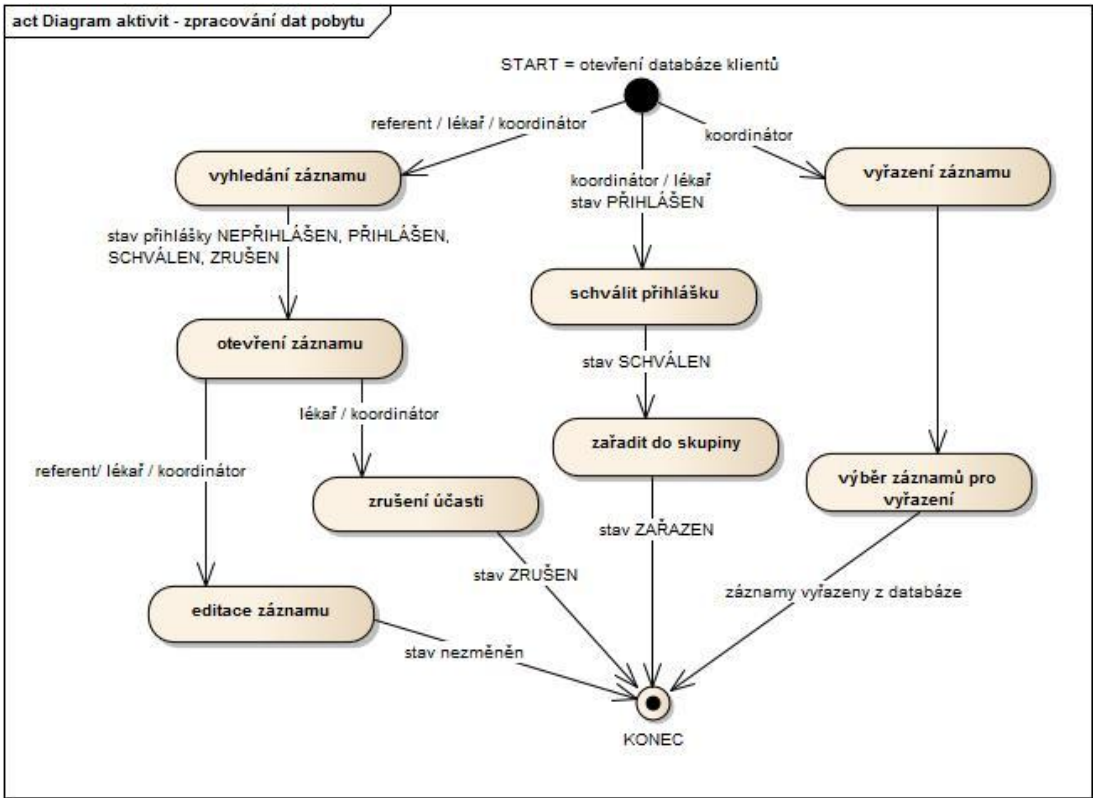
Přihláška klienta se může nacházet v následujících stavech:

- **NEPŘIHLÁŠEN** = stav všech záznamů v databázi před zahájením aktuálního ročníku pobytů; pokud klient není v databázi, pak jeho záznam zatím neexistuje,
- **PŘIHLÁŠEN** = klient se přihlásil k pobytům,
- **SCHVÁLEN** = přihlášku schválil lékař,
- **ZAŘAZEN** = pobyt byl klientem uhrazen a klient je zařazen do skupiny. Po ukončení daného ročníku pobytů koordinátor nastaví záznamy v tomto stavu do stavu *nepřihlášen*,
- **ZRUŠEN** = přihláška může být zrušena, tzn. klient se nemůže pobytu zúčastnit, z důvodu neschválení přihlášky, neuhrazení pobytu, na základě vlastního rozhodnutí nebo z jiných důvodů, které brání účasti.



Obr. 04 Diagram stavů přihlášky klienta.

5.4.3. Diagram aktivity - zpracování dat pobytu



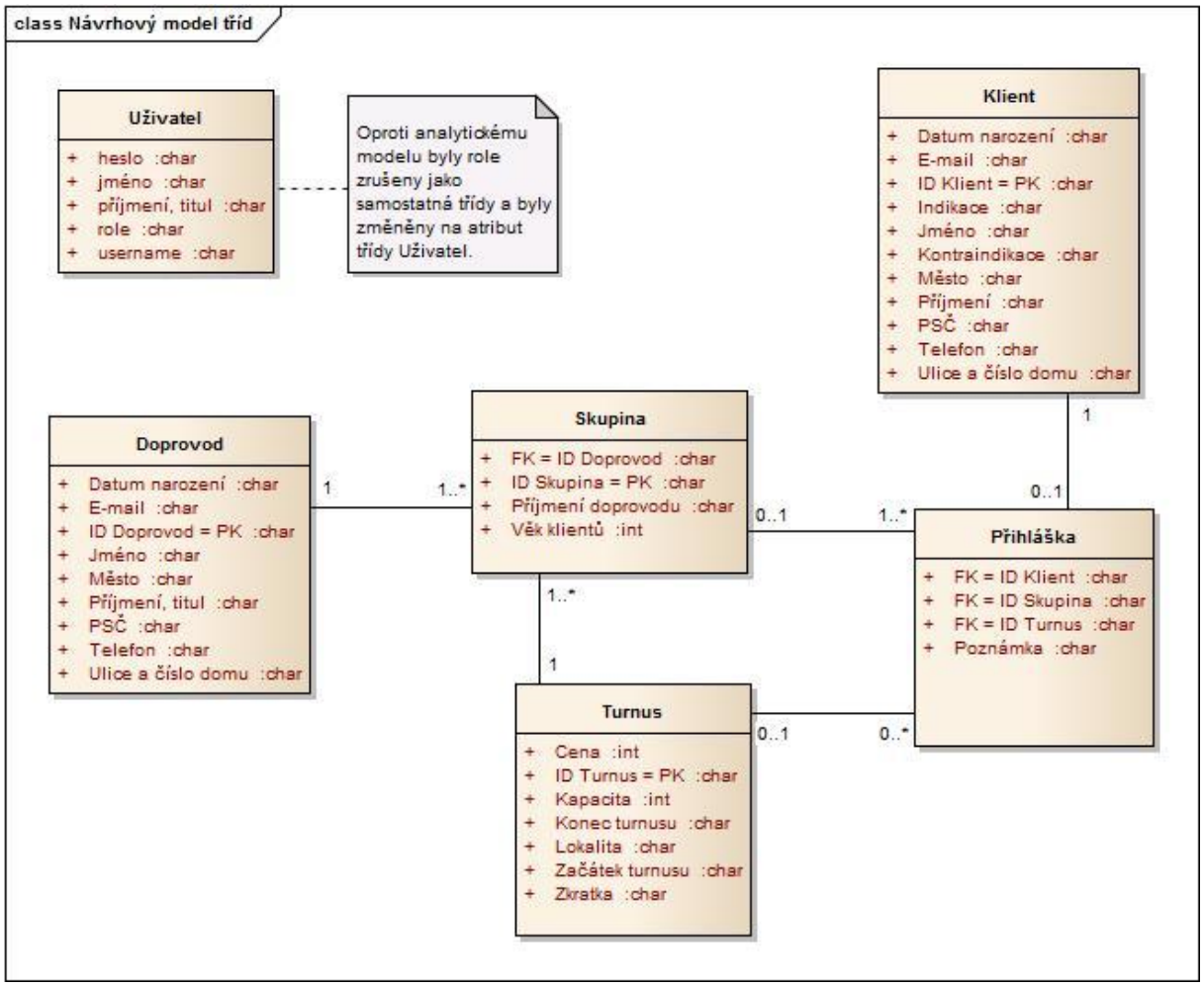
Obr. 05 Diagram aktivity – zpracování dat pobytu.

6. Návrhový model tříd

V této části vývoje informačního systému byl revidován datový model tříd. Informace o klientovi byly rozděleny do dvou částí

- základní osobní a korespondenční údaje, které chce zákazník uchovávat,
- informace týkající se pouze konkrétního ročníku pobytů, které zákazník archivovat nechce.

Z tohoto důvodu byla přidána nová třída Přihláška.



Obr. 06 Návrhový model tříd.

V sekci atributů je uvedeno u atributů reprezentující asociace mezi entitními třídami, zda jde o atribut reprezentující primární klíč (PK). Primární klíče jsou pak použity jako cizí klíče (FK) do jiných tabulek.

6.1. Klient

Klient je pojištěnec zákazníka ve věku 6 – 16 let, který má záznam v základní databázi. Záznam obsahuje základní osobní údaje. Tyto údaje zákazník v databázi archivuje. Kromě ID lze všechny údaje editovat.

Atribut	*Popis*
ID Klient	Primární klíč entity.
Jméno	Křestní jméno klienta.
Příjmení	Příjmení klienta.
Datum narození	Datum narození klienta. Je zadáváno ve formátu dd.mm.rrrr.
Ulice	Bydliště klienta – ulice + číslo domu.
Město	Bydliště klienta – město.
PSČ	Bydliště klienta – PSČ.
E-mail	E-mail klienta.
Telefon	Telefon klienta.
Indikace	Zdravotní indikace k pobytu.
Kontraindikace	Zdravotní omezení k pobytu.

6.2. Přihláška

Je formulář, kterým se klient přihlašuje k pobytu v aktuálním roce. Tyto údaje jsou platné jen pro daný ročník pobytů a nejsou archivovány.

Atribut	*Popis*
ID Klient	Cizí klíč z entity Klient.
ID Turnus	Cizí klíč z entity Turnus.
ID Skupina	Cizí klíč z entity Skupina.
Poznámka	Nepovinný údaj, doplnění informací.

6.3. Turnus

Entita specifikuje turnusy, do kterých je pobyt rozdělen.

Atribut	*Popis*
ID Turnus	Primární klíč entity.
Zkratka	Zkratka turnusu.
Termín od	Začátek turnusu.
Termín do	Konec turnusu.
Lokalita	Místo konání.
Cena	Cena pobytu za jednoho klienta.
Kapacita	Kapacita turnusu = maximální počet klientů pro jeden turnus.

6.4. Skupina

Skupina je jednotka turnusu. Každá skupina má maximálně 10 klientů stejného věku a jeden doprovod.

Atribut	*Popis*
ID Skupina	Primární klíč entity.
ID Doprovod	Cizí klíč z entity Doprovod.
Příjmení doprovodu	Příjmení doprovodu.
Věk klientů	Omezení pro zařazení klientů do jedné skupiny.

6.5. Doprovod

Doprovod jsou osoby ve věku od 21 let, které pečují o klienty v průběhu pobytu. Jeden doprovodný pracovník má na starosti vždy jednu skupinu klientů.

Atribut	*Popis*
ID Doprovod	Primární klíč entity.
Jméno	Křestní jméno doprovodu.
Příjmení, titul	Příjmení a titul doprovodu.
Datum narození	Datum narození doprovodu. Je zadáváno ve formátu dd.mm.rrrr.
Ulice	Bydliště osoby – ulice + číslo domu.
Město	Bydliště osoby – město.
PSČ	Bydliště osoby – PSČ.
E-mail	E-mail doprovodu.
Telefon	Telefon doprovodu.

6.6. Uživatel

Uživatel je zaměstnanec zákazníka, který má přístup do informačního systému a je oprávněn vykonávat operace nad databází v rozsahu přidělené role.

Atribut	*Popis*
Username	Uživatelské jméno
Příjmení, titul	Příjmení a titul uživatele.
Jméno	Křestní jméno uživatele.
Heslo	Heslo pro první přihlášení, je možné jej změnit.
Role	Lékař nebo referent. Koordinátor má administrátorský přístup.

7. Implementace

Kompletní kód aplikace je vygenerován na CD a tvoří Přílohu č. 1 této bakalářské práce. Aplikace byla programována v jazyku Java.

7.1. Technologie

Aplikace je vytvořena jako webová aplikace v technologii Java EE pro aplikační server GlassFish. To znamená, že se pro provádění nahrává na aplikační server ve standardním formátu war. Nicméně vlastní technologie Java EE využívá z aplikace přímo pouze servletovou specifikaci prostřednictvím použitého frameworku Simpleton a SQL server prostřednictvím knihovny JDBC. Aplikace tak může být jednoduše doplněna vestavěným webovým kontejnerem (např. Jetty) a vestavěnou databází (např. Embedded Derby) a provozována jako samostatná (standalone).

7.2. Simpleton

Framework Simpleton je jednoduchý javovský framework pro vytváření webových UI. Na rozdíl od standardních frameworků jako je např. JSF či Struts, které jsou primárně určeny pro webové designéry, je Simpleton zamýšlen především pro javovské vývojáře. Proto definuje

jenom minimální počet webových elementů a většina funkcionality je prováděna doprovodným javovským kódem. Na rozdíl od jiných frameworků nepoužívá Simpleton xml konfigurační soubory, anotace a skriptovací jazyk. Značkový jazyk Simpletonu je standardní XHTML rozšířený o jmenný prostor **www.smp.org**, ve kterém je definováno pouze několik nezbytně nutných elementů.
Každá xhtml stránka s dynamickým obsahem musí být doprovázena deklarací stejnojmenné třídy (tzv. page bean třídy). Tato třída, která musí být podtřídou **org.smp.Page**, musí obsahovat vlastnosti (properties), jejichž hodnoty určují dynamický obsah generované stránky. Hodnoty vlastností jsou ve stránce identifikovány svými jmény, umístěnými do závorek **${ … }** (resp **%{ …}**). Před tím, než se začne výsledná stránka generovat, vytvoří Simpleton instanci page bean třídy (tzv. page bean) a použitá vlastnost je ve stránce nahrazena jejich aktuální hodnotou z page bean.

Přílklad xhtml stránky a její odpovídající page bean třídy, která zobrazuje aktuální datum:

Stránka **HelloWorld.xhtml**:

```
<html xmlns="http://www.w3.org/1999/xhtml">
  <body>
    <div style="${style}"> Hello world at: ${date} </div>
  </body>
</html>
```

Odpovídající page bean třída:

```
public class HelloWorld extends Page {
  public Date getDate() {
    return new Date());
  }
  public String getStyle() {
    return "color:red";
  }
}
```

Deklarace getru v příslušné page bean není jediná možnost jak definovat hodnotu vlastnosti. Page bean je jenom jedna z bean, které se mohou účastnit generování stránky. Kromě page bean třídy, mohou být vlastnosti definovány i v dalších pomocných beanách, které mohou být libovolný javovský objekt. Všechny beany, které se účastní generování stránky, jsou uloženy v zásobníku, na jehož dně je umístěna page bean. Při hledání hodnoty vlastnosti se zásobník prohledává od vrcholu, takže vlastnosti se mohou zastiňovat. Zásobník bean je aplikaci přístupný operacemi **Page.pushbean** a **Page.popBean.** Tyto metody je možno volat v metodě **Page.createContent**, kterou Simpleton volá před vlastním generováním stránky.

Hodnoty vlastností nemusí být definovány pouze pomocí getrů. Pokud je pomocná bean deklarována jako mapa (implementuje rozhraní Map), potom jsou hodnoty uložené do této mapy chápány také jako hodnoty vlastností. Pro page bean je navíc možno hodnoty vlastnosti definovat voláním metody **Page.putItem** v metodě **Page.createContent**. Variantní forma page bean třídy pro stránku **HelloWorld.xhtml** může např. vypadat následovně:

```
public class HelloWorld extends Page {
  public void createContent()
    Map<String, String> styleMap = new HashMap<>();
    styleMap.put("style", "color.red");
    putItem("date", new Date());
    pushBean(styleMap);
  }
}
```

Pokud jsou pro odkaz na vlastnost v XHTML stránce použity závorky **${ … }** musí vlastnost povinně existovat a její hodnota nesmí být null, zatímco pro závorky **%{ … }** je existence vlastnosti nepovinná. Ve jmenném prostoru **www.smp.org** jsou definovány následující standardní elementy:

- **<smp:if condition="...">**, **<smp:else condition="...">** - podmíněné generování
- **<smp:loop iterable="...">** - iterace
- **<smp:option value="..." selected="...">** - generování elementů
- **<smp:include>** - vkládání do šablony

Podrobný popis frameworku Simpleton je uveden v příloze č. 3.

7.3. Architektura aplikace

Pro implementace aplikace byla zvolena standardní n-vrstvá architektura. Prezentační vrstva aplikace je umístěna v balících **akce**, které tvoří kontrolér, a v balíku **stranky**, který obsahuje třídy implementující webové rozhraní. Balík **business** obsahuje fasádu aplikace s business operacemi. Fasáda je realizována jako singleton. Balík **integration** obsahuje integrační část aplikace pro databázi Apache Derby. Ta je tvořena jednak DAO (Data Access Object) třídami obsahujícími CRUD operace nad jednotlivými databázovými tabulkami, a dále továrnou pro vytváření příslušných objektů DAO. Prezentační vrstva je realizována balíky **stranky** a **akce**, jejichž organizace je dána použitým webovým frameworkem Simpleton.

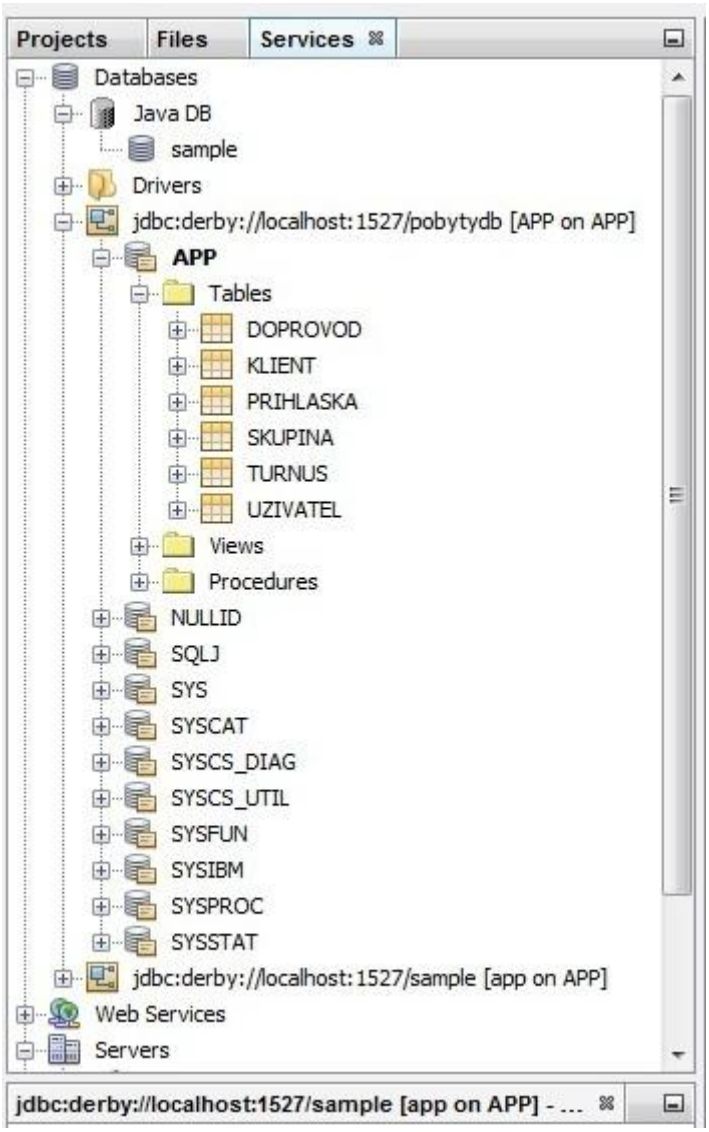
7.4. Vrstva datových zdrojů

Tato vrstva je tvořena databází Apache Derby. Výhodou této databáze je, že je implicitně součástí aplikačního serveru GlassFish, takže při nasazení aplikace na tento aplikační server není potřeba žádná další instalace databáze. Schéma databáze je tvořeno šesticí tabulek: **KLIENT**, **PRIHLASKA**, **TURNUS**, **SKUPINA**, **DOPROVOD** a **UZIVATEL**.

Ve schématu databáze jsou zavedena následující integritní omezení:

- primární klíče (s výjimkou **KLIENTID**) jsou definovány jako jednoznačné (**IDENTITY**) a jsou generovány automaticky,
- všechny atributy s výjimkou **DOPROVODID** v tabulce **SKUPINA** a **SKUPINAID** v tabulce **PRIHLASKA** jsou **NOT NULL**,
- atributy realizující asociace jsou definovány jako cizí klíče (**FOREIGN KEY**).

Databázové tabulky jsou vytvářeny automaticky při inicializaci integrační vrstvy.



Obr. 07 Náhled na uspořádání databáze.

7.5. Integrační vrstva

Integrační vrstva je vytvořena tak, aby umožňovala snadnou portaci aplikace pro jinou databázi než Apache Derby. Je tvořena balíkem **integration**, který obsahuje třídy podle návrhového vzoru DAO (Data Access Object). Jedná se o třídy **KlientDAODerby**, **PrihlaskaDAODerby**, **TurnusDAODerby**, **SkupinaDAODerby**, **DoprovodDAODerby** a **UzivatelDAO**. Každá z těchto tříd se stará o přístup do jedné z databázových tabulek: **KLIENT**, **PRIHLASKA**, **TURNUS**, **SKUPINA**, **DOPROVOD** a **UZIVATEL**.

Třídy DAO používají knihovnu JDBC a jazyk SQL pro realizaci datových operací (CRUD) nad jednotlivými databázovými tabulkami. Každá DAO třída obsahuje deklaraci objektů **java.sql.PreparedStatement**, které definují generické SQL příkazy **INSERT**, **SELECT**, **UPDATE** a **DELETE**.

Před vlastním vykonáním objektu **PreparedStatement** se objekt inicializuje konkrétními hodnotami parametrů. Jednotlivé SQL příkazy jsou vykonávány v autotransakčním módu.

Výsledek SQL příkazu **SELECT** je transformován na kolekci objektů odpovídajících entitních tříd.

Tovární třída **DerbyDAOFactory** vytváří jednak jednotlivé objekty DAO tříd a dále provádí počáteční vytvoření tabulek v databázi SQL příkazem **CREATE TABLE**. Toto řešení zajišťuje jednoduchou instalaci aplikace, při kterém není zapotřebí provádět žádný speciální inicializační skript.

7.6. Business vrstva

Business vrstva je tvořena balíkem **business**, který obsahuje návrhový vzor fasáda realizovaný abstraktní třídou Facade, ve které jsou deklarovány business operace, a její implementaci **FacadeImpl**. Oddělení deklarací a implementací business operací zajišťuje flexibilitu aplikace např. při změně technologie ukládání dat. Třída **FacadeImpl** implementuje jednotlivé business operace obvykle delegací na DAO objekty, které jsou vytvořeny továrnou **DerbyDAOFactory**.

7.7. Prezentační vrstva

Prezentační vrstva je obsažena v balících **stranky** a **akce**. Struktura balíku **stranky** je pevně daná použitým webovým frameworkem Simpleton, který požaduje, aby ke každé XHTML stránce existovala stejnojmenná třída rozšiřující knihovní třídu **org.smp.pages.Page** a obsahující definici dynamických vlastností stránek ve formě vlastností příslušné (viz Příloha č. 3). Vzhledem k tomu, že některé dynamické vlastnosti se v XHTML stránkách vyskytují opakovaně, jsou všechny třídy stránek v balíku **stranky** odvozeny od společné nadtřídy **AbstrPobytyStranka**, ve které jsou definovány společné vlastnosti, jako např. délky vstupních položek atp.:

```
public class AbstrPobytyStranka extends Page {
  public int getDatumSize() {
    return Datum.DATE_SIZE;
  }
  public int getJmenoSize() {
    return Jmeno.JMENO_SIZE;
  }
  . . .
}
```

Jednotlivé pagebean třídy, které odpovídají příslušným XHTML stránkám, pak obvykle definují jejich dynamický obsah na základě parametrů requestu v metodě **createContent**:

```
public class ViewKlient extends AbstrReferentStranka {
  @Override
  public void createContent() throws SmpException {
    KlientId id = new KlientId(getParameter("klientId"));
    Klient klient = Facade.getInstance().findKlient(id);
    pushBean(klient);
  }
}
```

Balík akce obsahuje třídy, které vykonávají akce požadované submitem formulářů v XHTML stránkách. Typická akce spočívá ve validaci a konverzi parametrů requestu a dále:

- provedení požadované business operace a redirektu na další stránku, pokud validace uspěla,
- nebo zobrazení stejné stránky s repopulovanými parametry a chybovými hláškami, pokud validace selhala.

Jako příklad uveďme akci přidání turnusu:

```
public class PridatTurnusAkce extends AbstractPobytyAkce {
  @Override
  public void execute() throws ServletException,
                                        IOException, SmpException {
  Map<String, Object> errorsMap = new HashMap<>();
  Cena cena = validateItem(Cena.class, errorsMap);
  Lokalita lokalita = validateItem(Lokalita.class, errorsMap);
  Kapacita kapacita = validateItem(Kapacita.class, errorsMap);
  Zkratka zkratka = validateItem(Zkratka.class, errorsMap);
  if (errorsMap.isEmpty()) {
    Facade.getInstance().addTurnus(zacatek, konec, lokalita,
                                          cena, kapacita, zkratka);
    redirect("/referentcast/turnus/TabTurnus.xhtml");
  } else {
    Page addTurnusBean =
         ceatePageBean("/referentcast/turnus/AddTurnus.xhtml");
    requestParamsToMap(errorsMap);      // repopulace parametrů
    addTurnusBean.pushBean(errorsMap);
    forward(addTurnusBean);
    }
  }
}
```

Pro validaci se využívá podpora frameworku Simpleton, který obsahuje generickou funkci **Page.validateItem**, která vyhledá pomocí introspekce a zavolá vlastní validační a konverzní metodu **validate**, která musí být nadeklarována ve třídě validovaného parametru. Tak např. metoda **validateItem** použitá v předcházející ukázce s parametrem **Lokalita.class** zavolá vlastní validační a konverzní metodu třídy **Lokalita**:

```
public static Lokalita validate(String lokalita)
                                        throws ValidationException {
  if (lokalita == null)
    throw new ValidationException("Prosím vyplňte lokalitu.");
  lokalita = lokalita.trim();
  if (lokalita.isEmpty())
    throw new ValidationException("Prosím vyplňte lokalitu.");
```

```
  if (lokalita.length() > LOKALITA_SIZE)
    throw new ValidationException(String.format(
    "Lokalita může být dlouhá maximálně %d znaků.",
                                                LOKALITA_SIZE));
  return new Lokalita(lokalita);
}
```

7.8. Konfigurace aplikace

Parametry aplikace jsou implementovány jako soubor vlastností **Parametry.properties** v balíku **pobyty**. Soubor obsahuje entry s následujícími klíči:

```
JNDIName =jdbc/pobytydb
koordinatorHeslo=892c70059ae7641e3e9b992abd2916f0
#ddp
firma=Název firmy
ulice=Ulice a číslo
mesto=PSČ Město
klientTelefon=+420 800 123 123
klientMobil=+420 777 123 123
klientFax=+420 123 456 789
klientMejl=info@pobyty.cz
koordinatorTel=+420 111 222 333
koordinatorMobil=+420 777 222 333
koordinatorMejl=koordinator@pobyty.cz
posudkarTel=+420 111 222 444
posudkarMobil=+420 777 222 444
posudkarMejl=posudkar@pobyty.cz
referentTel=+420 111 222 555
referentMobil=+420 777 222 555
referentMejl=referent@pobyty.cz
```

Hodnotou parametru koordinatorHeslo musí být 32 znakový MD5 hash hesla kordinátora. Pro vytvoření je možno použít např. webovou stránku: http://atutility.com/page/online-md5-calculator. Generické heslo korodinátora je **ddp**.

7.9. Zabezpečení

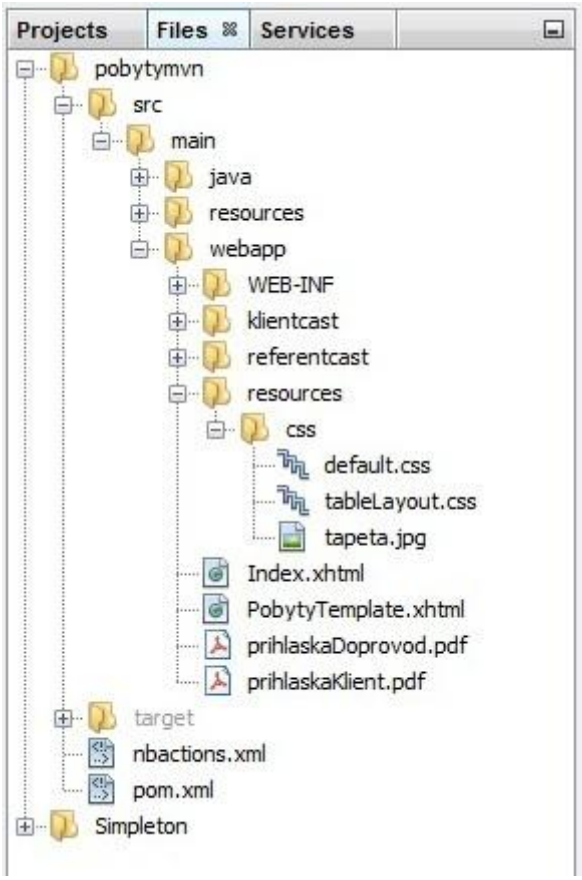
Vzhledem k tomu, že veškerá data, která aplikace spravuje, jsou důvěrné povahy, jsou všechny XHTML stránky aplikace umístěny v jedné bezpečnostní doméně, která je deklarována ve standardním průvodním konfiguračním souboru **web.xml**. Pro tuto doménu je stanovena hodnota atributu transport-guarantee jako **CONFIDENTIAL**, což znamená, že veškerý přenos dat se děje protokolem **https**.

Pro autorizaci uživatelů aplikace se nepoužívá standardní mechanismus Java EE. Tím je dosaženo, že aplikace nemusí být deployovaná pouze na aplikační server, ale může být provozována samostatně, pomocí vestavěného webového serveru (např. Jetty, Tomcat atp.). Hlavní uživatel aplikace je koordinátor (obdoba uživatel root), jehož heslo je jedním z parametrů aplikace. Pro ostatní uživatele si aplikace realizuje vlastní bezpečnostní doménu (realm) v tabulce **UZIVATELE**. V této tabulce jsou hesla uživatelů zaznamenána MD5

hashem. Autorizace přístupu k jednotlivým stránkám je zajištěna autorizační funkcí **isAllowed**, kterou poskytuje frameworku Simpleton. Předeklarací této metody je akceptován resp. odmítnut povolený přístup přihlášeného uživatel k dané stránce:

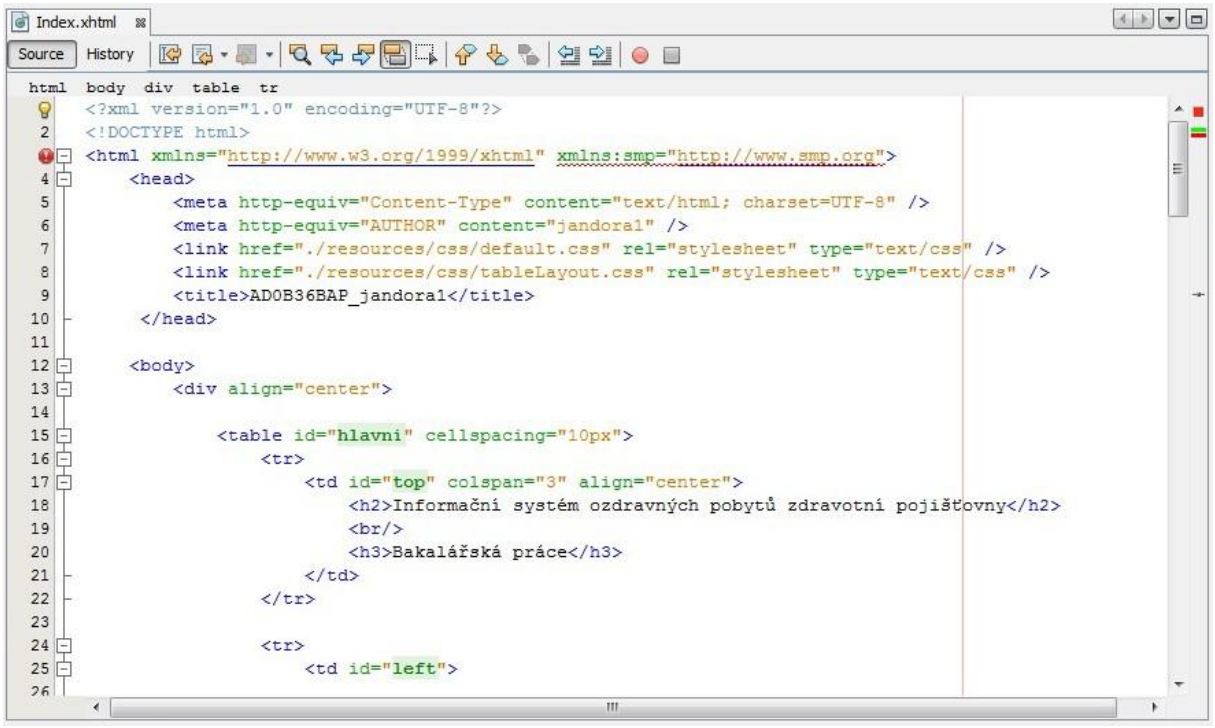
```
@Override
public void isAllowed(String resource, HttpServletRequest
                                        request) throws SmpException {
  Uzivatel uzivatel = (Uzivatel)
              request.getSession().getAttribute(Uzivatel.USER);
  if (uzivatel == Uzivatel.KOORDINATOR) return;
  if (resource.startsWith("/Index")) return;
  if (resource.startsWith("/referentcast/ReferentHlavni"))
                                                          return;
  if (resource.startsWith("/referentcast/LoginForm"))return;
  if (resource.startsWith("/klientcast")) return;
  . . .
  throw new AccessViolationException(
         "Nedovolený přístup: " + resource + " - " + uzivatel);
}
```

7.10. Webová část

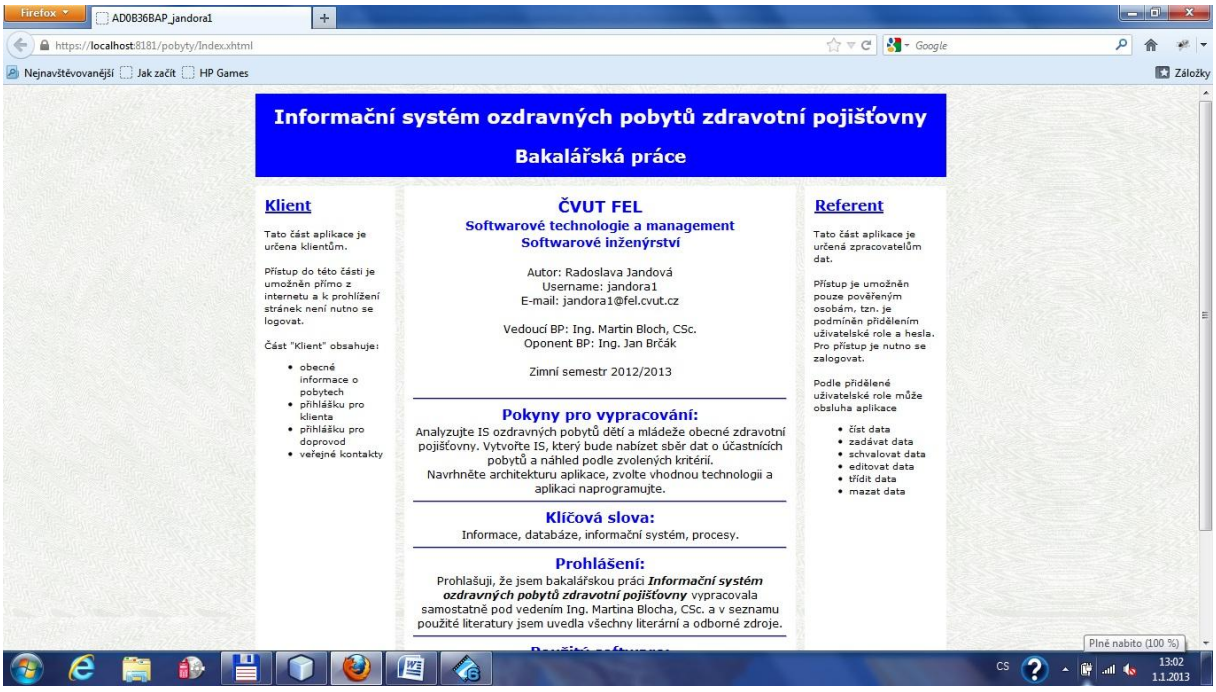


Obr. 08 Náhled uspořádání adresáře webapp.

Stránky pro webovou část jsou vytvářeny jako stránky xhtml a jsou organizovány v samostatném adresáři **webapp**, kde jsou dále rozděleny do logických celků (adresářů). Přímo v adresáři **webapp** jsou uloženy pouze dva hlavní xhtml soubory **Index.xhtml** a **PobytyTemplate.xhtml**, ze kterých je tvořena úvodní stránka webové části.



Obr. 09 Náhled kódu stránky Index.xhtml.



Obr. 10 Náhled stránky Index.xhtml ve webovém prohlížeči.

Stránky jsou tvořeny jako standardní xhtml stránky. Kaskádní styly stránek jsou uloženy odděleně v samostatném podadresáři webapp **resources**, a to ve dvou souborech **default.css** a **tableLayout.css**.

Webové stránky jsou organizovány do následujících podresářů:

- **WEB-INF** = konfigurační soubory k webovým stránkám,
- **klientcast** = stránky tvořící klientskou část aplikace,
- **referentcast** = stránky tvořící uživatelskou část aplikace. Adresář obsahuje další podadresáře:
  - **doprovod** = stránky určeny pro organizaci a zpracování dat doprovodu,
  - **klient** = stránky určeny pro organizaci a zpracování dat klientů,
  - **prihlaska** = stránky určeny pro organizaci a zpracování přihlášek,
  - **turnus** (+ podadresář **skupina**) = stránky určeny pro organizaci a zpracování dat turnusu a skupin,
  - **uzivatel** = stránky určeny pro organizaci a zpracování dat uživatelů.

8. Testování

Testování bylo připraveno podle předběžného plánu testování - viz bod 4. této části bakalářské práce.

8.1. Verifikace

Kontrola a testování kódu aplikace byly prováděny pravidelně v průběhu vývoje tak, aby byla zachována soudržnost modulů a kvalita vyvíjeného produktu. Pro testování některých tříd byly připraveny testy s využitím frameworku JUnit.

Příklad testu **MestoTest.java** pro třídu **pobyty.model.Mesto**:

Třída Město:

```
public class Mesto {
    public static int MESTO_SIZE = 30;
    public static String colName() {
        return Mesto.class.getSimpleName();
    }

    public static Mesto validate(String mesto) throws
                                          ValidationException {
        if (mesto == null) {
            throw new ValidationException("Prosím vyplňte
                                                    město.");
        }
        mesto = mesto.trim();
        if (mesto.isEmpty()) {
            throw new ValidationException("Prosím vyplňte
                                                    město.");
        }
        if (mesto.length() > MESTO_SIZE) {
```

```
            throw new ValidationException(String.format("Město
                  může být maximálně %d znaků.", MESTO_SIZE));
        }
        return new Mesto(mesto);
    }
    private String mesto;

    public Mesto(String mesto) {
        this.mesto = mesto;
    }

    public String getMesto() {
        return mesto;
    }

    @Override
    public String toString() {
        return mesto;
    }

    @Override
    public boolean equals(Object obj) {
        if (obj == null) {
            return false;
        }
        if (getClass() != obj.getClass()) {
            return false;
        }
        return mesto.equals(((Mesto) obj).getMesto());
    }

    @Override
    public int hashCode() {
        int hash = 3;
        hash = 79 * hash + Objects.hashCode(this.mesto);
        return hash;
    }
}
```

Třída MestoTest:

```
public class MestoTest {

    public MestoTest() {
    }

    @Test
    public void testValidate() throws Exception {
        System.out.println("validate");
        Mesto expResult = new Mesto("Praha");
```

```
        Mesto result = Mesto.validate("Praha");
        assertEquals(expResult, result);
    }

    @Test
    public void testGetMesto() {
        System.out.println("getMesto");
        Mesto instance = new Mesto("Praha");
        String expResult = "Praha";
        String result = instance.getMesto();
        assertEquals(expResult, result);
    }

    @Test
    public void testToString() {
        System.out.println("toString");
        Mesto instance = new Mesto("Praha");
        String expResult = "Praha";
        String result = instance.toString();
        assertEquals(expResult, result);
    }
}
```

Obr. 11 Výsledek testu.

8.2. Validace

Validační testování bude provedeno jednak pracovníky zákazníka a jednak nezávislými testery. Je žádoucí, aby byl každý test proveden vícekrát, vždy jiným testerem na jiném počítači. Pro validační testování budou vypracovány testovací scénáře podle dále uvedeného vzoru. Popis činností je uveden v bodu 8.3.

V každém scénáři bude stanoven rozsah činností a postup testování tak, aby byla funkčnost systému kompletně otestována. Tester vyplňuje pouze nezabarvenou část scénáře a u testovaných činností může vybrat vždy jednu ze dvou variant:

- "vyhovuje" = test proběhl úspěšně,
- nevyhovuje = test neproběhl úspěšně, při testování se vyskytly chyby. V tomto případě bylo nutno doplnit i slovní komentář se stručným popisem chyby.

TESTOVACÍ SCÉNÁŘ			
Testovací role:			
Přístupové údaje – username:		**heslo:**	
Cíl testu:			
Zařízení, na kterém je test prováděn:			
Operační systém:		**Typ internet. prohlížeče:**	
PRŮBĚH TESTOVÁNÍ			
Činnost		**Vyhovuje**	**Nevyhovuje**
Hodnocení pracovního prostředí:			
Hodnocení uživatelského manuálu:			
Komentář k prováděnému testu:			
Zhodnocení testu:			
Datum testování:	**Počet přiložených listů:**	**Podpis:**	

Obr. 12 Vzor testovacího scénáře pro validační testování.

8.3. Seznam testů a přehled testovaných činností

Pro testování aplikace budou připraveny testy v uvedeném rozsahu.

8.3.1. Přihlášení klienta k pobytům – nový klient

Cíl testu: zavedení nového záznamu do databáze klientů a přihlášení klienta k pobytu, tzn. stav přihlášky PŘIHLÁŠEN.

Průběh testu:

- otevřít aplikaci,
- zhodnotit přehlednost a srozumitelnost pracovní plochy,
- v menu vybrat „*Přihlášení k pobytu – nový klient*“,
- vyplnit několik kolonek a vyčistit formulář,
- do ID kódu vyplnit kód 123456,
- vyplnit částečně formulář a odeslat,
- vyplnit celý formulář a odeslat,
- systém ohlásí duplicitní ID kód – zvolit vlastní kód a odeslat,
- po odeslání zvolit „*Návrat zpět do přihlášky*“,
- editovat údaje,
- „*Potvrdit*“,
- v menu vybrat „*Vyhledat přihlášku*“,
- vyhledat přihlášku a zkontrolovat stav přihlášky = PŘIHLÁŠEN,

- zavřít aplikaci.

8.3.2. Přihlášení klienta k pobytům – klient v databázi existuje

Cíl testu: přihlášení již existujícího klienta k pobytům.
Přihlašovací údaje: ID kód: **123456**, příjmení: **Jandová**
Průběh testu:

- otevřít aplikaci,
- zhodnotit přehlednost a srozumitelnost pracovní plochy,
- v menu vybrat „*Obsazenost turnusu*“,
- v menu vybrat „*Vyhledat přihlášku*“,
- vyplnit chybně požadované údaje,
- vyčistit formulář,
- vyplnit přihlašovací údaje,
- stav přihlášky = „*Klient není dosud přihlášen k pobytu*“,
- editovat údaje,
- odeslat záznam,
- v menu vybrat „*Přihlášení k pobytu – evidovaný klient*“,
- vyplnit chybně ID kód nebo příjmení a odeslat,
- vyčistit formulář,
- vyplnit správně všechny údaje a odeslat,
- zvolit „*Návrat do přihlášky*“,
- editovat údaje,
- zkontrolovat stav přihlášky = PŘIHLÁŠEN,
- „*Potvrdit*“,
- v menu vybrat „*Vyhledat přihlášku*“,
- vyhledat přihlášku a zkontrolovat stav = PŘIHLÁŠEN,
- zavřít aplikaci.

8.3.3. Přihlášení doprovodu k pobytům

Cíl testu: přihlášení doprovodu k pobytu.
Průběh testu:
Činnosti ze strany doprovodu:

- otevřít aplikaci,
- zhodnotit přehlednost a srozumitelnost pracovní plochy,
- v menu vybrat „*Přihlášení k pobytu – doprovod*“,
- vytisknout přihlášku a vyplnit,
- zavřít aplikaci.

Činnosti ze strany referenta:
Přihlašovací údaje: uživatelské jméno: **novaj**, heslo: **12345**

- otevřít aplikaci a přihlásit se,
- zhodnotit přehlednost a srozumitelnost pracovní plochy,
- v menu vybrat „*Doprovod*“,
- „*Vyhledat doprovod*“ podle příjmení – test, zda již není doprovod zaveden v databázi,
- zvolit „*Přidat doprovod*“,
- vyplnit několik kolonek a odeslat,

- vyčistit formulář,
- vyplnit formulář ve všech kolonkách a odeslat,
- „*Vyhledat doprovod*“ podle příjmení,
- editovat údaje,
- „*Uložit a zavřít*“,
- „*Vyhledat doprovod*“ podle příjmení,
- smazat,
- znovu zadat doprovod,
- zavřít aplikaci.

8.3.4. Test činností referenta

Cíl testu: test činností povolených uživatelské roli referent.
Přihlašovací údaje: uživatelské jméno: **koloi**, heslo: **12345**

- otevřít aplikaci,
- zvolit v menu „*Přihlásit* se“,
- zadat chybně přihlašovací údaje a odeslat,
- vyčistit formulář,
- zadat správně přihlašovací údaje a odeslat,
- zhodnotit přehlednost a srozumitelnost pracovní plochy,
- v menu vybrat „*Změna hesla*“,
- zadat rozdílné hodnoty do položek „*Heslo*“ a „*Ověření hesla*“,
- zadat změnu hesla správně (nové heslo zapsat do formuláře) a potvrdit,
- odhlásit se a přihlásit se pod novým heslem,
- v menu vybrat „*Klienti*“,
- v části „*Vyhledat klienta*“ vyhledat klienta „*Xxx*“,
- editovat údaje klienta,
- smazat klienta,
- v menu vybrat „*Doprovod*“,
- v části „*Vyhledat doprovod*“ vyhledat doprovod „*Xxx*“,
- editovat údaje,
- „*Uložit a zavřít*“,
- znovu „*Vyhledat doprovod*“ „*Xxx*“,
- smazat,
- zavřít aplikaci.

8.3.5. Test činností lékaře

Cíl testu: test činností povolených uživatelské roli lékař.
Přihlašovací údaje: uživatelské jméno: **pavep**, heslo: **12345**

- test všech činností referenta – viz bod 8.3.4.,
- v menu vybrat „*Schválit*“,
- u konkrétního jména kliknout na „*Schválit*“,
- potvrdit tlačítkem „*Schválit*“ a provést kontrolu, že schválený klient již není v tabulce klientů ke schválení,
- u konkrétního jména kliknout na „*Schválit*“,
- kliknout na „*Zpět*“ a provést kontrolu, že klient je stále v tabulce,
- v menu vybrat „*Přihlášky*“,
- vybrat klienta, který nebyl schválen a vybrat „*Edit*“,

- zvolit stav přihlášky „*zrušen*“ a „*Potvrdit*“,
- zkontrolovat u klienta stav přihlášky – stav „*zrušen*“,
- v menu vybrat „*Schválit*“ a provést kontrolu, že klient není v tabulce klientů ke schválení,
- zavřít aplikaci.

8.3.6. Test činností koordinátora

Cíl testu: test činností povolených uživatelské roli koordinátor.
Přihlašovací údaje: uživatelské jméno: **xxx**, heslo: **ddp**

- test všech činností referenta – viz bod 8.3.4.,
- test všech činností lékaře – viz bod 8.3.5.,
- v menu vybrat „*Turnus*“ - přidat nový turnus, smazat turnus, editovat turnus,
- přidat skupinu a zadat věk klientů mimo rozsah 6 – 16 let,
- přidat skupinu a zadat pouze věk klientů mimo rozsah 6 – 16 let,
- přidělit skupině doprovod přes položku „*editovat doprovod*“,
- zařadit klienta do skupiny,
- smazat skupinu,
- v menu vybrat položku „*Zařadit*“ a potvrdit klienta k zařazení,
- provést kontrolu, že se stav přihlášky u klienta změnil na „zařazen“,
- v menu vybrat položku „*Uživatelé aplikace*“,
- přidat nového uživatele, smazat uživatele, editovat údaje,
- zavřít aplikaci.

Závěr

Ve své bakalářské práci jsem vycházela z vlastních zkušeností, které jsem získala v rámci své práce ve Všeobecné zdravotní pojišťovně ČR. Více jak 15 let jsem na pozici koordinátora zajišťovala realizaci dětských ozdravných pobytů u moře Mořský koník. Do mé pracovní náplně patřila správa metodiky a agenda organizace a třídění dat.

Data byla pořizována a spravována v aplikace MS Excel, který je sice jednoduchý na obsluhu, ale zcela nevhodný pro uchování dat z důvodu velmi nízkého zabezpečení proti poškození či ztrátě dat.

V průběhu mé praxe jsme několikrát s kolegy řešili možnost zařadit modul ozdravných pobytů do databázového systému VZP ČR. Riziko možného ukončení pobytů ale bylo příliš vysoké s ohledem na odhadované finančními náklady.

Uvítala jsem tak možnost, realizovat informační systém ozdravných pobytů v rámci své bakalářské práce.

Problematika realizace ozdravných pobytů je mi velmi dobře známa. S ohledem na omezení rozsahu bakalářské práce ale nebylo možné zpracovat celou šíři daného problému. Byly tak vybrány jen stěžejní části dané problematiky, aby bylo možno prezentovat alespoň základní funkcionalitu požadovanou v rámci realizace ozdravných pobytů, tj. přihlášení klienta k pobytu, schválení přihlášky a zařazení klienta do konkrétní skupiny.

V průběhu testování produktu se ukázalo, že aplikace splňuje základní požadavky na organizaci a zpracování dat ozdravných pobytů a umožňuje event. rozšíření o další moduly.

V rámci zpracování jsem měla možnost otestovat framework Simpleton, který mi za tímto účelem poskytnul vedoucí mé bakalářské práce. Ukázalo se, že uvedený framework je, i přes svou jednoduchost, pro daný účel dostačující.

Seznam obrázků

Seznam použité literatury

[1] POUR Jan, *Informační systémy a technologie* [online]. Poslední aktualizace neuvedena [cit. 2011-11-19]. Dostupné z WWW: <http://www.vsem.cz/data/data/sis-ukazky-kapitol/uc_ist_kapitola.pdf>.

[2] Doc. Ing. TVRDÍKOVÁ Milena, CSc., *Aplikace moderních informačních technologií v řízení firmy*. Grada Publishing, a.s., Praha 2008. ISBN: 978-80-247-2728-8.

[3] RNDr. JUDr. ŠMÍD Vladimír, *Management informačního systému* [online]. Poslední aktualizace neuvedena [cit. 2011-11-21]. Dostupné z WWW: <http://www.fi.muni.cz/~smid/managis.html>.

[4] VYMĚTAL Dominik, *Informační systémy v podnicích, teorie a praxe projektování*. Grada Publishing, a.s., Praha 2009. ISBN: 978-80-247-3046-2.

[5] *Informace* [online]. Poslední aktualizace 14. 12. 2011 [cit. 2011-12-15]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Informace>.

[6] RADEK KUČERA & daughter, *Pojem informace* [online]. ABZ slovník cizích slov, 2005-2006. Poslední aktualizace neuvedena [cit. 2011-12-15]. Dostupné z WWW: <http://slovnik-cizich-slov.abz.cz/web.php/slovo/informace>.

[7] prof. Ing. BASL Josef, CSc., *Podnikové informační systémy*. Grada Publishing, a.s., Praha 2008. ISBN: 978-80-247-2279-5.

[8] KUČEROVÁ Helena, *Definice informace. Data – informace, znalosti* [online]. Vyšší odborná škola informačních služeb, Praha 4, Pacovská 350/4. Poslední aktualizace 18. 10. 2011 [cit. 2011-12-19]. Dostupné z WWW: <http://info.sks.cz/users/ku/ZIZ/inform1.htm>.

[9] Ing. DANEL Roman, PhD., *Informační systém. Úvod a základní pojmy v oblasti informačních systémů* [online]. Poslední aktualizace neuvedena [cit. 2011-12-21]. Dostupné z WWW: <homel.vsb.cz/.../1%20Informacni%20systemy%20-%20uvod.ppt>.

[10] HOUDA Michal, *Informace, informační systémy, informační společnost* [online]. Poslední aktualizace xxx [cit. 2011-12-21]. Dostupné z WWW: <www2.ef.jcu.cz/~houda/infa/prednasky/ 01i-systemy-print6.pdf>.

[11] *Proces* [online]. Poslední aktualizace 6. 7. 2011 [cit. 2011-12-21]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Proces>.

[12] *Proces (program)* [online]. Poslední aktualizace 1. 7. 2011 [cit. 2011-12-21]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Proces_(program)>.

[13] EnviWeb, *Proces* [online]. Poslední aktualizace neuvedena [cit. 2011-12-21]. Dostupné z WWW: <http://www. enviweb.cz/eslovnik/186>.

[14] *Databáze* [online]. Poslední aktualizace 25. 6. 2011 [cit. 2011-12-21]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Databáze>.

[15] *Hierarchická databáze* [online]. Poslední aktualizace 23. 9. 2011 [cit. 2011-12-21]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Hierarchická_databáze>.

[16] *Síťová databáze* [online]. Poslední aktualizace 23. 9. 2011 [cit. 2011-12-21]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Síťová_databáze>.

[17] *Relační databáze* [online]. Poslední aktualizace 15. 11. 2011 [cit. 2011-12-21]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Relační_databáze>.

[18] Jandová Radoslava, *Database Models, Presentation Y04A2L* [online]. Poslední aktualizace 25. 3. 2011 [cit. 2011-12-21]. Dostupné z WWW: <http://www.vrstevnice.com/akce/grandaction/vskola/ 4semestr/4semestr.html#angl>.

[19] Doc. RNDr. Ing. Šeda Miloš, PhD., *Databázové systémy* [online]. Poslední aktualizace 2002 [cit. 2012-01-01]. Dostupné z WWW: <http://www.uai.fme.vutbr.cz/~mseda/DBS02_BS.pdf>.

[20] *Tovarna.cz, Informační systémy* [online]. Poslední aktualizace neuvedena [cit. 2012-01-02]. Dostupné z WWW: <http://www.tovarna.cz/cz/sluzby/informacni-systemy/>.

[21] Ing. Božena Mannová, Ph.D., *AD7B36SI2, stránky předmětu* [online]. Poslední aktualizace 14. 11. 2011 [cit. 2012-01-02]. Dostupné z WWW: <https://service.felk.cvut.cz/courses/Y36SI2/>.