Mendelova univerzita v Brně
Provozně ekonomická fakulta

Komplexné riešenie HelpDesk pre podporu informačného systému QI

Diplomová práca

Vedúci práce:
RNDr. Zuzana Prišťáková, Ph.D.

Bc. Barbora Štupáková

Brno 2017

Týmto by som chcela poďakovať vedúcej diplomovej práce RNDr. Zuzane Priščákovej Ph.D. za odborné vedenie pri písaní diplomovej práce. Taktiež by som chcela poďakovať rodine a priateľovi za podporu a spoločnosti it2b s.r.o. za odborné konzultácie.

Čestné prehlásenie

Prehlasujem, že som túto prácu: **Komplexné riešenie HelpDesk pre podporu informačného systému QI**
vypracovala samostatne a všetky použité zdroje a informácie sú uvedené v zozname použitej literatúry. Súhlasím, aby moja práca bola zverejnená v súlade s § 47b zákona č. 111/1998 Sb., o vysokých školách v znení neskorších predpisov, a v súlade s platnou *Směrnicí o zveřejňování vysokoškolských závěrečných prací.*
Som si vedomá, že sa na moju prácu vzťahuje zákon č. 121/2000 Sb., autorský zákon, a že Mendelova univerzita v Brně má právo na uzatvorenie licenčnej zmluvy a použitie tejto práce ako školského diela podľa § 60 odst. 1 Autorského zákona.
Ďalej sa zaväzujem, že pred spísaním licenčnej zmluvy o využití diela inou osobou (subjektem) si vyžiadam písomné stanovisko univerzity o tom, že predmetná licenčná zmluva nie je v rozpore s oprávnenými záujmami univerzity, a zaväzujem sa uhradiť prípadný príspevok na úhradu nákladov spojených so vznikom diela, a to až do ich skutočnej výšky.

V Brne, dňa 1.1.2017 .....

Abstract

Štupáková, B. Complex solution to support information system QI. Diploma thesis. Brno: Mendel University in Brno, 2017

Purpose of this theses is create and implement complex solution for making requests. Helpdesk synchronizes data with information system QI. This web application is implemented in PHP, more specifically framework Nette.

Keywords

Web application, Nette Framework, information system QI, helpdesk

Abstrakt

Štupaková, B. Komplexné riešenie HelpDesk pre podporu informačného systému QI. Diplomová práca. Brno: Mendelova univerzita v Brne, 2017

Cieľom tejto práce je vytvoriť a implementovať komplexné riešenie pre vytváranie požiadaviek. Helpdesk synchronizuje dáta s informačným systémom QI. Táto webová aplikácia je implementovaná v PHP, konkrétne vo frameworku Nette.

Kľúčové slová

Webová aplikácia, Nette Framework, informačný systém QI, helpdesk

Obsah

1 Úvod práce

V dnešnej dobe je pre neustále zväčšujúce sa množstvo spoločností neodmysliteľnou súčasťou systém pre spracovávanie a uchovávanie informácií, pre podporu riadenia firmy a evidencie podnikových procesov, a práve preto sa k týmto informačným systémom dostáva viac a viac ľudí.

Rôznorodosť spoločností spôsobuje neustále zmeny v pohľade na spracovávané dáta. V súčasnosti takmer každá spoločnosť vyžaduje individuálny prístup a má špecifické požiadavky na informačný systém. Preto je nutné vyhovieť zákazníkom a vytvoriť systém, ktorý bude čo najviac prívetivý. Práve kvôli tomu sa služba podpory informačných systémov čoraz viac zvyšuje. Komunikačný kanál, ktorý sprostredkuje nové požiadavky alebo problémy od zákazníkov smerom k poskytovateľom sa v praxi definuje ako helpdesk, technická podpora alebo informačné centrum. Každý z týchto názvov zastrešuje starostlivosť o zákazníka a prispôsobenie systému alebo riešenie problémov podľa potrieb a požiadaviek od zákazníka. Existuje viacero možností ako túto službu pre zákazníkov poskytnúť. Môže to byť formou telefonickej podpory, emailovej alebo webovej technickej podpory.

Hlavným dôvodom, prečo sa spoločnosť it2b s.r.o. (It2b, 2016) rozhodla využívať helpdesk, bolo zrýchlenie komunikácie medzi zákazníkmi a spoločnosťou it2b s.r.o., ktorá prevádzkuje informačný systém QI.

2 Cieľ a metodika práce

2.1 Cieľ práce

Cieľom diplomovej práce je nahradiť súčasný systém nejednotného zadávania požiadaviek v informačnom systéme QI jedným komplexným riešením. Komunikácia ohľadom požiadaviek medzi it2b s.r.o. a zákazníkom sa presunie na stranu webovej aplikácie, a tým sa odbúra nutnosť využívania emailovej komunikácie. Pre dosiahnutie cieľa, bude vytvorená webová aplikácia HelpDesk, ktorá bude pomocou webových služieb komunikovať s informačným systémom QI a uľahčí tak prácu zákazníkom ale aj zamestnancom systému. Od systému sa očakáva zlepšenie kvality požiadaviek, spresnenie a zrýchlenie riešenia, zvýšenie produktivity a spokojnosti zákazníkov.

V rámci práce je nutné zamerať sa na súčasný stav zadávania požiadaviek v systéme QI a analyzovať vnútorné procesy dodávateľa informačného systému. Konkrétny návrh systému bude vytvorený pomocou jazyka UML. Komunikácia aplikácie HelpDesk s informačným systémom QI bude zabezpečená webovými službami SOAP a REST, kde dáta budú predávané pomocou XML. Samotná štruktúra XML dokumentu bude vytvorená pomocou štandardu XSD. Taktiež bude vykonaná analýza aktuálnych riešení helpdesk na trhu.

2.2 Metodika práce

Správna metodika je nevyhnutná pre splnenie cieľa práce. Je potrebné stanoviť metodiku pre celý životný cyklus vytvárania aplikácie helpdesk. Budú vybrané technológie pre vývoj tak, aby boli prijateľné pre samotnú implementáciu.

Oboznámenie sa s informačným systémom QI je prvou fázou vývoja. Systém sa skladá z rôznych modulov, no podstatné je zamerať sa na časti súvisiace so zadávaním a riešením požiadaviek. Následne budú sformulované funkčné a nefunkčné požiadavky na systém podľa požiadaviek zadávateľa projektu. Po kompletizácií požiadaviek je nutné zamerať sa na možnosti synchronizácie dát. Kvôli získavaniu dát je potrebné rozdeliť aplikáciu na moduly, a to na modul synchronizácie dát a modul pre zadávanie a zobrazovanie dát.

Modul synchronizácie dát vyžaduje rozbor dát systému QI a vytvorenie entitne-relačného diagramu. Podľa týchto dát bude potrebné vytvoriť XSD štruktúru a následne XML dokument. Taktiež pre modul zadávania a zobrazovania dát sa vytvoria UML diagramy a to use case, diagram tried, stavový a procesný diagram. Podľa týchto návrhov bude realizovaná celá aplikácia, preto musia zachytiť všetkú funkcionalitu. Musia tam byť zachytené všetky požiadavky a väzby podľa systému QI.

Po schválení návrhu aplikácia helpdesk prejde do fázy implementácie. Aplikácia bude vytvorená v jazyku PHP, konkrétne vo frameworku Nette. Pre databázu bolo zvolené MySQL. Vývoj aplikácie začne modulom pre synchronizáciu dát. Až po dokončení môže začať implementácia modulu pre zadávanie a zobrazovanie dát.

Celý proces vývoja aplikácie bude prebiehať pod dozorom zadávateľa projektu, ktorému sa výsledky budú prezentovať postupne. Aplikácia sa bude upravovať podľa nových požiadaviek a pripomienok na zmenu.

V nasledujúcich kapitolách budú opísané princípy a postup pri návrhu a implementácií aplikácie.

3 Teoretické východiská

V tejto kapitole budú opísané technológie a princípy, ktoré budú použité pri návrhu a vývoji aplikácie. Táto časť sa zameriava ako na východiská k návrhu aplikácie, tak aj k jej implementácií.

3.1 Návrhové vzory

Pri návrhu softwaru sa často vyskytnú problémy, ktoré je možné riešiť všeobecnými postupmi. Tieto postupy sú známe ako návrhové vzory. Pri vývoji bolo použítých niekoľko návrhových vzorov, ktoré sú opísané v tejto kapitole.

3.1.1 MVC

Jednoduchosť a dostupnosť jazyka PHP môže spôsobiť vytváranie aplikácií, ktoré nie je možné spravovať. Typickým príkladom je spojenie modelu, pohľadu a radiča do jedného skriptu. To môže spôsobiť neprehľadnosť, ťažké spravovanie a v budúcnosti nemožnosť rozšírenia aplikácie. V správne navrhnutej aplikácií pomocou MVC nie sú žiadne skripty len komponenty. (Lecky-Thompson, 2010)

MVC je návrhový vzor, ktorý delí používateľské rozhranie, riadiacu logiku a dátový model do nezávislých komponentov. Vývojári zistili, že pri rozdelení zodpovednosti podľa vzoru MVC vzniká menej väzieb, čím sa kód ľahšie píše a udržiava. (Ruby, 2011)

Model

Model je dátový a najmä funkčný základ celej aplikácie. Obsahuje aplikačnú logiku od prihlásenia používateľa, zmenu v databáze až po akúkoľvek akciu používateľa. Model sa pripojí priamo k radiču a o existencii pohľadu nevie. Spravuje si svoj vnútorný stav, ktorý je prostredníctvom volania funkcií možné zmeniť. (Lecky-Thompson, 2010)

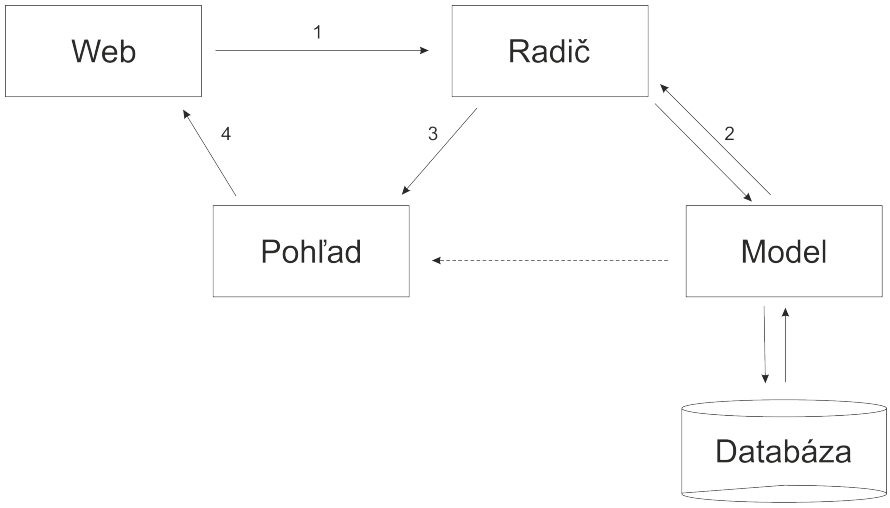
View

Pohľad je koncový bod aplikácie, ktorý prezentuje data od radiča. Pri vytváraní sa používa šablonovací systém ktorý vie, ako sa majú zobraziť jednotlivé komponenty. V pohľade existuje aj určitá logika, ako napríklad cykly pre zobrazenie dát v tabuľke alebo podmienka, či sa má daná položka zobraziť alebo nie. (Lecky-Thompson, 2010)

Controler

Pre spracovávanie požiadaviek používateľa sa využíva radič (controler). Ak si používateľ vyžiada stránku (viď obr. 1), čo je zobrazené prvým krokom v obrázku,

pošle sa požiadavka radiču. Následne radič podľa potreby vyžiada dáta od modelu (krok číslo 2). Po získaní týchto dát ich radič pošle pohľadu (krok 3) a ten tieto informácie zobrazí používateľovi (krok číslo 4). (Lecky-Thompson, 2010)



Obr. 1: MVC (Lecky-Thompson, 2010)

3.1.2 ORM

Ako už z názvu vyplýva, ORM alebo objektovo relačné mapovanie umožnuje mapovanie relačnej databázy na objekty. Objekt je reprezentovaný jedným riadkom v tabuľke relačnej databázy. Ide o abstraktnú vrstvu, kde je možné bežné databázové funkcie ako vytvorenie, aktualizácia, odstránenie záznamu z tabuľky vykonávať bez potreby písania SQL dotazov. ORM zaisťuje vyššiu prenositeľnosť na iné databázové platformy, ale aj menšie množstvo kódu, a tým aj zrýchlenie vývoja a redukciu počtu chýb. (Lecky-Thompson, 2010)

Pre potreby aplikácie helpdesk bol využitý ORM Framework Doctrine 2, ktorý je bližšie predstavený v kapitole 3.2 Doctrine.

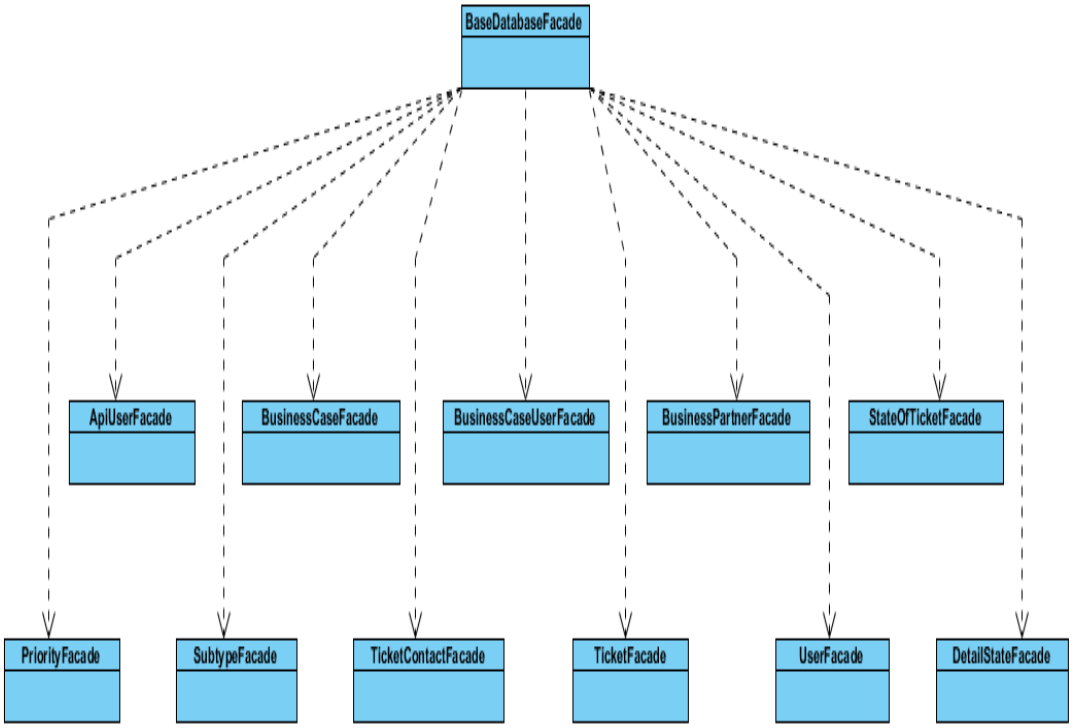
3.1.3 Fasáda

Návrhový vzor *Fasáda* poskytuje zjednodušené rozhranie pre sériu rozhraní. Toto rozhranie uľahčuje používanie základného systému, taktiež zjednodušuje existujúce rozhranie a výrazne redukuje počet tried, ktoré je nutné využívať. Pri dodržaní princípu čo najmenšieho počtu tried, známeho aj pod názvom *Information Hiding Principle* (Princíp skrývania informácií), sa zabráni silnému previazaniu tried. (Böhmer, 2012)

Častým použitím fasády je databázová fasáda. Tá umožňuje skryť prístup k databáze a k jazyku SQL za príslušnými metódami. Hlavnou výhodou je, že pri zmene štruktúry databázy alebo dokonca pri prechode na iný databázový systém sa

nemusí meniť kód aplikácie. Namiesto množstva malých zmien v celom kóde, stačí upraviť kód na jednom mieste a to v kóde fasády. (Böhmer, 2012)

Pri implementácií tohto návrhového vzoru je nutné určiť triedy a objekty, ktoré majú byť za fasádou schované a definovať operácie, ktoré fasáda bude umožňovať. Následne sa musia implementovať triedy fasády a vytvoriť metódy, cez ktoré bude možné pristupovať ku schovaným komponentom a implementovať metódy, ktoré umožnia zjednodušený prístup. (Böhmer, 2012)



Obr. 2: Diagram tried návrhového vzoru fasáda

Pre potreby helpdesku bolo nutné využívať databázovú fasádu, ktorá obaľuje Doctrine. Štruktúra fasád, ktorá bola využitá v helpdesku, je znázornená na obrázku číslo 2.

3.1.4 Adaptér

Návrhový vzor Adaptér umožňuje prispôsobenie rozhrania jednej triedy rozhraním inej triedy a tým umožní spoluprácu tried, ktoré by neboli schopné spolupráce, kvôli nekompatibilnému rozhraniu. (Böhmer, 2012)

Pre vytvorenie adaptéru je nutné vykonať niekoľko krokov:

- lokalizovanie rozdielu medzi ponúkaným a požadovaným rozhraním,
- implementácia novej triedy, ktorá bude poskytovať požadované rozhranie,

- vytvorenie spôsobu, ako predať adaptéru objekt, ktorý má byť prispôsobený,
- následne je nutná implementácia všetkých metód, ktoré sú vyžadované rozhraním a delegovať požiadavky ďalej na metódy pôvodného objektu,
- využívanie objektu adaptéra, pomocou ktorého obalíme pôvodný objekt. (Böhmer, 2012)

Na prvý pohľad sa návrhový vzor Adaptér podobá návrhovému vzoru Fasáda. Ale fasáda zjednodušuje existujúce rozhranie, na rozdiel od adaptéra, ktorý prispôsobuje jedno rozhranie druhému. (Böhmer, 2012)

3.1.5 Builder

Builder umožňuje oddeliť konštrukciu komplexných objektov od ich reprezentácie tak, aby bolo možné rovnakým konštrukčným procesom vytvoriť rôzne reprezentácie. Následne je vytvorenie nových reprezentácií veľmi jednoduché. (Böhmer, 2012)

3.2 Doctrine

Doctrine je ORM (Object-relation mapper viď kapitola 3.1.2) pre jazyk PHP 5.4+, ktorý poskytuje vysokú mieru abstrakcie databázovej vrstvy, čo umožnuje pracovať s dátami ako s objektami. Doctrine sa skladá z 3 hlavných balíkov:

- Common - táto vrstva obsahuje rozhrania, triedy a knižnice, ktoré sú všeobecné. Je tvorený komponentami, ktoré neobsahujú žiadne závislosti na ostatných vrstvách, čo umožňuje používať vrstvu Common aj samostatne. Namespace balíčka Common je *Doctrine/Common*.
- DBAL - Namespace DoctrineDBAL primárne rozširuje PDO, ale dokáže pracovať aj s inými databazovými ovládačmi. Taktiež zavádza aj DQL (Doctrine Query Language). Táto vrstva je síce závislá na balíčku Common, ale nezávislá na vrstve ORM.
- ORM - ako už bolo opísané, ORM zaisťuje mapovanie záznamov databázy na objekty. Táto vrstva je závislá na predchádzajúcich. (Getting Started with Doctrine, 2016)

3.2.1 Terminológia

V nasledujúcich podkapitolách budú opísané súčasti ORM Doctrine.

Entity

Entity sú PHP objekty, ktoré nesú dáta o konkrétnom zázname. Nemajú prístup k databáze a ich úlohou je uchovávať a predávať dáta. Pomocou entít, ktoré sú väzobné, sa môžeme dotazovať na iné entity podľa anotácií, ktoré sú v rámci nej

vytvorené. Taktiež môže obsahovať rozširujúcu funkcionalitu ako generovanie predvolených hodnôt, alebo validáciu jednotlivých parametrov. (Working with Objects, 2016)

Anotácie

Anotácie sú dôležitou súčasťou ORM Doctrine. Udávajú sa tam informácie o samotnej entite, ale taktiež o atribútoch, ich typoch a väzbách. Jednotlivé anotácie majú presnú štruktúru, ktorú je nutné dodržiavať, a ktorá je nasledujúca:

- *Table* - každá entita obsahuje anotáciu Table, ktorou sa označuje, aká tabuľka v databáze je priradená danej entite.
- *Column* - anotácia stĺpec nesie informácie o názve atribútu, jeho podrobnejšie vlastnosti môžeme upresniť pomocou:
  - *type* – dátový typ stĺpca,
  - *name* – názov atribútu v databáze,
  - *unique* – určenie unikátnej hodnoty,
  - *nullable* – povolenie nulovej hodnoty,
  - *length* – maximálna dĺžka reťazca,
  - *precision, scale* – presnosť desatinných čísiel.
- *Id* - anotáciou *Id* sa udáva, ktorý atribút je primárnym kľúčom danej entity.
- *GeneratedValue* - zvyšovanie hodnoty v stĺpci o jednotku sa vytvorí pomocou anotácie *GeneratedValue*.
- *JoinColumns* - anotácia nesie údaje o relačnej tabuľke, na ktorú je daný atribút namapovaný. Taktiež sa tu udáva konkrétny stĺpec, pomocou ktorého môžeme mapovanie zrealizovať.
- *OneToMany*
- *ManyToOne*
- *ManyToMany* (Annotations Reference, 2016)

Generovanie entít

Pre vytvorenie entít sa používa funkcia, ktorú voláme z príkazového riadka. Podľa konfigurácie databázy si načíta tabuľky, stĺpce, obmedzenia a vytvorí entity s atribútmi, *get* a *set* funkcie pre každý atribút. Pomocou *get* funkcií dokážeme zistiť, akú hodnotu ma daný atribút alebo mu môžeme nastaviť prostredníctvom *set* funkcie ľubovoľnú hodnotu podľa obmedzení daného atribútu. Príkaz na spustenie generovania entít je nasledujúci *doctrine:generate-entities*.

Entity Manager

Rozhranie, ktoré poskytuje prístupový bod k entitám počas celého životného cyklu, sa nazýva Entity Manager. Ten uľahčuje prístup k načítaniu entít a následnej práci s nimi, ako napríklad aktualizovanie, mazanie a ukladanie.

Pre načítanie entít z databázy je k dispozícií množstvo funkcií. V práci boli využité napríklad tieto:

- find – získanie entity podľa jej názvu a id,
- findOneByUsername – získanie entity podľa používateľského mena,
- getTicketByUser – získavanie požiadaviek podľa používateľa, ktorý ju vytvoril,
- getBusinessCaseByUserAndState() – získavanie obchodného prípadu podľa používateľa a určitého stavu. (Working with Objects, 2016)

Aktualizovanie entity zahŕňa niekoľko krokov a to načítanie danej entity, úpravu a zavolanie metódy flush(), ktorá všetky zmeny uloží do databázy. Vymazanie objektu sa podobne ako aktualizácia skladá z troch krokov. Najprv sa daná entita načíta, zavolá sa funkcia remove() a potvrdenie metódou flush(). (Databases and the Doctrine ORM, 2016)

DQL

DQL (Doctrine Query Language) môžeme definovať ako dotazovací jazyk frameworku Doctrine, ktorý poskytuje výkonné funkcie dotazovania nad databázovými entitami. Jazyk obsahuje niekoľko typov dotazov a to SELECT, UPDATE a DELETE, ktoré majú rovnaký význam ako v jazyku SQL. Dotaz typu INSERT v jazyku DQL neexistuje, pretože entity a ich vzťahy vytvára EntityManager. (Doctrine Query Language, 2016)

Pre ľahšie a rýchlejšie písanie dotazov sa používa sada tried a metód *QueryBuilder*. Tá umožňuje nepísať dotazy ako reťazce, ale ponúka funkcie ako napríklad *select(), from(), where(), order by()*, pomocou ktorých vytvoríme dotaz jednoducho. Objekt QueryBuilder obsahuje všetky metódy potrebné na tvorbu dotazu. Zavolaním metódy *getQuery()*, je vrátený Query object, ktorý sa používa na získanie výsledku dotazu. (Databases and the Doctrine ORM, 2016)

3.3 SOAP

Protokol SOAP (Simple Object Access Protocol) je jeden z najrobustnejších protokolov pre webové služby. Pri využívaní umožní vyhľadávanie aplikačných funkcií, automatickú správu dátových typov, validáciu dát. Pre výmenu dát sa v protokole SOAP využíva formát XML a pre komunikáciu protokol Http. (Lecky-Thompson, 2010)

SOAP dokáže predávať štrukturované dáta, čo umožnuje vyriešiť mnoho problémov s konverziou dát a následne integráciou do systému. Je to komunikačný protokol, ktorý nie je závislý na platforme, čo znamená, že je dostupný pre všetky platformi, ktoré sú schopné prevádzkovať XML parser. Pre PHP existuje niekoľko knižníc ako napríklad PEAR::SOAP, NuSOAP alebo rozšírenie SOAP pre PHP. Medzi hlavné triedy možeme zaradiť SoapClient a SoapServer. (Lecky-Thompson, 2010)

SoapClient

Prostredníctvom triedy SoapClient sa dokážeme pripojiť k webovej službe poskytovateľa a získavať informácie. Na pripojenie k webovej službe potrebujeme URI adresu dokumentu WSDL, ktorý definuje, ako používať danú službu.

SoapServer

Ak je potrebné aby systém posielal údaje prostredníctvom SOAP, je vytvorený SOAP Server. Podobne, ako pri získavaní informácií prostredníctvom SOAP klienta, sa aj SOAP server vytvára objektovo pomocou triedy *SoapServer*. Táto trieda ponúka metódy na pridávanie funkcií pre spracovanie požiadaviek SOAP a to *addFunction()*. Pre získanie zoznamu funkcií definovaných serverom sa používa funkcia *getFunction()*.

3.3.1 WSDL

Pri komunikácii prostredníctvom SOAP je nutné, aby bola opísaná štruktúrovane. Práve WSDL dokument rieši túto problematiku pomocou XML gramatiky. Používa sa pre opis sieťových služieb ako kolekciu komunikačných koncových bodov, ktorá umožní prenášať správy. (Lecky-Thompson, 2010)

Dokument WSDL obsahuje niekoľko hlavných komponentov:

- *Definitions* - je to koreňový element, ktorý definuje názov a menný priestor.
- *Types* - tento atribút nie je povinný. Má za úlohu definovať použité dátove typy.
- *Message* - správa reprezentujúca požiadavku. Každá sa skladá z niekoľkých logických častí.
- *PortType* - je množina abstraktných operácií. Vstupná a výstupná správa by mala byť obsahom každej operácie (metódy). Poznáme 4 typy operácií:
  - *Request-response*
  - *One-way* - obsahuje iba vstupnú správu.
  - *Notification* - obsahuje výstupnú správu.
  - *Solicit-response* - end point pošle správu a očakáva odpoveď.

- *Bindings/operation* - portType sa priradí ku konkrétnemu protokolu a formátu správy.
- *Service* - množina servisných portov.
- *Documentation*
- *Import*

3.4 REST

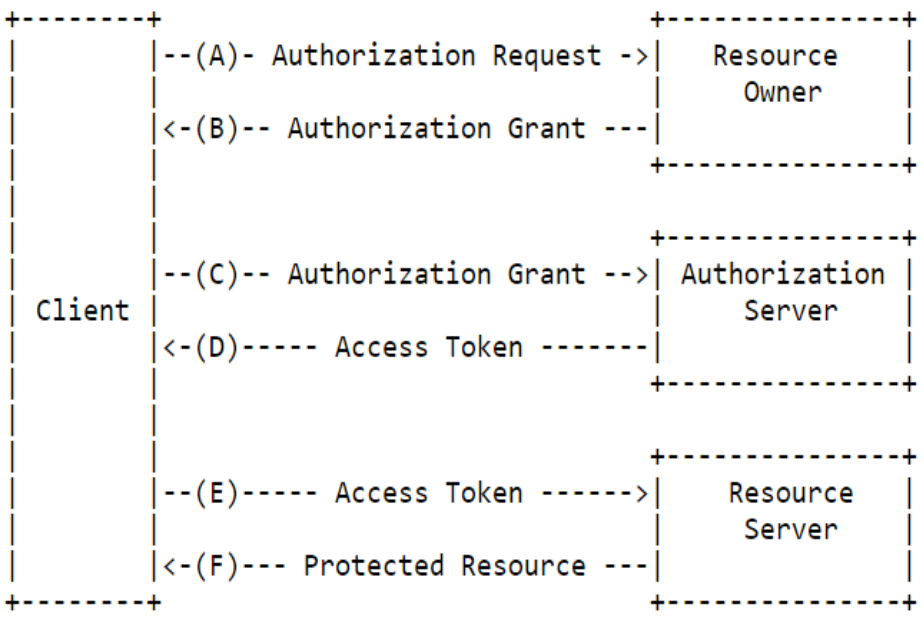
Roy Fielding ako prvý definoval iný pohľad na webové služby vo svojej dizertačnej práci v roku 2000. Narozdiel od protokolu SOAP, REST využíva HTTP protokol a jeho motódy GET, POST, PUT a DELETE. Tieto funkcie umožňujú prístup a manipuláciu s webovou službou pomocou URL adries. REST nepoužíva zložitú štruktúru v podobe WSDL dokumentu ani presný dátový formát odpovede. Na danej URL adrese môžeme nájsť obrázok, dokument HTML, XML alebo akýkoľvek iný formát. (Lecky-Thompson, 2010)

Pre Php existuje viacero rozšírení pre REST. Konkrétne pre framework Nette bol použitý balíček *drahak/Restful*. Toto rozšírenie obsahuje CRUD operácie:

- metóda POST *Presenter:create*,
- metóda GET *Presenter:read*,
- metóda PUT *Presenter:update*,
- metóda DELETE *Presenter:delete*. (Nette REST API, 2016)

Tento balíček obsahuje viacero možností ako zabezpečiť webové služby. Každá z nich sa využíva pre iný typ služieb a podľa ich potrieb. Sú to napríklad:

- *BasicAuthentication* - táto možnosť ponúka základné zabezpečenie, založené na autentizácií prihláseného používateľa.
- *SecuredAuthentication* - používateľovi sú posielané hashované dáta s privátnym kľúčom. (Nette REST API, 2016)
- *OAuth2* - protokol pre overovanie autentizácie a autorizácie. Tok dát je znázornený na obrázku 3. Klient najprv zadá požiadavku na autorizáciu, v obrázku bod A. Následne obdrží parameter grant type (bod B), podľa špecifikácie OAuth od vlastníka danej služby. Odpoveď, ktorú dostane od poskytovateľa služby, pošle na autorizačný server (bod C), ktorý mu vráti prístupový token (bod D). Prostredníctvom neho môže klient získavať zabezpečené dáta zo servera (bod E a F). (OAuth 2.0, 2012)



Obr. 3: Abstraktný tok dát (OAuth 2.0, 2012)

3.5 Analytické diagrami

Pri vytváraní projektov je nevyhnutné presne špecifikovať požiadavky a navrhnúť systém podľa štandardizovaného jazyka. Tomuto jazyku by mal rozumieť nielen celý tím programátorov, ale aj zákazníci. Na tieto účely sa využíva UML (Unified Modeling Language). Tento jazyk sa zaoberá primárne štandardizovanými diagramami, pričom každý z nich popisuje určitú oblasť návrhu softwaru. (Lecky-Thompson, 2010)

Pre účely diplomovej práce boli využité diagrami use case, statový, entitne-relačný, procesný diagram a diagram tried.

3.5.1 Entitne relačný diagram

ER diagram patrí medzi diagrami, ktoré využívajú top-down prístup k modelovaniu dát. Tento diagram je dôležitý z pohľadu dát, ktoré sú vytvorené, uložené a používane systémom. Dáta sú organizované a vzťahy medzi nimi definujú väzby. (Langer, 2008)

ERD obsahuje tri prvky:

- entita - znázorňuje objekt v reálnom svete, ktorý má určité vlastnosti,
- atribút - vlastnosti danej entity,
- vzťah - udáva vzťahy medzi entitami. (Lesson 1: Systematically Approaching Design Stages, 2016)

3.5.2 Use case

Pohľad používateľa na daný systém znázorňuje diagram prípadu použitia. Prípad použitia definuje akcie, ktoré môže používateľ vykonať. (Lecky-Thompson, 2010)

Diagram prípadu použitia sa skladá z troch prvkov:

- aktér - používateľ alebo externý systém, ktorý môže vykonať určité akcie,
- prípad použitia - akcia, alebo funkcionalita, ktorú môže daný aktér vykonať,
- väzba - spojenie aktéra s prípadom použitia, alebo prepojenie prípadov použitia. (Lecky-Thompson, 2010)

3.5.3 Stavový diagram

Stavové diagrami znázorňujú aspekty dynamického chovania systému. Využívajú pri znázornení systému zmeny stavu počas celého životného cyklu daného objektu. Tento diagram má tri základné prvky:

- stav - môžeme definovať ako situáciu, na ktorú daný objekt reaguje určitou aktivitou, čaká na určitú udalosť, ktorá nastane alebo spĺňa určitú podmienku v danom čase,
- udalosť - výskyt niečoho v čase alebo priestore,
- prechod - zmena stavu z jedného do druhého, pričom je vyvolaný určitou udalosťou. (Arlow, 2007)

3.5.4 Sekvenčný diagram

Sekvenčné diagrami sú často spojené s konkrétnym prípadom použitia, kde sa zachytáva dynamické správanie systému. Bližšie špecifikuje interakciu medzi objektami a zasielanie správ medzi nimi.

Vlastnosti sekvenčného diagramu:

- komunikácia medzi objektami prostredníctvom zasielania správ,
- popisuje priechod správ systémom,
- neobsahuje presné výrazové prostriedky pre vetvenie a podmienky.

3.6 Verzovacie systémy

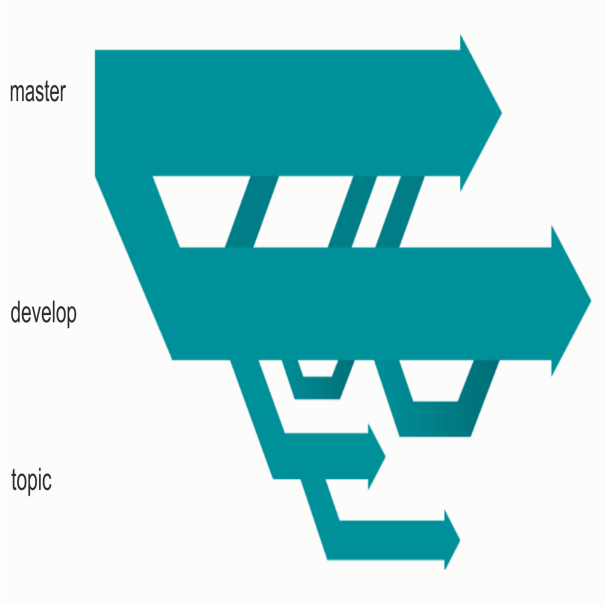
Pri tvorbe každého systému vzniká mnoho problémov, či už pri písaní kódu, jeho spravovaní alebo obnovení predchádzajúcej verzie. Problémy nastávajú obzvlášť pri vytváraní väčšieho systému, kde sa na tvorbe kódu podieľa niekoľko programátorov, návrhárov alebo dokonca samotný klient. Tomuto sa snažia predísť verzovacie systémy, ktorých úlohou je:

- snaha zabránenia konfliktom medzi viacerými programátormi, ktorí spravujú ten istý kód,
- zaznamenávanie zmien v kóde, čo umožňuje vrátiť sa na predchádzajúcu verziu,
- uchovávanie kódov v sieťovom repozitári, odkiaľ je možné ich kedykoľvek stiahnúť.

Git

Git je open source systém pre správu verzií, ktorý je navrhnutý tak, aby zvládal spravovanie ako malých, tak aj rozsiahlých projektov. Existuje rôzny verzovací software, ktorý je možné využívať, ako napríklad Microsoft Visual SourceSafe, SourceTree, CVS. (About Git, 2016)

Pri vytvaraní systému viacerými programátormi môžu vznikať kolízie, ktorým sa verzovacie systémy snažia predísť. Pri vytvárani nového projektu sa štandardne inicializuje *Gitflow*, kde sa vytvoria vetvy *master* a *develop* (viď obr. 4). Vo vetve *master* je ustálená verzia systému, ktorá je nasadená na produkčnej verzii servera. Vetva *develop* slúži ako pracovná verzia, z ktorej vychádzajú jednotlivé vetvy vývojárov. Každý programátor si pre novú funkcionalitu môže vytvoriť vetvu, kde daný kód spravuje. Po dosiahnutí finálnej verzie funkcionality zlúči zmeny s vetvou *develop*. Ukončenie funkcionality vo vetve *develop* a jej otestovanie umožňuje zlúčenie s verziou *master*. (About Git, 2016)



Obr. 4: Vetvenie v Git (About Git, 2016)

3.7 Použité programovacie jazyky

Medzi hlavnými požiadavkami zadávateľa projektu bolo vytvorenie systému ako webovej aplikácie. Pre implementáciu si zvolil jazyk PHP a framework Nette. Práve tieto technológie budú opísané v nasledujúcej podkapitole.

3.7.1 PHP

Jazyk PHP patrí medzi najobľúbenejšie skriptovacie jazyky pri vývoji webových stránok. Bol vytvorený v roku 1995 Rasmusom Lerdorfom. Čo sa týka syntaxe jazyka, je ovplyvnený viacerými programovacími jazykmi, ako napríklad Perl, C a ASP. Uvedenie aplikácie PHP nevyžaduje zložité kroky a zdĺhavú konfiguráciu servera. Pre beh stačí webový server a modul PHP. Novšie verzie jazyka PHP sa snažia odstrániť nedostatky z predchádzajúcich verzií. Častým terčom kritiky bola nedostatočná podpora objektovo orientovaného prístupu, čo sa vyriešilo vo verzii PHP 5. Taktiež absencia menných priestorov bola odstránená vo verzii PHP 5.3. (Lecky-Thompson, 2010)

Framework Nette

Framework môžeme považovať za vopred pripravený kód, kolekcie tried. Aj napriek tomu, že jazyk PHP obsahuje veľa užitočných funkcií, často sa stáva, že pri programovaní v čistom PHP v každom projekte je nutné programovať základnú funkcionalitu. Práve tento prípad riešia frameworky, ktoré obsahujú rôzne knižnice, pomocou ktorých sa nová aplikácia vytvorí rýchlejšie a jednoduchšie. Viaceré frameworky sa snažia vytvoriť základnú štruktúru aplikácie, ktorú si programátor môže do určitej miery zmeniť. To obzvlášť platí pre frameworky, ktoré vychádzajú z návrhového vzoru MVC, čo znamená oddelenie kódu od zobrazovacej logiky. (Lecky-Thompson, 2010)

Pre diplomovú prácu bol zvolený framework Nette. Tento framework bol vybratý firmou it2b s.r.o., ktorá považovala framework Nette ideálny pre vývoj helpdesku. Nette ponúka jednoduché vytváranie formulárov, zobrazovanie stránok pomocou šablonovacieho systému Latte. Taktiež ponúka neustále rozrastajúcu sa ponuku doplnkov, ktoré je možné využiť v aplikáciách.

3.7.2 HTML

HTML je značkovací jazyk pre tvorbu webových stránok. HTML dokument pozostáva zo stromu elementov a textu (viď obr. 5). Každý element je definovaný pomocou značiek. Element môže obsahovať atribúty, ktoré určujú, ako dané prvky fungujú. (Introduction, 2016)

Jazyk HTML prešiel od začiatku vzniku mnohými zmenami a bolo vytvorených viacero verzií. Prvá bola uverejnená v roku 1993 a odvtedy bolo uverejnených päť ďalších verzií, pričom posledná v roku 2014 pod označením HTML5. HTML5 je

postavené na rôznych princípoch, ktoré majú priniesť novú víziu možností a praktickosti. Táto vízia stavia na pojmoch kompatibilita, prínos, interoperabilita a obecný prístup. Taktiež v tejto verzii je obmedzené používanie zásuvných modulov, vďaka podpore viacerých funkcií, ktoré kedysi boli možné len s modulmi. (Lubbers, 2011)

Dôvodom, prečo prejsť na HTML5 je skutočnosť, že je prispôsobené na všetky prehliadače. Rovnako sa kladie dôraz na podporu starších verzií, keďže veľa webov nie je pripravených na takúto zmenu a v poslednej verzii došlo k zavrhnutiu niektorých značiek. Avšak pribudlo aj niekoľko nových, ako napríklad:

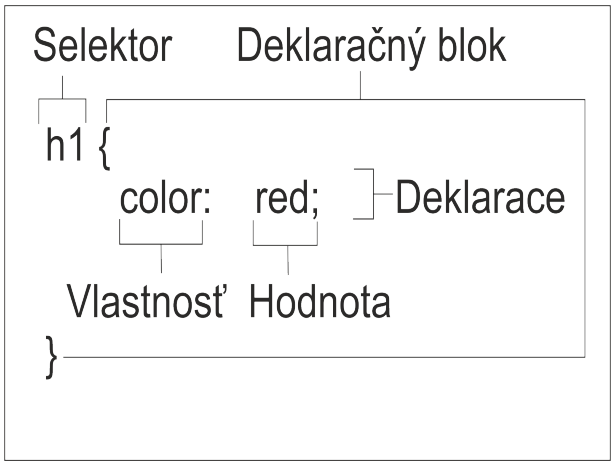
- štruktúra dokumentu - section, article, header, footer,
- mediálne značky - canvas, video, audio. (Lubbers, 2011)

3.7.3 CSS

Kaskadové štýly (CSS, Cascading Style Sheet) sú pre moderné webové stránky veľmi dôležité. Dobrý grafický návrh pomôže používateľovi pomôcť k lepšej orientácií na stránke. Príjemný a moderný vzhľad môže taktiež upútať používateľa, a práve preto sa na daný web vráti. Je veľa dôvodov, prečo sa v súčasnosti kladie veľký dôraz na grafický design. Práve kvôli tomu je programátorom k dispozícií množstvo šablón, ktoré riešia čo najlepšie a najefektívnejšie rozloženie prvkov a grafický design pre konkrétny web.

Šablóna je vlastne sada pravidiel, ako sa majú zobraziť jednotlivé elementy. Každé pravidlo sa skladá z dvoch hlavných častí (viď obr. 5):

- selektor - určuje, na ktoré elementy sa dané pravidlo má uplatniť,
- deklaračný blok - skladá sa z dvojíc vlastnosť - hodnota, čo tvorí deklaračný blok. Tento blok presne špecifikuje, čo by sa s vybraným elementom, ktorý je vybratý v selektore malo stať. (Lubbers, 2011)



Obr. 5: Pravidla štýlu (Lubbers, 2011)

4 Súčasný stav

Helpdesk alebo technickú podporu môžeme vnímať ako oddelenie vo vnútri organizácie, ktoré je zodpovedné za zodpovedanie technických dotazov svojich používateľov. Štandardne helpdesk ponúka jediné kontaktné miesto, ktoré je ušami a očami organizácie. Úroveň Helpdesku vo veľkej miere ovplyvňuje názor o úrovni poskytovateľa služby. (HILES, 2016)

Projektovo orientovaná metodika riadenia IT služieb ITIL (Information Technology Infrastructure Library), preferuje názov "Single point of contact"(SPOC), a teda jednotný kontaktný bod pre všetky požiadavky. (Bróska, 2009) Metodika riadenia IT služieb ITIL je súbor knižných pubikácií, obsahujúcich zbierku najlepších skúsenosti z odboru riadenia služieb informačných technológií. Britská spoločnosť AXELOS, Ltd., ktorá je vlastníkom týchto publikácií, dohliada aj na ďalšie celosvetovo používané rámce a metodiky. (Čo je to ITIL, 2016)

ITIL v aktuálnej verzii obsahuje základné publikácie:

- *Service Strategy*,
- *Service Design*,
- *Service Transition*,
- *Service Operation*,
- *Continual Service Improvement*. (Čo je to ITIL, 2016)

Medzi publikácie Service Operation patrí služba Service desk alebo SPOC. Základné povinnosti SPOC:

- vytvoriť jednotný kontaktný bod, ktorý bude k dispozícií pre používateľov služieb,
- obnovenie služby v prípade výpadku,
- všetky požiadavky musia byť spracované,
- komunikácia so zákazníkom.(Service desk, 2016)

Vytvoriť správny helpdesk pre spoločnosť je náročné. Na trhu je mnoho možností. Či už spoločnosť využije metodiku ITIL, alebo akýkoľvek systém vyskytujúci sa na trhu, je nutné vykonať analýzu, čo helpdesk systému musí obsahovať. V nasledujúcich podkapitolách bude opísaná spoločnosť it2b s.r.o., ktorá je poskytovateľom informačného systému QI, samotný systém QI, požiadavky spoločnosti a základné funkcie, ktoré služba helpdesk musí obsahovať, ale aj systémy, ktoré sa nachádzajú na trhu.

4.1 O spoločnosti it2b s.r.o.

Spoločnosť vznikla pod názvom Orlax, s.r.o. v roku 2004 a hlavným cieľom bolo poskytovanie outsourcingových služieb zahrňujúcich informačné a komunikačné technológie. Spoločnosť zmenila názov na it2b s.r.o. v roku 2008 a začala sa zameriavať na individuálny prístup a starostlivosť o jednotlivých zákazníkov. (O společnosti it2b s. r. o., 2016)

It2b s.r.o. ponúka svojím zákazníkom viacero služieb:

- webhosting,
- tvorbu webových informačných systémov,
- tvorba webových prezentácií a aplikácií,
- webdesign,
- externú správu počítačových sietí, informačných systémov a technológií. (O společnosti it2b s. r. o., 2016)

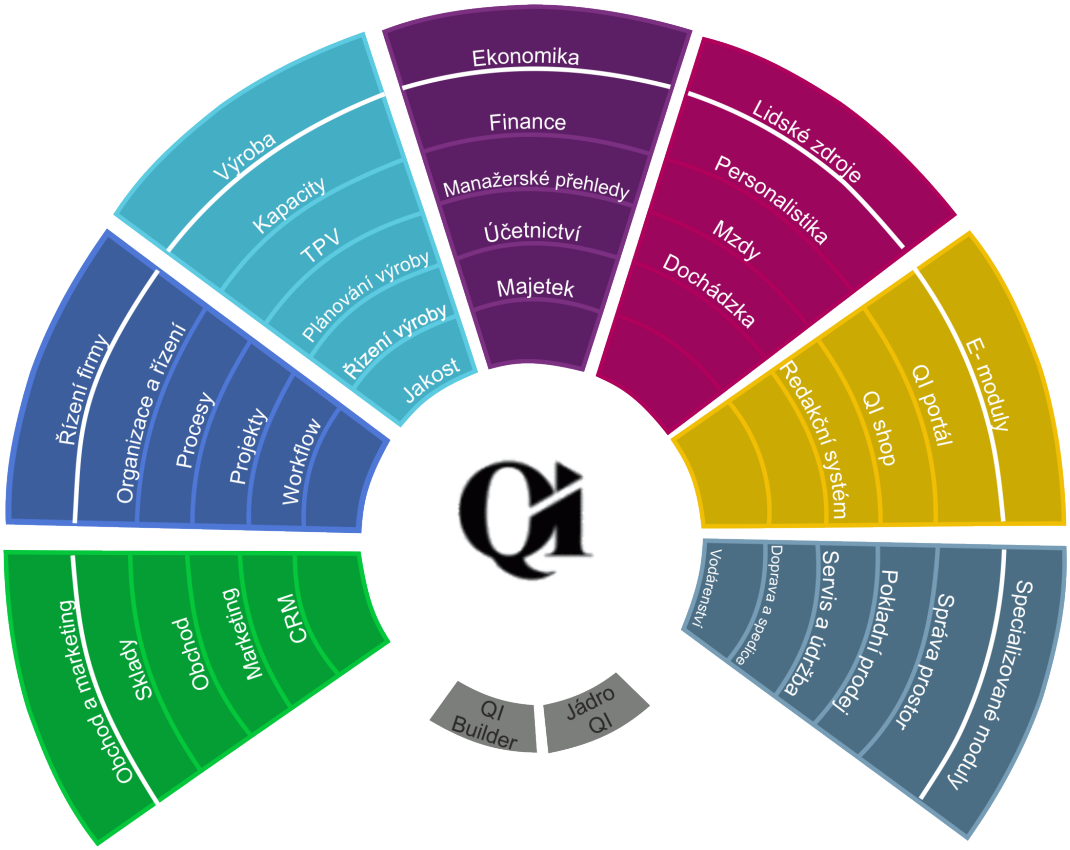
4.1.1 Informačný systém QI

Spoločnosť it2b s.r.o. je dodávateľom prvého elastického informačného systému QI. QI je komplexný informačný systém, ktorý spĺňa požiadavky na riadenie firmy a spracovanie podnikových inormácií. Systém QI sa dokáže čo najviac prispôsobiť zákazníkom, preto je využívaný vo veľkých, stredných ale aj malých firmách a u fyzických osôb. (Informační systém QI, 2016)

Informačný systém QI ponúka:

- ľahké ovládanie a technologickú vyspelosť,
- bezpečnosť dát,
- prispôsobenie požiadavkám zákazníka,
- možnosť prepojenia na ďalší software,
- referencie spokojných zákazníkov.

Systém QI obsahuje 7 skupín (modulov) (viď obr.6), ktoré združujú viac než 300 obchodných jednotiek. Každá firma si vyberá to, čo potrebuje a tak si môže vytvoriť optimálny systém. (Informační systém QI, 2016)

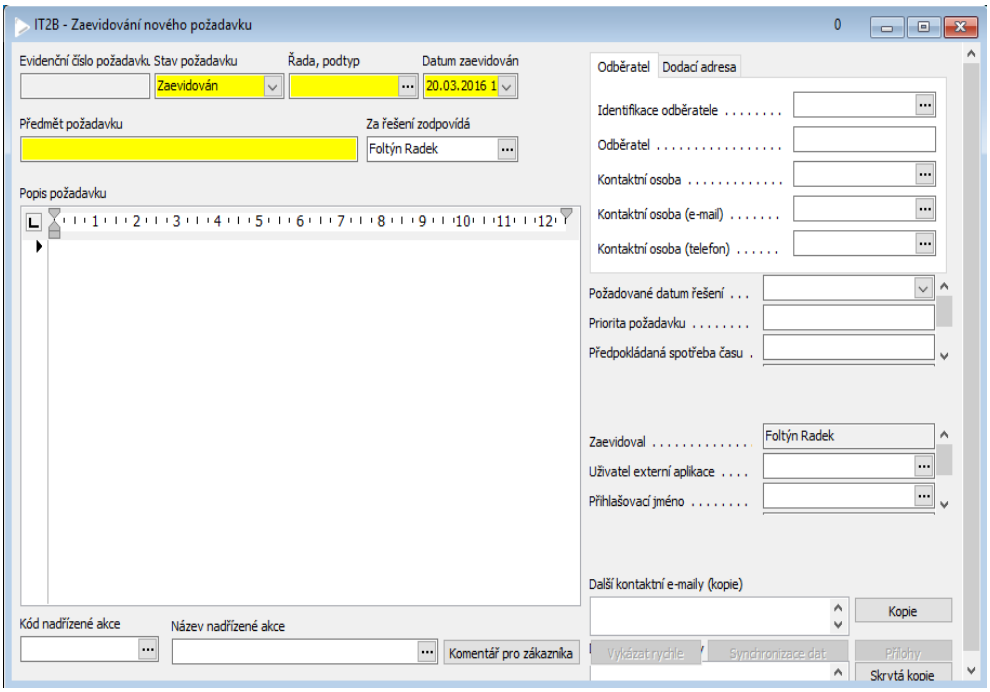


Obr. 6: Prehľad modulov QI (Informační systém QI, 2016)

4.2 QI helpdesk

V súčasnosti nemá informačný systém QI jednotný kontaktný bod na zadávanie požiadaviek od zákazníkov smerom k poskytovateľovi informačného systému. Požiadavky od zákazníkov sú zaznamenávané v systéme QI, ale zadávajú ich zamestnanci podpory.

Ak zákazník chce pridať novú požiadavku, napíše všeobecný email, ktorý sa rozpošle všetkým konzultantom. Konzultanti na základe support plánu (plán rozdelenia projektov medzi jednotlivých konzultantov), vidia email od svojho zákazníka a následne ho zaevidujú. Konzultant zadáva požiadavku do systému pomocou formulára (viď obr. 7).

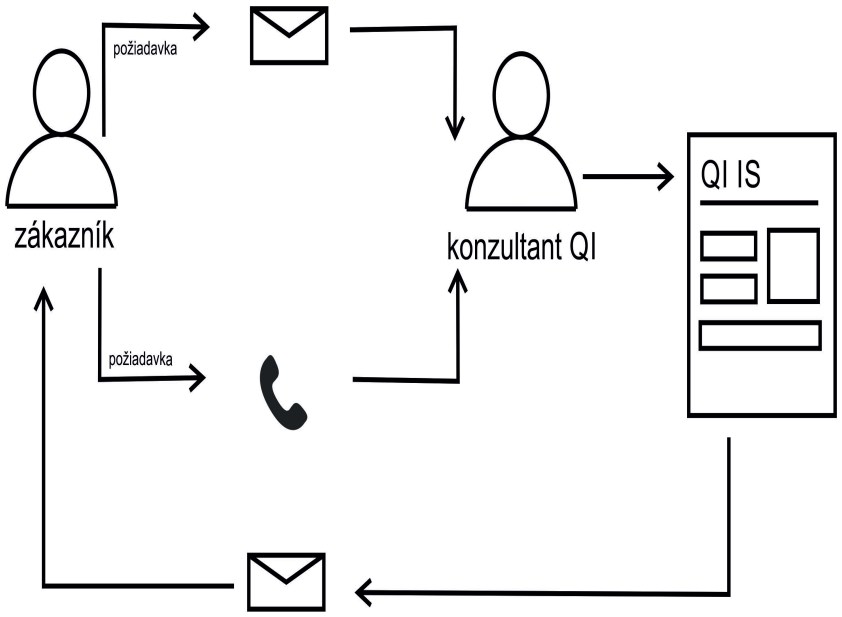


Obr. 7: Formulár zadávania požiadaviek v QI

Ak je požiadavka správne vyplnená, je uložená do stavu zaevidovaná. Každá požiadavka má prideleného riešiteľa, ktorý po dokončení zmení stav požiadavky na vyriešený. Ak požiadavka zmení stav, zákazník, ktorý je uvedený ako kontaktná osoba, je informovaný emailom o vyriešení požiadavky.

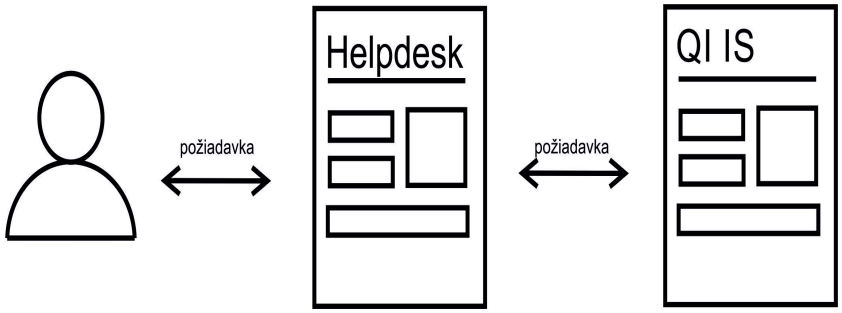
Požiadavka obsahuje aj komunikáciu so zákazníkom. Táto komunikácia je skopírovaná z emailu. Do systému sa vkladá vždy s časovým razítkom, podľa času, kedy bola zadaná. Problém nastáva pri telefonickom zadávaní požiadavky.

Súčasný tok dát medzi zákazníkom a informačným systémom QI je znázornený na obrázku 8. Hlavnou nevýhodou súčasného toku dát je posielanie požiadaviek od zákazníka do QI prostredníctvom emailu alebo telefónu a následne konzultantovi QI, ktorý danú požiadavku zadá do informačného systému QI.



Obr. 8: Súčasný tok dát pri zadávaní požiadaviek

Nové riešenie helpdesku má za úlohu znížiť čas a náročnosť pridávania požiadaviek. Navrhnutý tok dát s využitím webového helpdesku je znázornený na obrázku číslo 9. Zákazník pridáva novú požiadavku do webového helpdesku, ktorý následne synchronizuje údaje s informačným systémom QI. Hlavnou výhodou tohto prenosu je skrátenie času zadávania, schvaľovania ale aj riešenia požiadavky.



Obr. 9: Tok dát pri zadávani požiadaviek v helpdesku

4.3 Základné funkcie helpdesk

V súčasnosti základnou funkciou helpdesku je vyriešiť problém zo strany zákazníka s informačným systémom QI. V systéme je možné danú požiadavku:

- Vytvoriť - konzultant QI vytvorí požiadavku na základe emailovej alebo telefonickej komunikácie od zákazníka.

- Editovať - sytém umožňuje takú úpravu požiadavky, ktorá ju bližšie špecifikuje. Nie je možné zmeniť jej hlavnú podstatu. Ak by bola zmena z hľadiska funkcionality výrazná, je nutné vytvoriť novú požiadavku.
- Prideliť odberateľa (zákazníka) s kontaktnými údajmi, dodacou adresou.
- Prideliť osobu zodpovednú za vyriešenie.
- Zmeniť stav.
- Zmeniť radu, podtyp.
- Zadať požadovaný dátum riešenia.
- Zadať/zmeniť prioritu požiadavky - priorita požiadavky sa určuje po konzultácií so zákazníkom.
- Zadať predpokladanú spotrebu času pri riešení požiadavky.
- Zadať ďalšie kontaktné emaily - zmena stavu sa oznámi zákazníkovi uvedenému v kontaktných emailoch, prípadne v položke kontaktná osoba prostredníctvom emailu.
- Zadať popis požiadavky a komentár pre zákazníka.
- Pridávať prílohy - systém umožňuje zákazníkovi aj konzultantovi pridávať prílohy ako bližšiu špecifikáciu požiadaviek.

4.4 Existujúce systémy

Na trhu existuje množstvo systémov zameraných na oblasť, ktorá rieši požiadavky od zákazníkov. Každá spoločnosť má iné kritéria, preto je tažké nájsť to optimálne riešenie pre danú spoločnosť. Pri výbere je nutné zvážiť množstvo kritérií ako napríklad:

- koľko ľudí bude využívať helpdesk,
- aké sú možnosti nadefinovania krokov jednotlivých požiadaviek,
- možnosť prispôsobenia na požiadavky firmy,
- inštalácia helpdesku na vlastný server.

4.4.1 Informačný systém QI

Ako už bolo vyššie opísané, informačný systém QI v súčasnosti disponuje evidovaním a spracovávaním požiadaviek.

Systém vyhovuje zadaniu:

- zodpovedá procesnému nastaveniu systému QI.

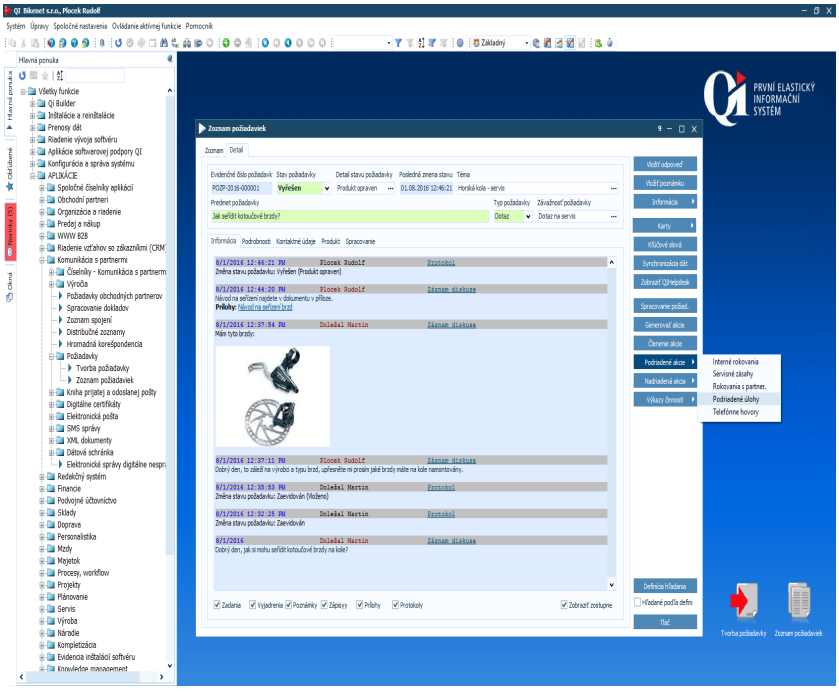
Systém nevyhovuje zadaniu:

- zákazník zadáva požiadavky prostredníctvom emailu, alebo telefonátu, ktoré následne musí do systému QI zadať konzultant,
- používateľská prívetivosť,
- množstvo času, ktorý konzultanti musia vynaložiť na zadávanie požiadaviek,
- komunikácia medzi zákazníkom a konzultantom nie je úplne zdokumentovaná.

4.4.2 DC Concept

Ďalším systémom, ktorý spoločnosť s.r.o. na spracovávanie požiadaviek od používateľov využívala, bol QI Helpdesk od spoločnosti DC Concept (viď obr. 10).

Tento helpdesk umožňuje všetky otázky, nové zakázky, opravy, reklamácie a rôzne požiadavky dostať ihneď k zodpovedným osobám. Zákazník má vždy aktuálny prehľad o stave svojej požiadavky. Taktiež umožňuje ukladať ich históriu, preto je možné vyhodnotiť činnosť technickej podpory. Vďaka helpdesku je možné zistiť spätnú väzbu od zákazníkov, a zistiť aké požiadavky sa najčastejšie opakujú. (QI Helpdesk, 2016)



Obr. 10: Zaevidovaná požiadavka QI Helpdesk DC Concept

Narozdiel od systému, ktorý súčasne využíva it2b s.r.o., v tomto helpdesku je možný priamy zápis požiadaviek od používateľov do QI systému. Nie je však možné ho napojiť na paušály, zmluvy, evidovať viac firiem pre jeden účel. Úpravy, ktoré by

bolo nutné vykonať v tomto systéme, by boli pracné a vyžadovali by ďalšie zložité úpravy.

Systém vyhovuje zadaniu:

- automatické zadávanie požiadaviek do systému,
- systém vychádza z QI.

Systém nevyhovuje zadaniu:

- neumožňuje napojenie na paušály, zmluvy,
- evidencia viacerých firiem na jeden účet,
- nie je možnosť schvaľovania požiadaviek,
- nedajú sa pridávať prílohy,
- systém sa zložito generuje, aby mohol byť editovateľný z QI (vzhľad), sú nutné veľké úpravy.

4.4.3 Taskpool

Na českom trhu sa nachádza aj helpdesk Taskpool, ktorý ponúka zákaznícky helpdesk, vnútrofiremný helpdesk, firemné úlohy, helpdesk pre mestá a obce. Tento systém je založený na základných princípoch ITIL (viď kapitola 4). Taskpool sa skladá z modulov, kde základný je samotný helpdeskový systém, ďalšími sú napríklad EDM (External database manager), SMS notifikácie.

Zákanícky helpdesk

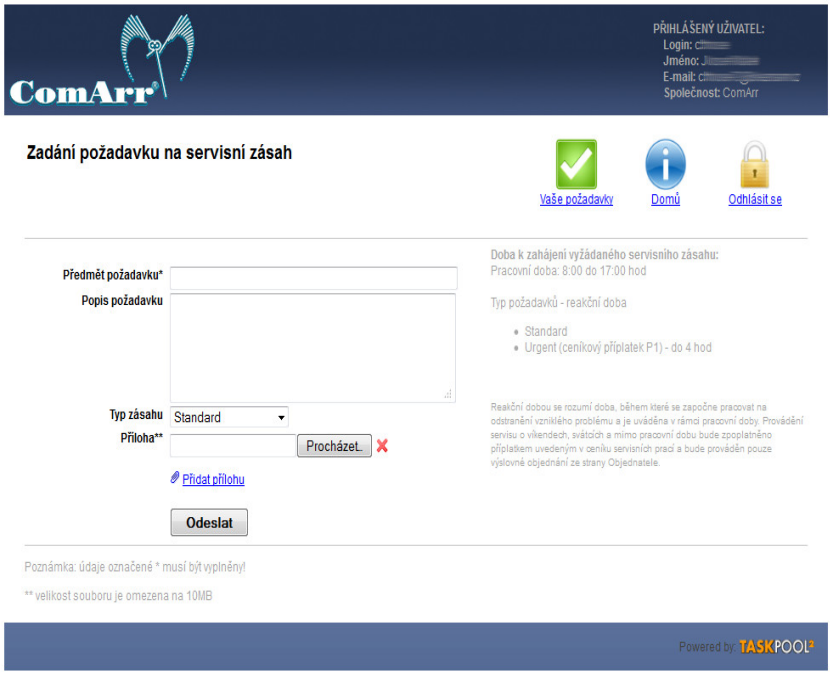
Všetky požiadavky má zákazník prístupné na jednej obrazovke, kde vidí, v akom sú stave a môže pridávať komentáre alebo otázky. Systém Taskpool uchováva všetky historické požiadavky. Zákaznícky helpdesk umožňuje definovať pre každú oblasť premenné hodnoty. Tieto dynamické polia slúžia k lepšej identifikácií požiadaviek.

Systém vyhovuje zadaniu

- zákazník môže zadávať požiadavky pomocou webovej aplickácie (viď obr. 11),
- jednoduchý a intuitívny systém,
- história požiadaviek,
- dynamická úprava požiadaviek.

Systém nevyhovuje zadaniu

- napojenie systému na QI,
- neumožňuje napájanie na zmluvy,
- evidencia viacerých firiem na jeden účet.



Obr. 11: Taskpool - zadávanie požiadaviek

Pri Taskpool systéme podobne ako pri ostatných je problém s napojením na systém QI. Inštalačný balíček ponúka niekoľko verzií, a to:

- zákazník si nastaví helpdesk sám,
- základný inštalačný balíček a helpdesk na mieru.

Systém na mieru je niekoľko krát drahší ako základný balíček. Náklady na využívanie tohto helpdesku by rástli s ďalšími požiadavkami na prispôsobenie Taskpoolu pre systém QI. (Taskpool, 2016)

5 Návrh

V tejto kapitole sú stanovené funkčné a nefunkčné požiadavky, ktoré určujú, čo je úlohou vyvíjanej aplikácie. Tiež sú navrhnuté UML diagramy a grafický návrh softwaru.

5.1 Požiadavky

Vývoj softwaru vyžaduje požiadavky, ktoré určujú hlavnú funkcionalitu systému. Ich úlohou nie je určiť kvalitu aplikácie, len potreby a podmienky nového produktu. Požiadavky často nie sú presne stanovené a počas vývoja sa menia. Avšak čo najlepšie stanovenie výrazne pomôže pri implementácií. (Rochkind, 2013)

Požiadavky sú definované zadávateľom projektu, spoločnosťou it2b s.r.o. Hlavnou podmienkou bolo vytvoriť webovú aplikáciu, ktorá bude dostupná pre všetkých zákazníkov. Podstatou aplikácie je zadávanie požiadaviek od zákazníkov a taktiež zobrazovanie celého životného cyklu požiadavky. Podstatným prvkom je aj synchronizácia údajov z informačného systému QI.

5.1.1 Funkčné požiadavky

- Získavanie údajov - údaje pre Helpdesk sa budú pravidelne získavať z informačného systému QI a následne sa budú ukladať do databázy. Zo systému QI sa budú automaticky sťahovať údaje o:
  - používateľoch - systém bude obsahovať databázu používateľov, ktorá bude disponovať údajmi ako meno, prihlasovacie údaje. Prihlásiť sa do systému môže iba používateľ, ktorý bude evidovaný v tejto databáze. Používateľ nemôže meniť osobné údaje, okrem prihlasovacieho hesla. Po zmene hesla sa údaje ihneď odošlú na synchronizáciu do systému QI.
  - právach používateľov - každý používateľ má definované do akej role patrí, čo bude zahŕňať aj obmedzenia a práva v helpdesku. Obmedzenia a práva budú určovať viditeľnosť jednotlivých požiadaviek a taktiež to, či môže schvaľovať požiadavky v danom obchodnom prípade.
  - firmách - databáza všetkých firiem, ktorých zamestnanci majú prístup do helpdesku QI. Ku každej firme sa budú evidovať údaje ako názov firmy, adresa, IČO, DIČ. Každá firma má obchodné prípady, ku ktorým evidujeme názov, taktiež sa budú evidovať informácie o paušáloch a prečerpaných hodinách, kedy začína a končí paušálne obdobie.
  - požiadavkách - požiadavky sa budú evidovať pre každú firmu a konkrétny obchodný prípad. Pri požiadavkách sú dôležité údaje o názve, popise, rovnako aj o stave, v ktorom sa požiadavka nachádza. Pri vytvorení požia-

davky sa dáta synchronizujú ihneď po odoslaní formulára. Zo systému QI sa vráti odpoveď s pridelenými identifikačnými údajmi systému QI.

  - stavoch, fázach, druhu, priorite.

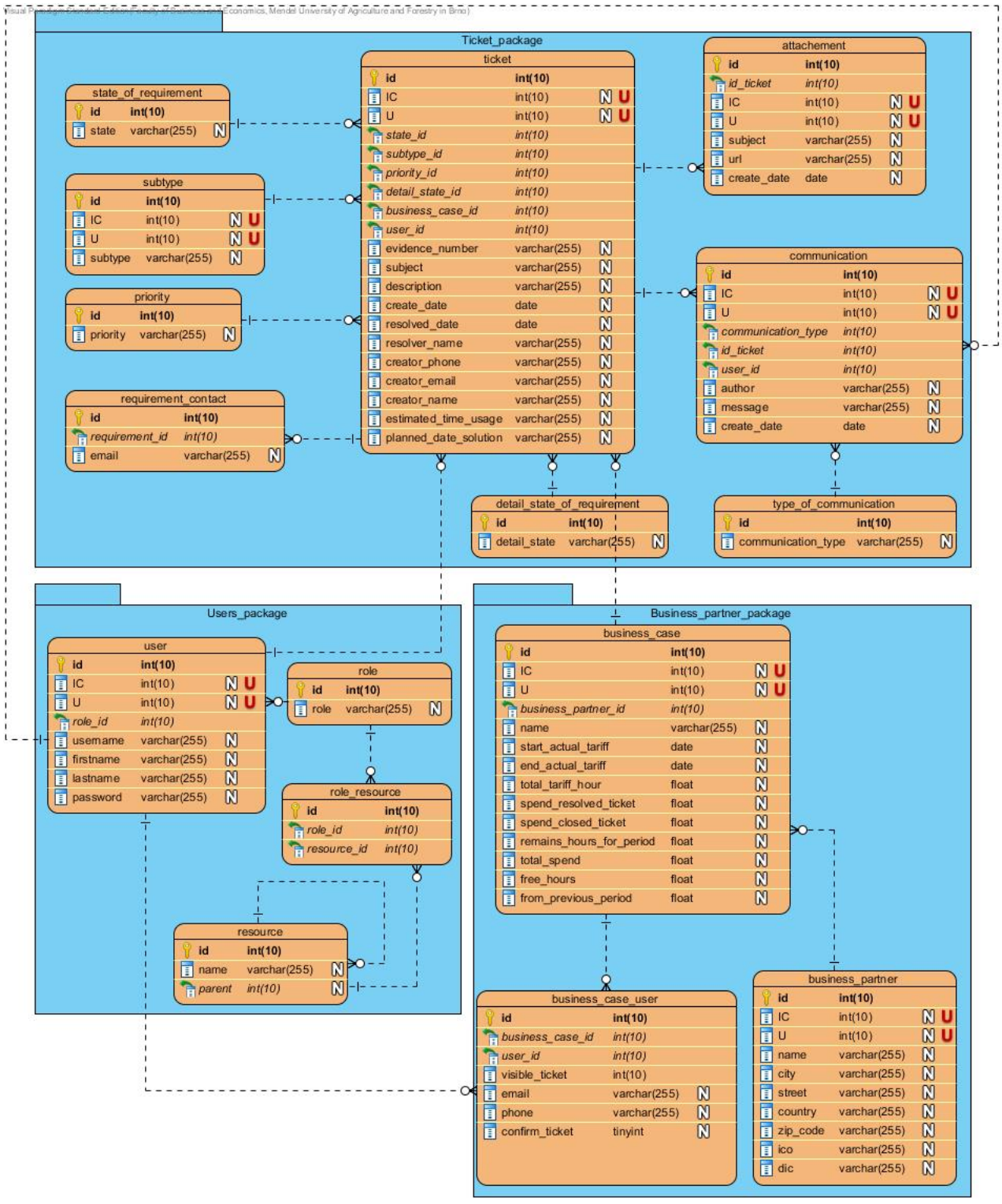
- Prihlásený používateľ môže mať priradených viacero firiem. V rámci každej firmy, podľa stanovených práv, môže schvaľovať alebo zobraziť požiadavky daného obchodného prípadu.
- V helpdesku je možné zobraziť všetky požiadavky používateľa, ktorý je prihlásený. K jednotlivým požiadavkám pridať komentár na upresnenie zadania. Komentáre môžu byť rôzneho typu, kde pri každom sa zobrazí dátum a meno používateľa, ktorý pridal komentár.
- Všetká komunikácia medzi zadávateľom a riešiteľom bude uchovaná vo forme komentárov v požiadavke. Uchovávať sa budú aj zmeny stavu, detailu stavu.
- Používateľ môže k jednotlivým požiadavkam pridať ľubovoľný počet príloh a taktiež kontaktné emaily na používateľov, ktorí majú byť informovaní o zmenách riešenia zadaného problému.

5.1.2 Nefunkčné požiadavky

- Implementácia - systém bude implementovaný v jazyku PHP vo frameworku Nette, pričom bude využívať databázu MySQL.
- Synchronizácia údajov - pri posielaní údajov z informačného systému QI do helpdesku sa bude využívať protokol SOAP.
- Design - helpdesk musí byť jednoduchý, prehľadný a intuitívny. Používateľ musí mať jasne stanovené prvky pre základnú funkcionalitu. Helpdesk bude mať responzívny design.
- Bude zaistená kompatibilita údajov s informačným systémom a taktiež zabezpečený prístup do helpdesku.

5.2 Dátový model

Databázová štruktúra, ktorá bola vytvorená pre helpdesk obsahuje 16 tabuliek (viď obr. 12). Každá tabuľka obsahuje atribút id, ktorý slúži ako primárny kľúč pre daný záznam v tabuľke. Atribúty IC a U sa používajú ako identifikátori v informačnom systéme QI, kde každý záznam je identifikovaný podľa týchto dvoch hodnôt. Súčasťou databázy sú aj tabuľky pre funkcionalitu OAuth2, ktoré v diagrame nie sú znázornené. Dátová štruktúra bola vytvorená na základe zadávaciej dokumentácie, konkrétne štruktúry dátových rezov od prevádzkovateľa informačného systému QI (viď príloha B).



Obr. 12: Dátový model helpdesku

V nasledujúcej časti sú opísané jednotlivé tabuľky a ich atribúty:

- *Ticket* - zoznam všetkých požiadaviek, ktoré boli zadané používateľmi. Každá požiadavka je definovaná identifikátorom helpdesku (atribút *id*) a identifikátormi informačného systému QI (*IC* a *U*). Atribúty *IC* a *U* môžu byť prázdne, ak používateľ vytvoril požiadavku v helpdesku a ešte jej neboli pridelené identifikačné údaje zo systému QI. Ďalšie povinné údaje, ktoré sa evidujú pri požiadavkách sú *subject*, *description*, bez ktorých nie je možné vytvoriť danú požiadavku. Pri vytvorení sa automaticky vyplnia údaje *user id, create_date, creator_email, creator_name*. Tieto informácie sa vzťahujú k používateľovi, ktorý danú požiadavku vytvoril.
- *State of requirement* - tabuľka obsahuje číselník stavov, ktoré môžu nastať počas životného cyklu požiadavky.
- *Subtype* - číselník podtypov požiadaviek, ktoré sa evidujú v systéme QI.
- *Priority* - číselník priorít, ktoré môžu nadobúdať požiadavky.
- *Requirement contact* - pri každej požiadavke môžu byť evidované ďalšie kontakty, na ktoré sa budú zasielať zo systému QI emaily.
- *Attachement* - prílohy môžu byť súčasťou každej požiadavky. Pri zadaní prílohy z helpdesku, budú údaje IC a U prázdne, kým sa im po synchronizácií nepridelia z QI.
- *Communication* - tabuľka communication obsahuje komentáre pridané používateľmi k určitej požiadavke. Pri zadávaní komentára musí prihlásený používateľ vybrať, o aký typ komentára ide. Automaticky sa vyplnia údaje o používateľovi *user_id* a *author*.
- *Detail state of requirement* - niektoré zo stavov sa musia bližšie špecifikovať podstavmi aby bolo jednoznačné, ako postupuje práca na danej požiadavke. Tieto podstavy sa ako číselník evidujú v tabuľke *detail_state_of_requirement*.
- *Type of communication* - číselník typov komentárov, ktoré je možné v helpdesku vytvoriť.
- *User* - zoznam používateľov, ktorý majú prístup do helpdesku QI. Základné údaje sa do helpdesku posielajú zo systému QI.
- *Role* - číselník používateľských rolí, ktoré sa vyskytujú v helpdesku.
- *Resource* - zoznam jednotlivých akcií, ktoré sú v aplikácií helpdesk.
- *Role resource* - táto tabuľka slúži na definovanie oprávnení v helpdesku. Je vyžadovaná použitými technológiami Nette.
- *Business partner* - zoznam firiem, u ktorých definujeme názov a údaje o adrese.

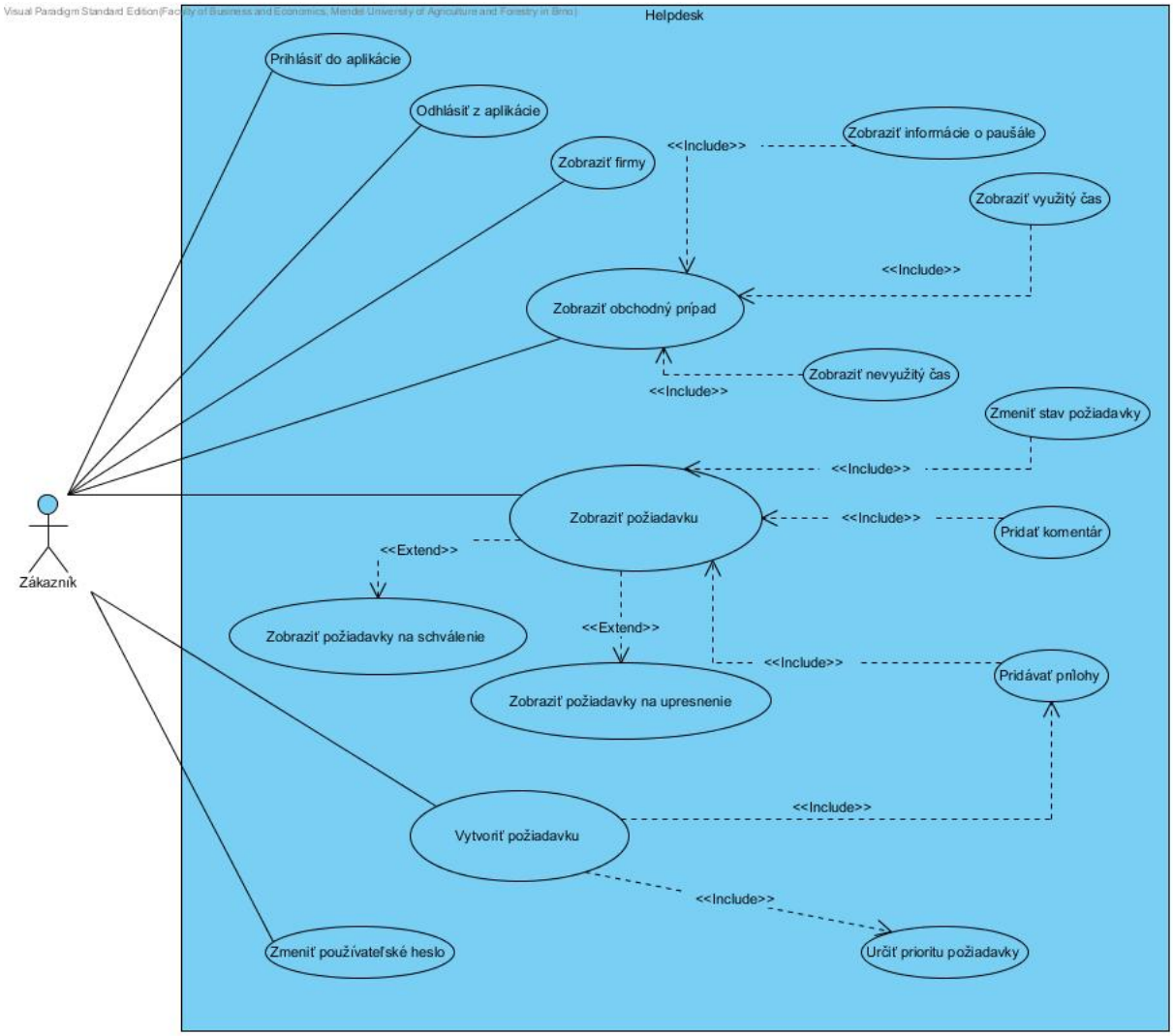
- *Business case* - obchodný prípad obsahuje okrem identifikačných atribútov aj atribúty o paušále, kedy začal a dokedy bude trvať. Takisto aj koľko hodín práce obsahuje paušál a koľko z toho sú minuté hodiny.
- *Business case user* - je to spájacia tabuľka medzi tabuľkami *business_case* a *user*. Definuje, ktorý používateľ má prístup k danému obchodnému prípadu, ale dokonca aj to, či používateľ môže schvaľovať požiadavky ostatných používateľov a či má zobrazené všetky požiadavky patriace obchodnému prípadu.

5.3 Návrh spracovania požiadavky

V nasledujúcej kapitole sú navrhnuté diagramy potrebné k vytváraniu, spracovávaniu a práci s požiadavkou.

5.3.1 Use Case

V rámci systému Helpdesk je možné sa zamerať na prípady použitia z pohľadu zákazníka a z pohľadu systému, ktorý synchronizuje dáta.V tejto kapitole je bližšie popísaný prípad použita z pohľadu zákazníka (viď obr. 13).



Obr. 13: Prípad použitia helpdesku

Helpdesk obsahuje tieto prípady použitia:

- Prihlásenie a odhlásenie z aplikácie.
- Zobrazenie firmy.
- Zobrazenie obchodného prípadu - systém helpdesk umožňuje zobraziť informácie o obchodnom prípade, ako napríklad zobraziť informácie o paušále. Tieto údaje obsahujú dátumy začatia a ukončenia prípadu, počtu predplatených hodín. Pre používateľov sú podstatné informácie o využitom a nevyužitom čase.
- Zobrazenie požiadavky - pri zobrazení detailu požiadavky môže oprávnený používateľ meniť stav požiadavky, pridať komentár alebo vložiť novú prílohu. Taktiež si používateľ môže vybrať, či chce zobraziť všetky požiadavky, alebo len tie

na schválenie alebo na upresnenie.

- Vytvorenie požiadavky.
- Zmena používateľského hesla.

5.3.2 Diagram tried

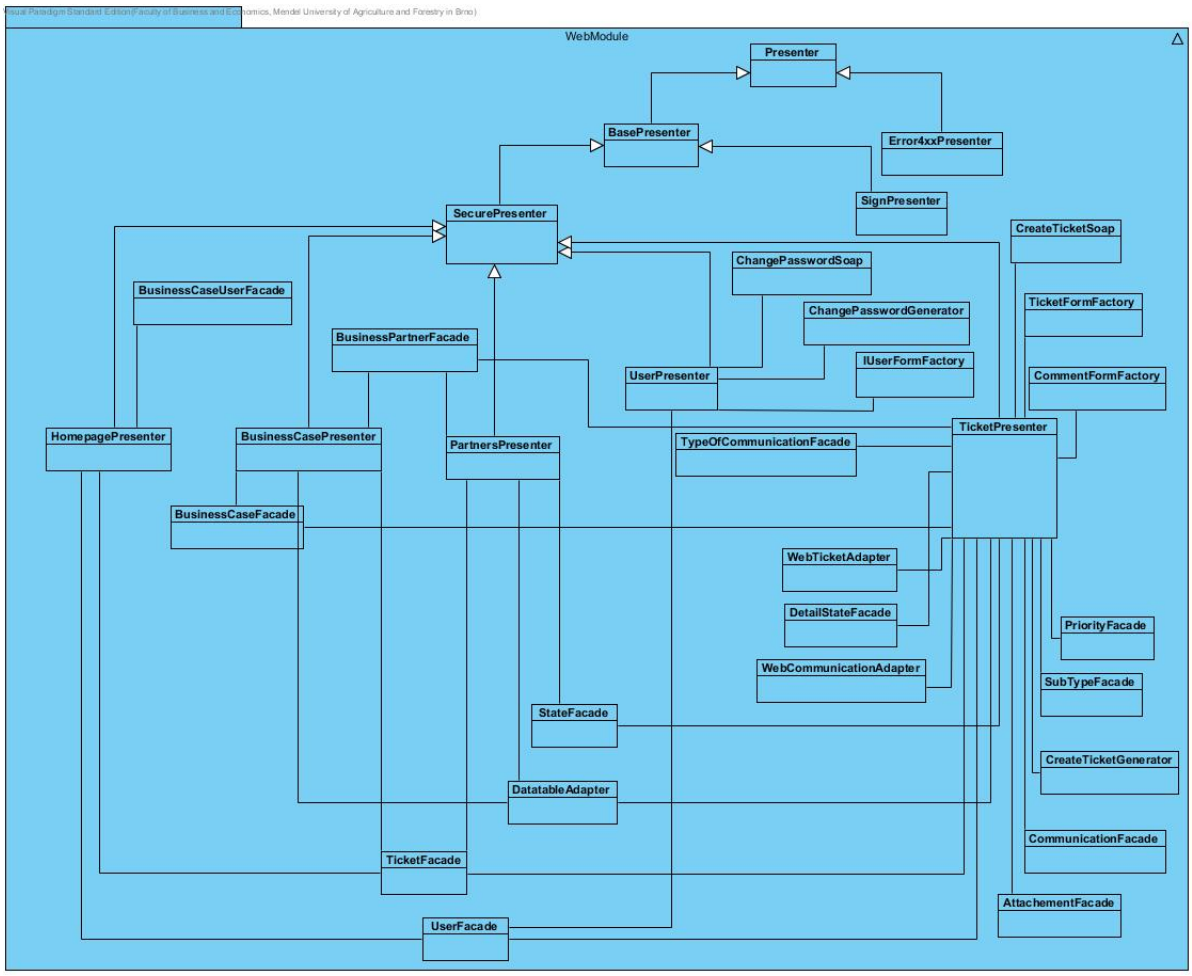
Aplikácia helpdesk bola navrhnutá modulovo tak, aby sa pri rozširovaní alebo zmene logiky musela upravovať čo najmenšia časť kódu a len na jednom mieste. Preto bol kód rozdelený do logických častí, a to ApiModulu a WebModulu.

ApiModul

ApiModul je časť aplikácie, ktorá rieši synchronizáciu zo strany systému QI. Úlohou prezentéra je prekladanie url na konkrétnu akciu, taktiež na základe typu požiadavky parsuje a validuje XML. Potom tieto dáta pošle do adaptéra, kde z nich vytvorí entity, ktoré následne fasáda spracuje uložením alebo aktualizáciou do databázy.

WebModul

Modul, ktorý zabezpečuje synchronizáciu dát prihlasenému používateľovi. Šablonovací systém vykreslí údaje o firmách, obchodných prípadoch a požiadavkách. Taktiež obsahuje formuláre, ktoré po odoslaní automaticky odošlú údaje do systému QI. Bližšie je webmodul opísaný na obrázku 14.



Obr. 14: Diagram tried

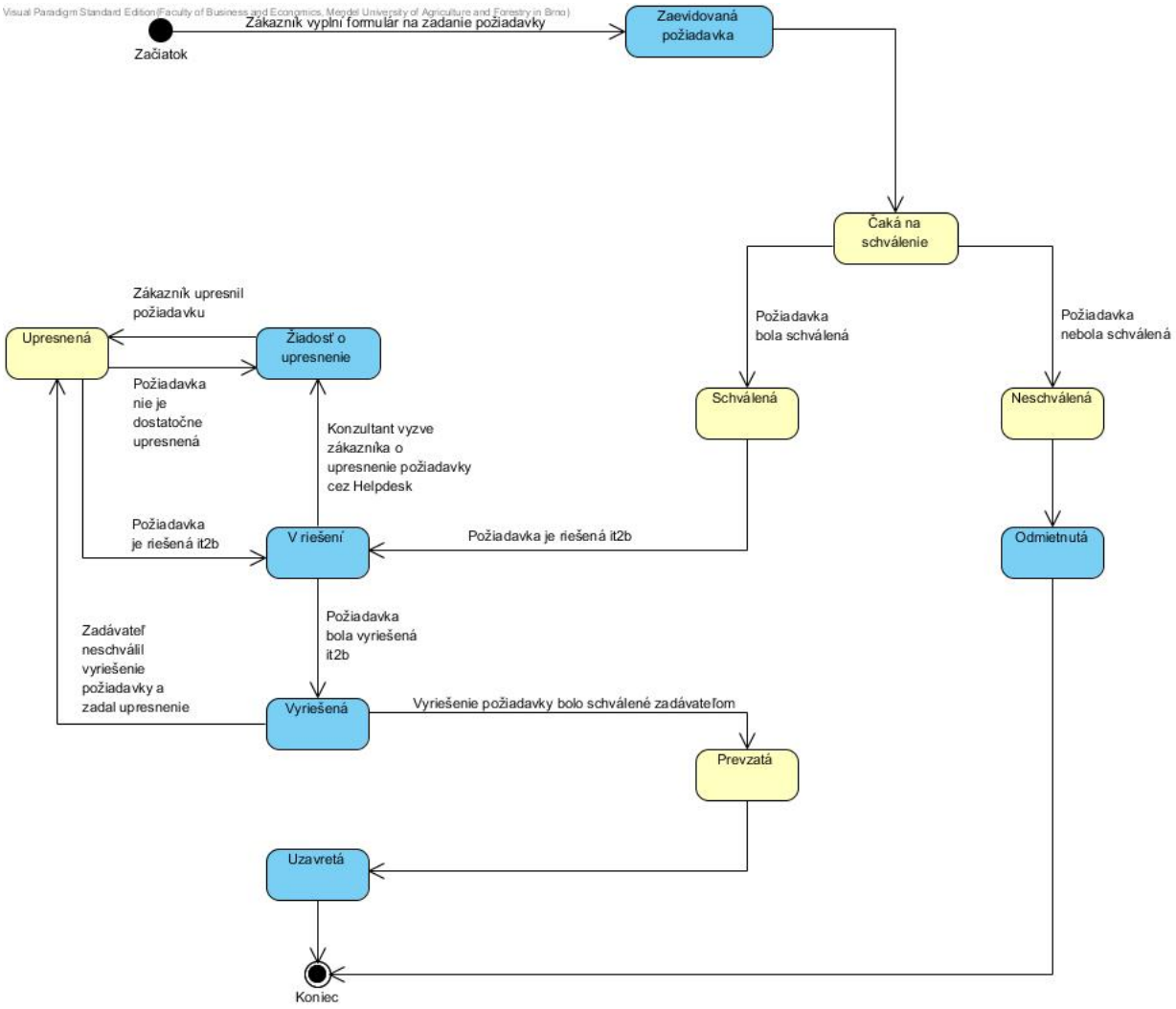
Diagram tried WebModulu pozostáva z nasledujúcich častí:

- Presenter - jedna z vrstiev MVC. Prezentér na základe url adresy určí akciu, ktorá sa má vykonať. V rámci prezentéru sa neuchováva logika aplikácie, tá je uložená v modeloch, ktoré zavolá prezentér v danej akcii.
- Facade - Triedy facade v rámci implementácie obaľujú funkcionalitu spojenú s databázou, ako napríklad získanie údajov podľa určitého používateľa, firmy.
- Adapter - Vo WebModule rozlišujeme dva typy adaptérov. Prvý typ sa zameriava na vytváranie entít z dát, ktoré sú poslané prostredníctvom formulára. Úlohou druhého typu adaptéra je pripraviť dáta na ich zobrazenie pre plugin DataTable.
- Factory - triedy, ktoré obsluhujú vytváranie a spracovávanie formulárov.

- Generator - pomocou tried Generator sa po vytvorení nových záznamov vytvorí XML dokument, ktorý slúži na synchronizáciu dát v systéme QI.
- Soap - pripojenie k vytvoreným službám Soap sa vykonáva vrámci tried Soap. Nachádza sa tu konkrétny názov služby, na ktorý sa daná trieda chce pripojiť. Ako vstupné parametre k soap funkcii sa uvádzajú prihlasovacie údaje a XML dokument vytvorený v triede Generator.

5.3.3 Stavový diagram

Každá požiadavka prechádza počas celého životného cyklu niekoľkými stavmi a ich podstavmi. Systém QI má šesť základných stavov. Pre bližšiu špecifikáciu bolo nutné vytvoriť rozsiahlejšiu štruktúru stavov, čo informačný systém QI neumožňoval. Z tohto dôvodu boli zavedené ich detaily. Priebeh životného cyklu je znázornený na obrázku 15.



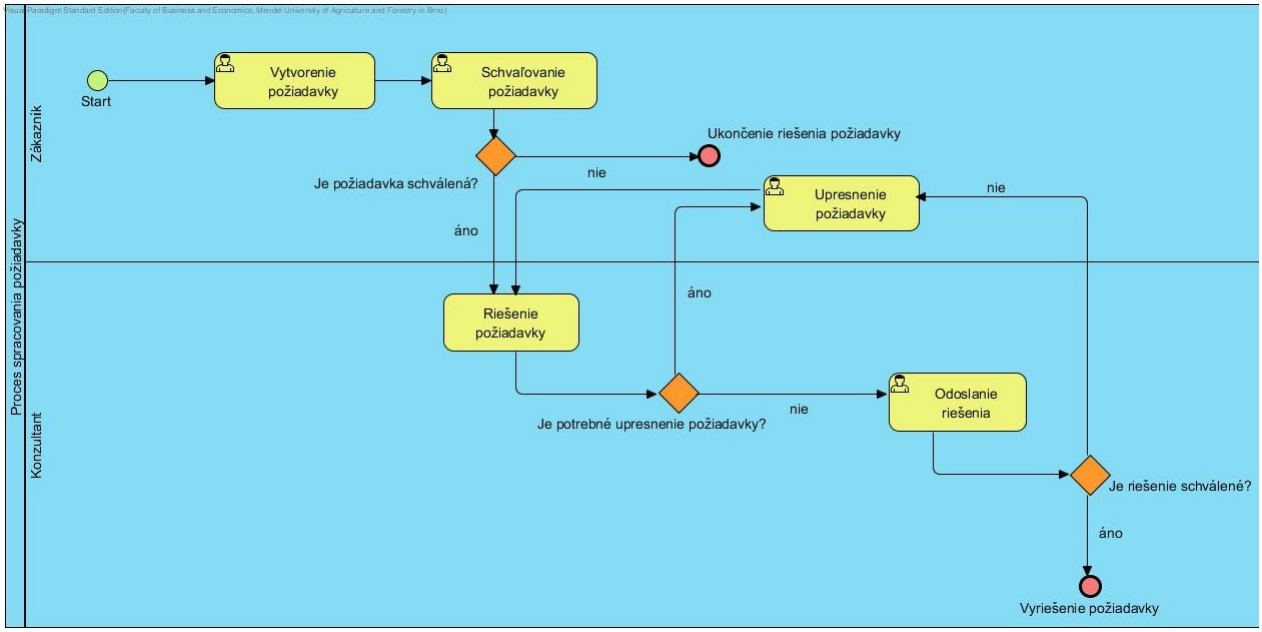
Obr. 15: Stavový diagram helpdesku

Po odoslaní formulára na vytvorenie požiadavky sa automaticky vyplní stav *zaevidovaná* a detail stavu *čaká na schválenie*. Používateľ, ktorý má právo schvaľovať požiadavky pre daný obchodný prípad, je oprávnený zmeniť detail stavu na *schválená* alebo *neschválená*. V prípade, že požiadavka nie je schválená k ďalšiemu spracovávaniu, automaticky sa vyplní stav *odmietnutá* a životný cyklus končí. V opačnom prípade požiadavka čaká na riešenie zo strany prevádzkovateľa systému QI. Ten po začatí riešenia problému zmení stav na *v riešení*. Ak konzultant QI nerozumie zadaniu, prejde požiadavka do stavu *žiadosť o upresnenie*. Používateľ, ktorý zadal požiadavku, ju môže upresniť pomocou komentára a následne zmeniť detail stavu na *upresnená*. Ak konzultant vyrieši požiadavku, zmení stav na *vyriešená*. Pokiaľ riešenie nie je podľa predstáv zadávateľa, môže vrátiť detail stavu na *upresnená* a konzultant opäť rieši zadanie podľa nového upresnenia. V opačnom prípade zmení detail stavu na *prevzatá*, pričom automaticky prejde do stavu *uzavretá* a životný

cyklus sa ukončí.

5.3.4 Procesný diagram

Proces spracovania požiadavky počas celého jej životného cyklu od vytvorenia až po jej vyriešenie je znázornená na obrázku 16.



Obr. 16: Procesný diagram spracovania požiadavky

Životný cyklus spracovania požiadavky začína jej vytvorením. Následne prostredníctvom vedúceho pracovníka prechádza procesom schvaľovania. Ak požiadavka nie je schválená, je celý proces spracovávania požiadavky ukončený. V opačnom prípade je požiadavka riešená konzultantom. Ak zadanie požiadavky nie je pre konzultanta dostačujúce, požiadavka prechádza do procesu upresnenia, kde zákazník bližšie špecifikuje zadanie. Po vyriešení daného problému konzultant odošle riešenie, ktoré ak je vyhovujúce proces riešenia sa ukončí, ináč zákazník bližšie špecifikuje požiadavku a zopakujú sa potrebné kroky.

5.4 Synchronizácia dát

K synchronizácií dát bol zvolený dokument XML. Pre vytvorenie správnej schémy bola vytvorená špecifikácia a pravidlá, aby bol dokument XML validný. K validácií vstupných dokumentov sa využíva dokument XSD (viď obr. 17).

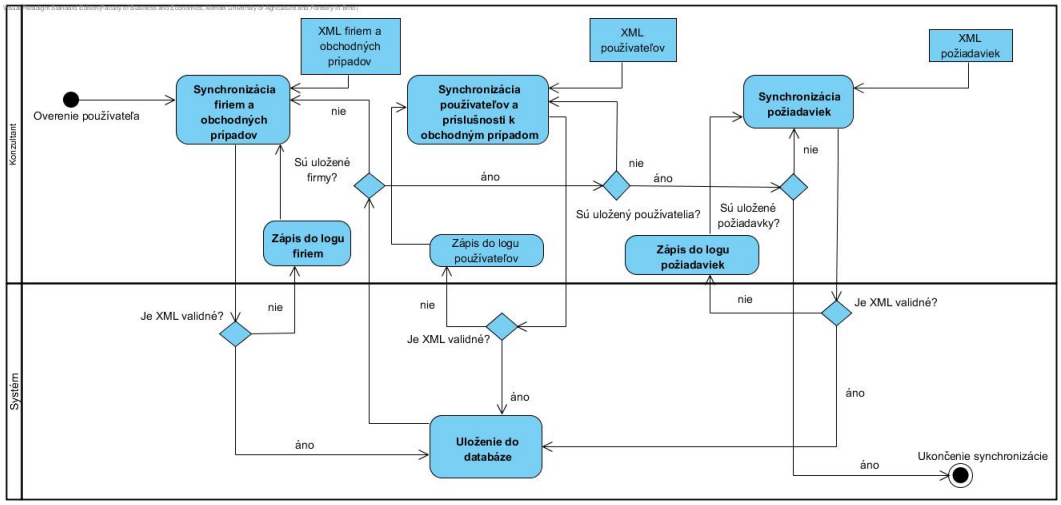
```xml
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="business_partners">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="business_partner" minOccurs="0" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:choice maxOccurs="unbounded">

                            <xs:element name="city" type="xs:string" />
                            <xs:element name="zip_code" type="xs:string" />
                            <xs:element name="dic" type="xs:string" />
                            <xs:element name="ico" type="xs:string" />
                            <xs:element name="name" type="xs:string" />
                            <xs:element name="street" type="xs:string" />
                            <xs:element name="business_case" minOccurs="0" maxOccurs="unbounded">
                                <xs:complexType>
                                    <xs:choice maxOccurs="unbounded">
                                        <xs:element name="name" type="xs:string" />
                                        <xs:element name="start_actual_tariff" type="xs:date" minOccurs="0"/>
                                        <xs:element name="end_actual_tariff" type="xs:date"  minOccurs="0" />
                                        <xs:element name="total_tariff_hour" type="xs:float"  minOccurs="0" />
                                        <xs:element name="spend_resolved_requirement" type="xs:float"  minOccurs="0"/>
                                        <xs:element name="spend_closed_requirement" type="xs:float"  minOccurs="0" />
                                        <xs:element name="remains_hours_for_period" type="xs:float"  minOccurs="0" />
                                        <xs:element name="total_spend" type="xs:float"  minOccurs="0" />
                                        <xs:element name="free_hours" type="xs:float"  minOccurs="0" />
                                        <xs:element name="from_previous_period" type="xs:float"  minOccurs="0" />
                                    </xs:choice>
                                    <xs:attribute name="IC" type="xs:integer"/>
                                    <xs:attribute name="U" type="xs:integer"/>
                                </xs:complexType>
                            </xs:element>
                        </xs:choice>
                        <!-- definition of attributes -->
                        <xs:attribute name="IC" type="xs:integer"/>
                        <xs:attribute name="U" type="xs:integer"/>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

Obr. 17: Návrh XSD pre firmu

V XSD dokumente sú špecifikované atribúty, ktoré sa vypĺňajú pre jednotlivé firmy a ich obchodné prípady. Nachádza sa tam sekvencia firiem, pričom minimálny počet firiem v dokumente je 0 a maximálny počet je neobmedzený. Pri každom elemente sú definované ich typy. Element *business_case* definuje obchodné prípady patriace k danej firme, kde podobne ako u elementu *business_partner* je minimálny počet 0 a maximálny nie je obmedzený. Pre každý obchodný prípad a firmu sú evidované atribúty IC a U, ktoré označujú jedinečný identifikátor v rámci systému QI.

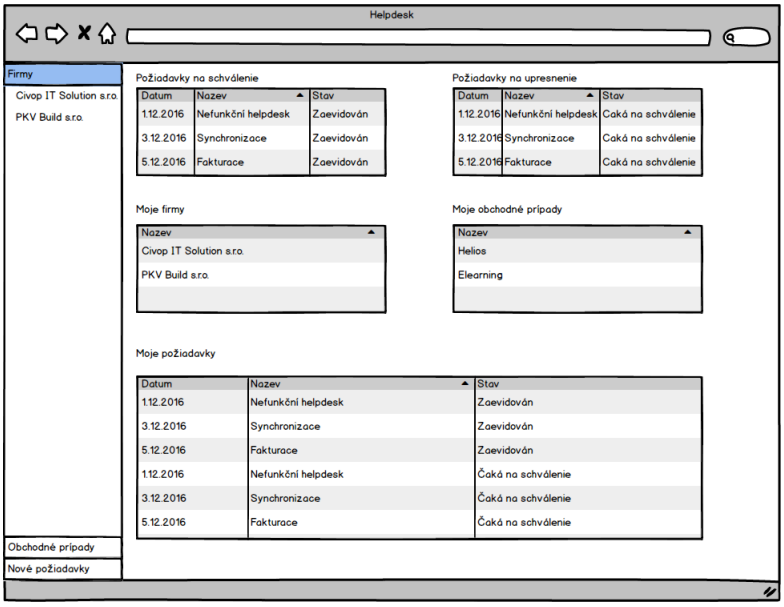
Podobne ako pre synchronizáciu obchodných partnerov a prípadov, aj pre používateľov a požiadavky sú vytvorené špecifikácie XSD, podľa ktorých sa vytvára dokument XML. Synchronizácia údajov musí prebiehať v určitom poradí (viď obr. 18), ktoré sa musí dodržať kvôli závislostiam medzi jednotlivými synchronizačnými entitami.



Obr. 18: Proces synchronizácie dát

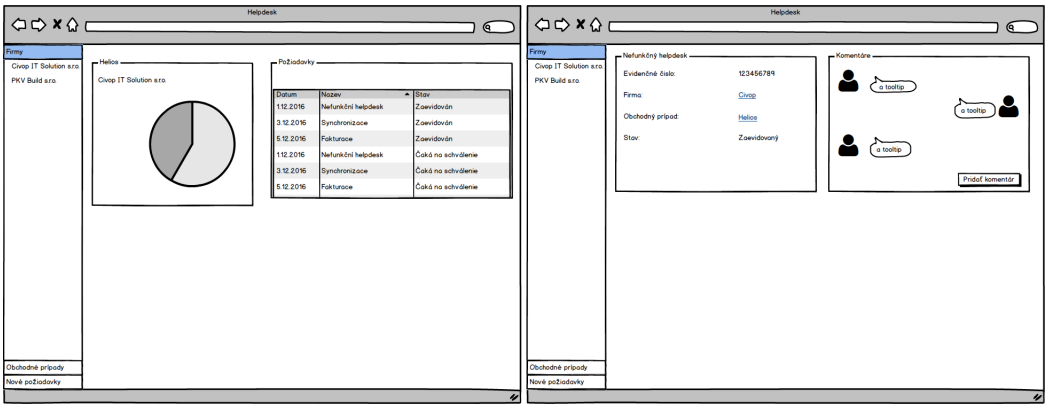
5.5 Vizuálny návrh systému

Pri návrhu systému je dôležitým krokom vytvorenie grafického návrhu. Grafický návrh musí byť prispôsobený požiadavkám spoločnosti it2b s.r.o., ktorá pri stanovení požiadaviek na systém vyžadovala jednoduchosť a intuitívnosť systému. Návrh bol vytvorený pomocou mockupov, konkrétne nástroja Balsamiq Mockups. Pri tvorbe grafického návrhu neboli presne stanovené farby jednotlivých prvkov, hoci požiadavka zo strany zákazníka bola použitie farieb červená, šedá, biela. Pre zákazníka je podstatné rozloženie prvkov ma stránke.



Obr. 19: Grafický návrh dashboardu

Základnou obrazovkou po prihlásení je dashboard (viď obr. 19). Na tejto stránke by mali byť zobrazené dôležité informácie, ku ktorým sa má používateľ dostať čo najrýchlejšie. V hornej časti obrazovky sa budú nachádzať požiadavky, ktoré čakajú na schválenie alebo upresnenie od prihláseného používateľa. Následne sú vykreslené tabuľky, ktoré obsahujú zoznam firiem a obchodných prípadov, ku ktorým má používateľ prístup. Zoznam požiadaviek daného používateľa je vykreslený v dolnej časti stránky, nakoľko ak bude mať viac požiadaviek, nebudú zakrývať iné dôležité informácie.



Obr. 20: Grafický návrh detailu obchodného prípadu (vľavo) a detail požiadavky (vpravo)

Detail konkrétneho obchodného prípadu (viď obr. 20 vľavo) zobrazuje požiadavky, ktoré patria obchodnému prípadu a taktiež informácie o paušále. Pri paušále je nutné zobraziť, koľko hodín je z celkového počtu vyčerpaných, preto bol zvolený graf.

Pre vykreslenie detailu požiadavky (viď obr. 20 vpravo) bolo zvolené rozdelenie obrazovky na dve časti. Prvá z nich obsahuje údaje o požiadavke, ako napríklad do akého obchodného prípadu patrí, aké je jej evidenčné číslo. V druhej časti obrazovky sú zobrazené komentáre k danej požiadavke. Tie slúžia na upresnenie a evidovanie komunikácie medzi konzultantom a zákazníkom. V časti s komunikáciou sa budú zaznamenávať všetky zmeny, ktoré sa s požiadavkou vykonajú. Napríklad pri zmene stavu sa zobrazí čas a meno používateľa, ktorý danú zmenu vykonal.

6 Implementácia

V tejto kapitole je opísaná implementácia helpdesku, ktorá vychádza z návrhu aplikácie, ako to bolo opísané v kapitole 5. Jednotlivé podkapitoly budú zamerané na dôležité komponenty aplikácie.

Na vývoj aplikácie bolo využité IDE PhpStorm (PhpStorm, 2016), ktoré sa zameriava na vývoj aplikácií v jazyku PHP. PhpStorm obsahuje nástroj Test RESTful Web Service, ktorý umôžňuje testovanie služieb Rest.

6.1 Synchronizácia údajov

Synchronizácia je jednou z najdôležitejších častí aplikácie. Musí prebiehať obojsmerne, teda zo systému QI smerom do helpdesku a naopak. Pre posielanie dát do systému QI bola určená webová služba Soap, ktorou informačný systém QI disponuje a má už naimplementované niektoré služby. Preto spoločnosť it2b s.r.o. požadovala prenos dát pomocou tejto technológie. Pre opačný prenos dát bola zvolená služba Rest Api, ktorá poskytuje jednoduchý spôsob prenosu dát medzi aplikáciami. Tento spôsob je taktiež jednoduchý na implementáciu. Nie sú k tomu potrebné zložité knižnice, avšak pri integrácií tejto technológie do frameworku Nette je vytvorených niekoľko knižníc. Tie sprehľadňujú a uľahčujú vývoj. Služba Rest Api ponúka aj niekoľko výhod, pre ktoré bola implementovaná v tejto aplikácii, a to sú napríklad:

- Aplikácia s využitím služby Rest Api je ľahko prenositeľná. V prípade, že spoločnosť it2b s.r.o. sa rozhodne prepísať zdrojové kódy informačného systému do iného jazyka, v aplikácii helpdesk nebudú nutné žiadne úpravy posielania údajov do QI.
- Táto technológia patrí medzi najrozšírenejšie, dokáže preniesť akýkoľvek typ informácií.
- Využíva štandardné typy HTTP požiadaviek.
- V budúcnosti možná rozšíriteľnosť aplikácie helpdesk. Pri zobrazovaní dát na strane klienta je možné volanie služby Rest Api.

6.1.1 Soap

Pre využívanie Soap servera naimplementovaného na strane QI bolo nutné vytvoriť Soap klienta v helpdesku. Testovanie webových služieb prebiehalo v aplikácii SoapUI (SoapUI, 2016), kde po prihlásení je dostupný zoznam služieb implementovaných v QI. Táto aplikácia umožňuje sledovať, čo konkrétne sa posiela v jednotlivých požiadavkách.

Po odoslaní formulára, napríklad na vytvorenie požiadavky, sa z vyplnených údajov vygeneruje XML dokument podľa vopred stanovenej špecifikácie. Pri implementácií je nutné vytvoriť Soap klienta pomocou definície WSDL dokumentu. Nasledujúci príklad ukazuje vytvorenie Soap klienta.

```
new \SoapClient('http://99.99.99.99/cgi-bin/icdisp.exe?act=
    ↪ wsdl')
```

Pri volaní konkrétnej služby sa zadávajú parametre:

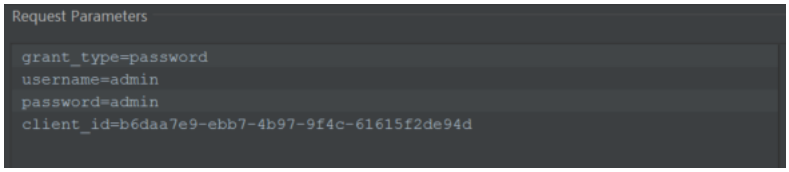
- používateľské meno,
- používateľské heslo,
- vstupný dokument XML - vygenerovaný z odoslaného formulára,
- vystupný dokument XML - QI vráti údaje poslané z formulára doplnené o identifikačné údaje IC a U,
- výstupná správa - vráti stav spracovania údajov, v prípade, že služba prebehla v poriadku vráti OK, ak nastala pri spracovaní nejaká chyba vráti ERROR + <popis chyby>.

```
client->HelpdeskChangePassword('stupakova','12345', $xml,
    ↪ $xmlOut, $message)
```

6.1.2 Rest Api

Pre synchronizáciu firiem, obchodných prípadov a používateľov sa zavolá webová služba na strane helpdesku. Pre využívanie týchto služieb je potrebné prihlásenie prostredníctvom OAuth2. Následne môže prebiehať komunikácia. Testovanie webových služieb prebiehalo pomocou IDE PhpStorm a nástroja Test RESTful Web Service.

Pred zavolaním služieb v helpdesku je nutné prihlásenie pomocou OAuth2. Prihlásenie má presne definovanú url adresu s určitými parametrami (viď obr. 21), ktoré sú presne stanovené. Každý používateľ má svoje používateľské meno a heslo, prostredníctvom ktorých sa môže prihlásiť.



Obr. 21: Požiadavka access token

Pre získanie tokenu sa podľa parametrov zloží následujúca url adresa.

```
http://helpdesk.it2b.cz/api/v1/authorization/token?grant\
    ↪ _type=password\&username=admin\&password=admin\&
    ↪ client\_id=b6daa7e9-ebb7-4b97-9f4c-61615f2de94d
```

V odpovedi sa vygeneruje:

- token,
- čas expirácie,
- typ tokenu,
- refresh token.

Vygenerovaný token zadáme pri volaní iných služieb. Token je po určitom čase neplatný, preto sa v odpovedi generuje aj *refresh_token*. Pomocou neho je možné po uplynutí času požiadať o nový token bez nutnosti overovania.

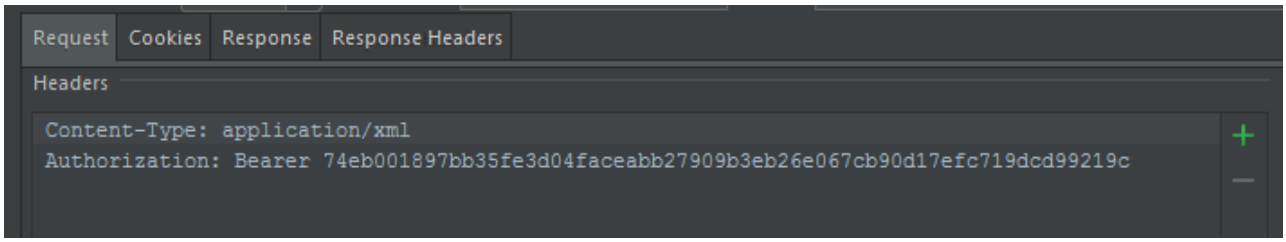
Konkrétne služby, ktoré môžeme volať prostredníctvom Rest Api:

- synchronizácia firiem a obchodných prípadov,
- synchronizácia používateľov a ich príslušnosti k obchodným prípadom,
- synchronizácia požiadaviek.

Každú z týchto služieb môžeme zavolať prostredníctvom nasledujúcej url adresy.

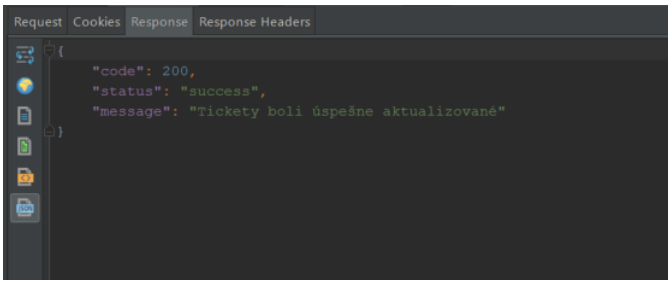
```
http://helpdesk.it2b.cz/api/v1/<názov žsluby>/default
```

A každá HTTP požiadavka obsahuje v hlavičke parametre (viď obr. 22).

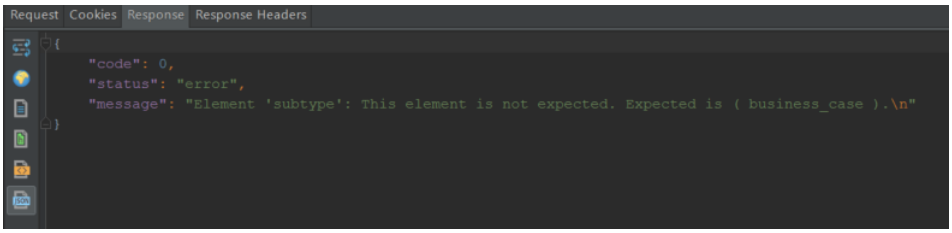


Obr. 22: HTTP hlavička webovej služby

Ako telo požiadavky sa vloží XML súbor, ktorý je potrebné spracovať. To sa následne zvaliduje podľa vopred vytvoreného XSD. Dáta z dokumentu sa aktualizujú alebo uložia do databázy, tento proces je opísaný v podkapitole Aktualizácia dát. Ak spracovanie prebehne v poriadku, odpoveď požiadavky vráti status *success* (viď obr. 23). V opačnom prípade vráti status *error* s bližším popisom chyby, ktorá nastala (viď obr. 24).



Obr. 23: HTTP hlavička webovej služby



Obr. 24: HTTP hlavička webovej služby

Aktualizácia dát

Keďže v aplikácií helpdesk sa budú zobrazovať rovnaké údaje ako v informačnom systéme QI, je nutné ich synchronizovať. Pretože firmy a obchodné prípady sa v helpdesku nevytvárajú, synchronizácia funguje len jedným smerom a to v smere z QI do helpdesku. Preto bolo nutné ošetriť ako ukladanie tak aj aktualizáciu údajov.

Existuje niekoľko možností, ako sa môže táto funkcionalita vykonať. Jednou z nich je aktualizácia na úrovni ORM, druhou na úrovni databázy. Pri rozhodovaní, či sa majú dané údaje uložiť, aktualizovať alebo sa v oboch systémoch zhodujú sa viac osvedčila metóda na úrovni databázy. Hlavne z dôvodu, že aktualizácia v ORM je výpočetne náročnejšia.

V zvolenej variante synchronizácie údajov na úrovni databázy, boli vytvorené MySQL procedúry. Tieto procedúry vytvoria temporárne tabuľky, v ktorých sa nachádzajú dáta z XML dokumentu. Temporárne tabuľky sa uchovávajú v pamäti, preto práca s nimi je nenáročná a rýchla.

Napríklad synchronizácia firiem pozostáva z nasledujúcich krokov:

- Vytvorenie temporárnej tabuľky *input_partner*, ktorá obsahuje atribúty zo vstupného XML dokumentu.
- Výber unikátnych údajov z tabuľky *business_parnter*.
- Vytvorenie alebo updatovanie údajov prebieha pomocou príkazu *INSERT INTO ... ON DUPLICATE KEY UPDATE ...*. Pričom ak existuje v tabuľke *business_parnter* kombinácia údajov IC a U, ktoré sú v systéme QI jednoznačným

identifikátorom, údaje sa len aktualizujú. V opačnom prípade sa uložia do databázy.

- Vymazanie firiem, ktoré sa v súbore XML nenachádzajú.
- Vymazanie temporárnej tabuľky.

Rovnaký postup sa vykonáva pre tabuľky obchodných prípadov, používateľov, požiadaviek.

6.2 Databázová vrstva

Na základe dátového modelu opísaného v kapitole 5.2 (Dátový model) bola vytvorená databázová schéma. Databázový server použitý pre aplikáciu helpdesk je MySQL, čo bolo jednou z požiadaviek zadávateľa projektu. Konkrétne bol použitý ORM framework Doctrine, ktorý pracuje s dátami ako s objektami.

6.2.1 Generovanie entít

Na generovanie entít sa využíva skript, ktorý dedí od triedy *Console/Command* a je súčasťou Symfony komponent. Tento generátor nebol dostačujúci pre potreby aplikácie helpdesk, preto bol vytvorený nový, ktorý má podobnú logiku, avšak je prispôsobený pre vyvíjanú aplikáciu.

Skript na generovanie entít obsahuje napríklad:

- nastavovanie namespace *Models* - priečinok, do ktorého sa majú entity vygenerovať,
- nastavenie *entityManager* - ORM Doctrine disponuje viacerými verziami entityManager, v rámci skriptu na generovanie entít sa určí, ktorý bude aplikácia helpdesk využívať,
- viditeľnosť premenných - pre potreby helpdesku je nutné aby atribúty mali viditeľnosť nastavenú ako *protected*, aby s nimi mohli fasády pracovať,
- generovanie anotácií,
- nastavenie prefixu pri anotáciách.

Konkrétna ukážka generátora:

```
$generator->setFieldVisibility(EntityGenerator::
    ↪ FIELD_VISIBLE_PROTECTED);
$generator->setGenerateAnnotations(true);
$generator->setAnnotationPrefix('ORM\\');
$generator->setGenerateStubMethods(true);
$generator->setBackupExisting(false);
$generator->setRegenerateEntityIfExists(false);
```

```
$generator->setClassToExtend('\\Nette\\Object');
$generator->setUpdateEntityIfExists(false);
```

6.2.2 DQL

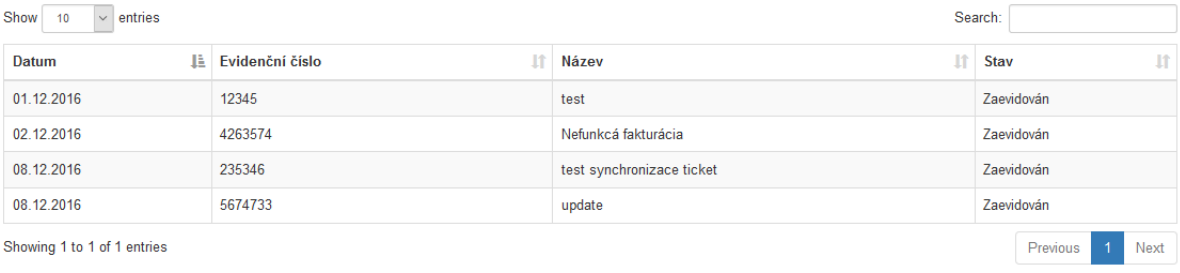
Pre vytvorenie zložitejších dotazov sa v ORM Doctrine využíva jazyk DQL. Pre výstavbu DQL dotazov sa v rámci aplikácie využíva návrhový vzor Builder, konkrétne QueryBuilder. Poskytuje triedy a funkcie, pomocou ktorých je možné ľahko zostaviť aj zložitejší dotaz. V aplikácii helpdesk sa využíva jazyk DQL pre získavanie dát do tabuliek firiem, obchodných prípadov a požiadaviek. Nasledujúca ukážka demonštruje vytvorenie DQL dotazu pomocou QueryBuilder:

```
$qb = $repository->createQueryBuilder()
            ->select('u')->from(User::class, 'u')
            ->innerJoin('u.businessCaseUser', 'bcu')
            ->innerJoin('bcu.businessCase', 'bc')
            ->innerJoin('bc.businessPartner', 'bp');
$qb->addSelect('bcu, bc, bp');
```

Funkcie *select(), from(), join()* a podobné plne nahrádzajú dotazy s rovnakým pomenovaním ako v SQL. Pritom pracujú s entitami ORM Doctrine, kde je definovaný názov tabuľky, stĺpce a taktiež vzťahy na relačné entity. Pre potreby pluginu DataTable bolo potrebné implementovať dotazy pre filtrovanie, stránkovanie a radenie. Boli na to využité funkcie:

- *setParameter()* - priradenie hodnoty do premennej v DQL dotaze,
- *addOrderBy()* - radenie hodnôt podľa stĺpca, ktorý je uvedený ako parameter funkcie,
- *setMaxResult()* - počet vrátených záznamov,
- *setFirstResult()* - udáva pozíciu, od ktorej sa získavajú dáta.

Výsledkom je vytvorenie tabuľky, v ktorej je možné vyhľadávať, radiť, zobraziť určitý počet údajov (viď obr. 25).



Obr. 25: DataTable obchodných prípadov

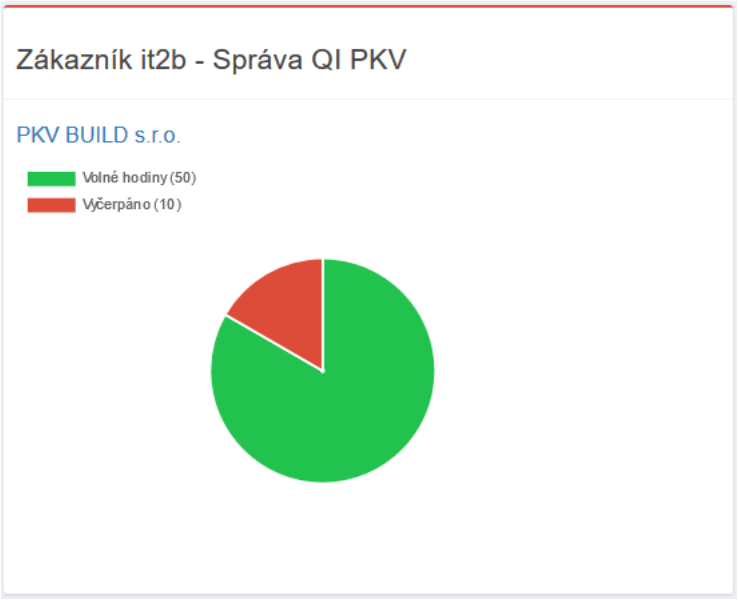
6.3 Šablóna

Pre vytvorenie aplikácie helpdesk bola využitá šablóna Admin LTE2. Táto šablóna vyhovovala spoločnosti it2b s.r.o. pre vzhľad helpdesku. Je jednoduchý, disponuje viacerými farebnými variantami. Taktiež obsahuje množstvo prvkov či už tlačidlá, formuláre, tabuľky, grafy.

Šablóna je založená na CSS frameworku Bootstrap3. Táto knižnica obsahuje množstvo mudulov v HTML, CSS a javascripte. To zabezpečuje, že všetky tlačidlá, formulárové elementy, nádpisy sú v jednom štýle. Taktiež podporuje responzívny dizajn, čo umožňuje zobraziť helpdesk aj na mobilných zariadeniach, čo bolo ďalšou požiadavkou zadávateľa. Mobilným zariadenia sa prispôsobujú tabuľky, menu, grafy.

Súčasťou šablóny sú aj grafy, ktoré pomáhajú lepšie vizualizovať dáta. Pre používateľov to je prívetivejšia forma prezentácie dát. Na vykreslenie grafov bol zvolený plugin *Chart.js* (Chart.js, 2016), ktorý ponúka rôzne typy grafov. Pre každý graf sú definované číselné hodnoty, ktoré sa v grafe zobrazujú. Podľa druhu hodnôt a ich prezentácie je zvolený vhodný typ grafu a taktiež jeho vizuálna stránka ako napríklad farby grafu, farba písma, umiestnenie legendy (viď obr. 26).

V budúcnosti je plánované zobrazenie viacerých grafov týkajúcich sa požiadaviek a paušálu. Tieto požiadavky však ešte neboli zo strany zadávateľa presne definované.

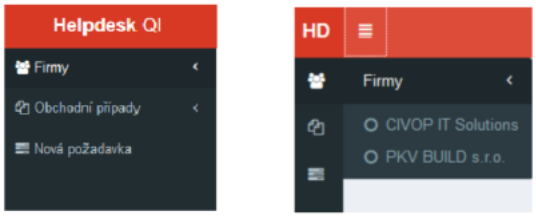


Obr. 26: Graf paušálu

6.3.1 Menu

Aplikácia obsahuje viacúrovňové menu, ktoré je zobrazené v ľavej časti obrazovky (viď obr. 27). Menu je možné zabaliť, pričom sa zobrazia len grafické ikony a po

kliknutí sa rozbalí daná položka. To je výhodou pri zobrazovaní webu na mobilných zariadeniach, kde je dôležitejší obsah stránky.



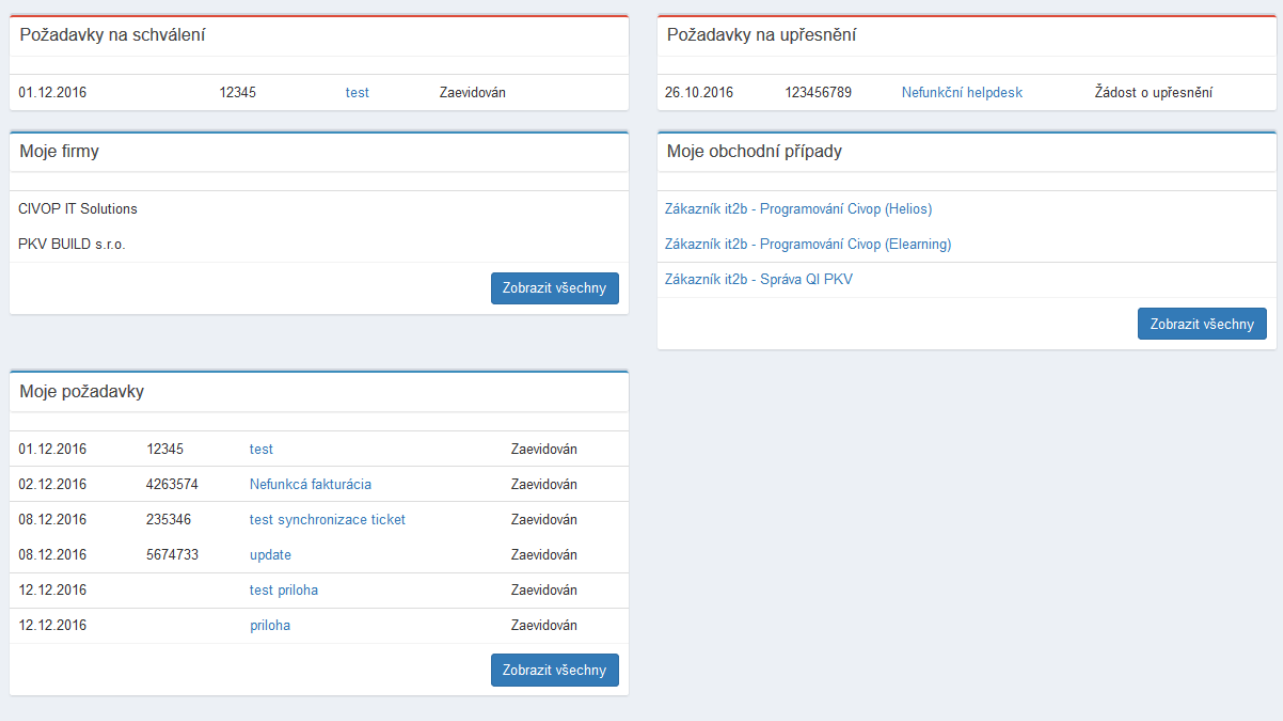
Obr. 27: Menu

Menu je tvorené dvoma typmi položiek a to:

- statické - medzi statické položky patrí *Firmy, Obchodné prípady a Nová položka.* Tieto sú zobrazené pre každého používateľa. Sú definované v konfiguračnom súbore spolu s ikonami, ktoré sa pri jednotlivých položkách zobrazujú.
- dynamické - tvoria druhú úroveň položiek menu. Položky sú načítavané z databázy, podľa prihláseného používateľa. Zobrazujú sa mu firmy a obchodné prípady, ku ktorým má prístup.

6.3.2 Dashboard

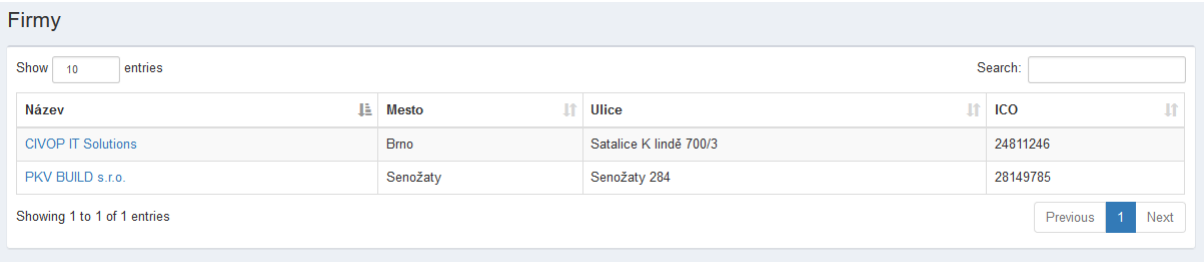
Po prihlásení sa používateľovi zobrazí hlavná stránka - dashboard (viď obr. 28). Na tejto stránke má nájsť prihlásený používateľ všetky dôležité informácie týkajúce sa firiem, obchodných prípadov a požiadaviek, ku ktorým má prístup. Obrazovku je možné rozdeliť na tri sekcie, ktoré zobrazujú rôzne typy informácií. Požiadavky v prvej sekcii čakajú na určitú akciu od prihláseného používateľa. V druhej sekcii sú zobrazené firmy a obchodné prípady, s ktorými je spojený. A v poslednej sekcii požiadavky.



Obr. 28: Dashboard

V hornej časti obrazovky sú zobrazené požiadavky, ktoré majú priamu súvislosť s prihláseným používateľom a čakajú na jeho schválenie alebo upresnenie. Zadané požiadavky musia prejsť procesom schválenia od používateľa, ktorý má pridelené právo schvaľovať požiadavky. Požiadavka po schválení je riešená konzultantom QI, v opačnom prípade je odmietnutá a jej životný cyklus končí. V sekcii *Požiadavky na upresnenie* sa požiadavka zobrazí vtedy, ak prihlásený používateľ zadal problém na riešenie, no konzultant potrebuje bližšiu špecifikáciu problému.

Ďalšia časť obsahuje zoznam firiem a obchodných prípadov. Na dashboarde je zobrazený len stručný výpis, kde je zobrazených len päť firiem a päť obchodných prípadov, ktoré majú najviac požiadaviek. Taktiež sa tam nachádza možnosť zobraziť všetky položky daného zoznamu (viď obr. 29). Po zobrazení zoznamu je možné vyhľadávať, zoradiť jednotlivé záznamy.

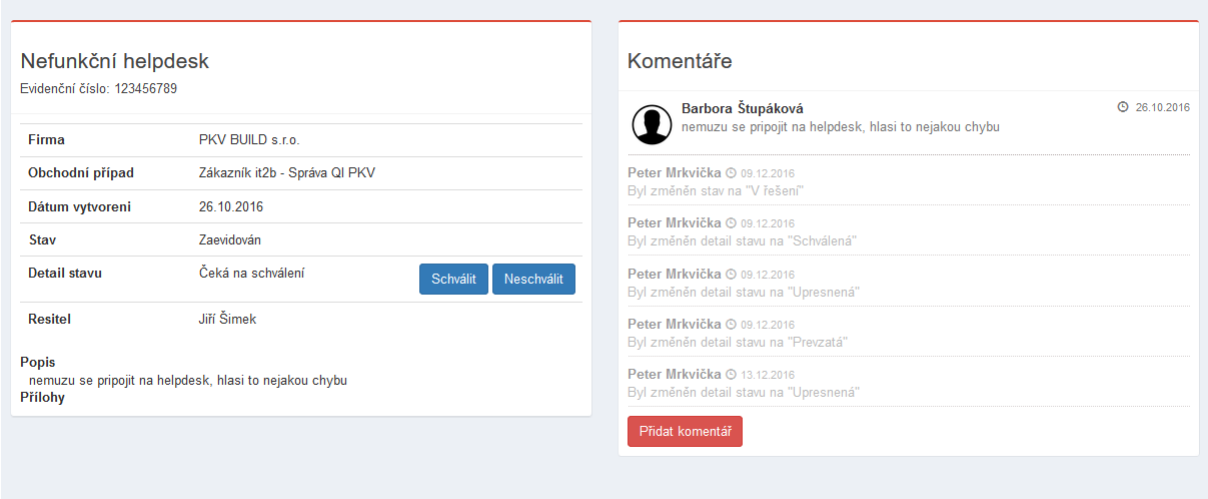


Obr. 29: Zoznam firiem

V dolnej časti obrazovky sa nachádza zoznam požiadaviek súvisiacich s prihláseným používateľom. Podobne ako pri výpise firiem a obchodných prípadov je možné medzi požiadavkami vyhľadávať, zoraďovať podľa jednotlivých stĺpcov.

6.3.3 Požiadavky

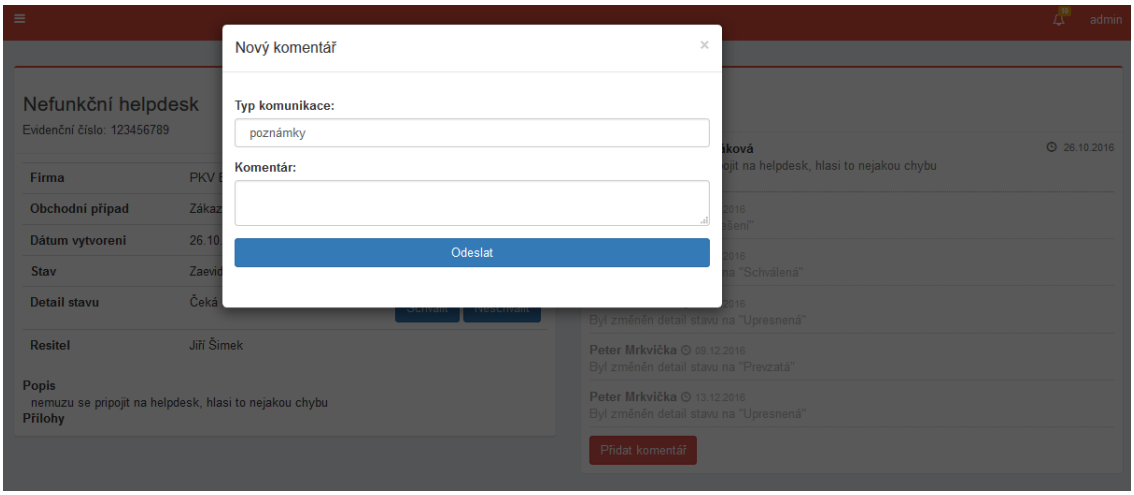
Najdôležitejšou časťou aplikácie je práca s požiadavkami. Ak zákazník potrebuje vyriešiť nejaký problém s informačným systémom QI, zadá požiadavku, ktorá je následne spracovaná konzultantmi QI.



Obr. 30: Detail požiadavky

Zobrazenie detailu požiadavky (viď obr. 30) je rozdelené na dve časti. Vpravo sú zobrazené detaily jednotlivej požiadavky ako napríklad akej firme a obchodnému prípadu patrí, kedy bola vytvorená alebo v akom je stave. Pri určitom stave alebo jeho detaile sa za presne daných podmienok nachádza možnosť zmeniť ho.

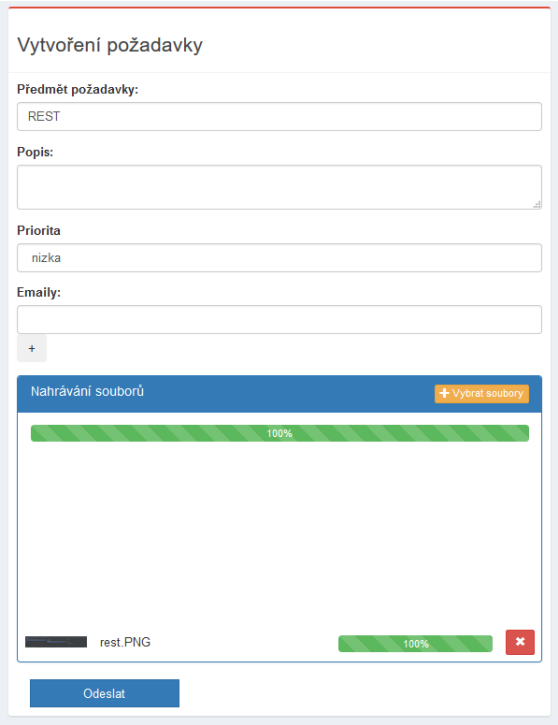
Vľavo sa nachádzajú komentáre k danej požiadavke. Ako komentáre sa zobrazujú aj zmeny stavov, alebo akákoľvek zmena požiadavky. Úlohou helpdesku bolo zaznamenávať všetkú komunikáciu medzi firmami a spoločnosťou it2b s.r.o. Každý záznam, či už ide o komentár alebo o zaznamenanie zmeny požiadavky má presnú štruktúru. Uchováva sa meno používateľa, ktorý zmenu vykonal, čas a samotný text. Pridávanie komentára je možné pomocou modálneho okna (viď obr. 31).



Obr. 31: Detail požiadavky

6.3.4 Upload súborov

Pri vytváraní požiadavky je aj možnosť pridať prílohu. Po výbere prílohy sa pri načítaní zobrazí *progress bar*, ktorý graficky znázorňuje ako sa príloha načítava (viď obr. 32). Pridávanie príloh bolo implementované asynchrónne. Bolo to hlavne z dôvodu, že pri pridaní príloh sa ihneď uložia do temporárneho priečinku, takže pri odosielaní formulára sa súbory už nespracovávajú. Používateľ pri pridaní viacerých príloh nemusí čakať pri odoslaní formulára. Každý používateľ má vlastný priečinok, kde sa ukladajú prílohy, ktoré vložil. Názov je vo formáte *dátum_názov súboru*.



Obr. 32: Vytvorenie požiadavky - upload súboru

6.3.5 Gulp

Cieľom npm balíčka gulp je zjednodušiť a zrýchliť webové aplikácie. V rámci aplikácie automatizuje kompiláciu CSS preprocesorov, spája a minifikuje súbory pre produkčné nasadenie, čím optimalizuje rýchlosť načítania webových stránok.

Čo sa týka optimalizácie webových sránok, je nutné si uvedomiť, čo všetko môže spôsobiť spomalenie. Spomalenie môže vzniknúť:

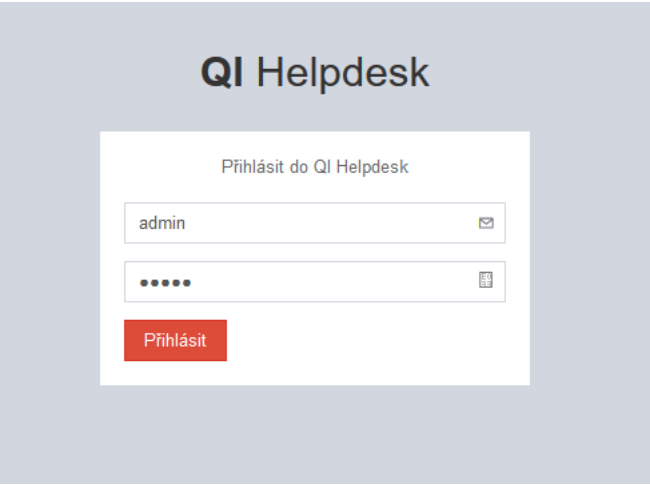
- veľkým množstvom dát prenášaných zo servera,
- každý HTML dokument obsahuje odkazy na štýly, skripty, obrázky. Každý odkaz si vyžaduje spojenie so serverom, a čím viac spojení, tým pomalšie je načítanie webových stránok.

Samotnú aplikáciu je možné optimalizovať rôznymi spôsobmi, ako napríklad minifikovaním štýlov a súborov javascript, alebo ich spájanie do čo najmenšieho množstva súborov. Pri spájaní súborov sa nemenia pôvodné súbory, ale výstup je zapísaný do nového súboru.

V rámci diplomovej práce boli minifikované a spojené CSS a JS súbory šablóny. Bol vytvorený minifikovaný súbor základných súborov css *style.min.css*, ktoré obsahovala šablóna. Následne bol vytvorený súbor *plugins.min.css*, ktorý obsahoval štýly všetkých pluginov, ktoré boli využívané. To isté sa vykonalo so súbormi javascriptu, pričom vznikol súbor *appConcat.js*.

6.4 Autentizácia a autorizácia

Pre používanie systému helpdesk je nevyhnutná autentizácia používateľov. Každý používateľ sa pri vstupe do systému musí autentizovať pomocou používateľského mena a hesla (viď obr. 33).



Obr. 33: Vytvorenie požiadavky - upload súboru

Aplikácia helpdesk neumožňuje registráciu samotným používateľom. Používateľské meno a heslo je vytvorené v systéme QI a pomocou synchronizácie údajov uložené do databázy helpdesku. Prostredníctvom týchto údajov sa používateľ môže následne autentizovať. Údaje o používateľovi nie je možné meniť, okrem prihlasovacieho hesla do helpdesku.

Po prihlásení sa kontroluje, či má používateľ právo vykonať určitú akciu. To sa vykonáva pomocou objektu *Authorizator* a metódy *isAllowed()*, ktorá rozhoduje, či používateľ má právo vykonať akciu. Na kontrolovanie jednotlivých akcií je vytvorený *SecurePresenter*, od ktorého dedia ostatné prezentéry. V ňom sa nachádza funkcia *startup()*, ktorá sa v životnom cykle prezentéra spúšťa vždy pred každou akciou. Práve preto je to ideálne miesto pre ošetrenie práv používateľov.

Pre lepšiu udržateľnosť sa práva uchovávajú v databáze. Tabuľka *resource* obsahuje názvy všetkých akcií, ktoré môže používateľ vykonať. Pre každú rolu sa v databáze v tabuľke *role_resoruce* uchovávajú akcie, ku ktorým má prístup.

7 Testovanie

Testovanie aplikácie prebiehalo počas celého vývoja aplikácie. Od zadania projektu až po jeho ukončenie prebehlo niekoľko stretnutí so zadávateľom. Počas celého vývoja sa aplikácia postupne menila s meniacimi sa požiadavkami spoločnosti it2b s.r.o.

Pri počiatočnom stretnutí boli stanovené základné požiadavky na aplikáciu helpdesk. Taktiež bolo vysvetlené fungovanie informačného systému QI a väzby medzi prvkami, ktoré boli použité pre helpdesk. Jednotlivé časti systému sa riešili v nasledujúcom poradí:

- Návrh grafického dizajnu - výber šablóny bol konzultovaný so zadávateľom. Bola vybraná farebná kombinácia podľa webových stránok spoločnosti. Taktiež boli zvolené jednoduché prvky. Po konzultácií boli zmenené prvky pre odhlásenie používateľa a zmenu hesla.
- Návrh databázy - podľa dátových rezov systému QI bola navrhnutá databáza. Aplikácia helpdesk však nemusí obsahovať všetky atribúty ako systém. Z tohto dôvodu boli niektoré atribúty z databázy helpdesku odstránené. Taktiež boli odstránené niektoré tabuľky, ktoré pre základné fungovanie helpdesku neboli potrebné. V tabuľkách, ktoré sa synchronizujú so systémom QI boli pridané atribúty IC a U, ktoré sú základným identifikátorom každého atribútu v QI.
- Návrh XSD dokumentov - podľa dát, ktoré bolo nutné synchronizovať, bola navrhnutá štruktúra XSD.
- Testovanie samotnej synchronizácie - testovanie synchronizácie so sebou prinieslo viacero problémov. Boli potrebné úpravy ako na strane helpdesku, tak aj na strane QI. Problém bol s pripojovaním na webové služby, s ohlasovaním chýb, ktoré sa pri synchronizácií vyskytli.
- Kompletizácia aplikácie - pri finalizácií vzniklo ešte niekoľko požiadaviek na úpravu ako napríklad pridanie javascriptu *drag and drop* na pridávanie príloh, zaistiť, aby v tabuľkách bolo možné pri kliknutí na celý riadok presmerovanie na požadovanú položku.

8 Diskusia

Výsledná aplikácia spĺňa požiadavky, ktoré boli určené na začiatku vývoja. Umožňuje synchronizovať údaje z informačného systému QI, zobrazovať ich používateľovi a zadávať nové požiadavky ku konkrétnemu obchodnému prípadu a firme. Taktiež podporuje vyhľadávanie medzi jednotlivými položkami, ich zoraďovanie.

Vývoj aplikácie helpdesk mal urýchliť zadávanie požiadaviek používateľov do informačného systému, bez využívania telefonického kontaktu alebo emailovej komunikácie. Oproti riešeniu, ktoré firma it2b s.r.o. využívala aplikácia helpdesk ušetrí čas konzultantom, keďže nebudú musieť zadávať požiadavky do systému QI podľa emailov alebo telefonických rozhovorov. Taktiež bude zachovaná všetká komunikácia medzi zákazníkom a riešiteľom požiadavky, vrátane časov a všetkych zmien, ktoré boli s požiadavkami vykonané.

Inovatívne prvky oproti súčasnému riešeniu je samotné zadávanie požiadaviek bez kontaktovania konzultantov. Taktiež zákazník vidí celý proces spracovania požiadavky a v priebehu celého životného cyklu môže komunikovať prostredníctvom aplikácie s riešiteľom požiadavky. Komunikácia je uchovávaná v rámci aplikácie a taktiež je možné nahrať prílohy, ktoré bolo nutné posielať emailom.

9 Záver

Cieľom tejto práce bolo vytvoriť komplexné riešenie helpdesk pre podporu informačného systému QI. Jedná sa najmä o nahradenie súčasného systému nejednotného zadávania požiadaviek jedným riešením, ktoré sa presunie z emailovej a telefonickej komunikácie na stranu webovej aplikácie.

Pred začatím návrhu a vývoja aplikácie bolo nutné preštudovanie a oboznámenie sa s funkcionalitou systému QI. Systém QI je veľmi rozsiahly, preto sa stačilo zamerať na spravovanie firiem, obchodných prípadov a správu zadaných požiadaviek. Taktiež spísanie funkčných a nefunkčných požiadaviek pomohlo presne špecifikovať, čo bude úlohou aplikácie helpdesk. Celý proces bol konzultovaný so zamestnancami spoločnosti it2b s.r.o.

Na základe oboznámenia sa so systémom QI a požiadavkami na helpdesk bol vytvorený návrh systému a synchronizácie údajov. Po schválení návrhu zadávateľom projektu a vedúcou práce bol systém vyvíjaný. Výsledný systém spĺňa stanovené ciele zadané na začiatku. Testovacia verzia helpdesku je dostupná na adrese http://helpdesk.it2b.cz/web/sign/in?_fid=21oo s vytvoreným testovacím účtom. Prihlásiť sa do aplikácie je možné pod prihlasovacím menom *admin* a heslom *admin*.

Návrh a implementácia prebiehala pod dohľadom konzultantov systému QI a vedúcej práce. Pripomienky a úpravy aplikácie boli riešené v priebehu vývoja. Pre implementáciu sa využívali moderné technológie, aby bola aplikácia v budúcnosti ľahko rozšíriteľná. Jednotlivé elementy boli z dôvodu prehľadnosti rozdelené do čo najmenších komponentov, aby to bolo prehľadné. Systém bol implementovaný v jazyku PHP vo frameworku Nette s využitím databázového systému MySQL.

10 Literatúra

About Git [online] 2016 [cit. 2016-12-14]. Dostupné z: https://git-scm.com/about/branching-and-merging.

Annotations Reference [online] 2016 Doctrine Team [cit. 2016-12-13] Dostupné z: http://docs.doctrine-project.org/projects/doctrine-orm/en/latest/reference/annotations-reference.html.

Arlow, Jim a Ila Neustadt. *UML 2 a unifikovaný proces vývoje aplikací: objektově orientovaná analýza a návrh prakticky 2.* aktualiz. a dopl. vyd. Brno: Computer Press, 2007, ISBN 978-80-251-1503-9.

Balsamiq Mockups [online] 2014 [cit. 2015-01-01]. Dostupné z: http://balsamiq.com/products/mockups/.

Bestpractice [online] 2016 [cit. 2016-09-27]. Dostupné z: https://www.bestpractice.sk/sk/Best-practice/-ITSM-ITIL/Co-je-to-ITIL-.alej.

Bestpractice [online] 2016 [cit. 2016-09-27]. Dostupné z: https://www.bestpractice.sk/sk/Best-practice/-ITSM-ITIL/Funkce-ITIL-/Service-desk.alej.

BRÓSKA, Michal. *Hotline, Call Centrum, Help Desk alebo Service Desk?* EFocus. 2009(4), 3..

Databases and the Doctrine ORM [online] 2016 [cit. 2016-12-13]. Dostupné z: http://symfony.com/doc/current/doctrine.html.

Doctrine Query Language [online] 2016 Doctrine Team [cit. 2016-12-13]. Dostupné z: http://docs.doctrine-project.org/projects/doctrine-orm/en/latest/reference/dql-doctrine-query-language.html.

Getting Started with Doctrine [online] 2016 Doctrine Team [cit. 2016-12-13] Dostupné z: http://docs.doctrine-project.org/projects/doctrine-orm/en/latest/tutorials/getting-started.html.

HILES, Andrew. *Creating a Customer-Focused Help Desk: How to Win and Keep Your Customers* Vyd. 1. Brookfield: Rothstein Publishing, 2016, ISBN 9781944480066..

Chart.js [online] 2016 [cit. 2016-12-19]. Dostupné z: http://www.chartjs.org/.

Informační systém QI [online] 2016 [cit. 2016-09-19]. Dostupné z: http://www.it2b.cz/informacni-system-qi/.

Introduction, 2016 HTML 5.1 [online] 2016 cit. 2016-12-15]. Dostupné z: https://www.w3.org/TR/2016/REC-html51-

20161101/introduction.html#introduction.

It2b [online] 2016 cit. 2016-12-15]. Dostupné z: http://it2b.cz/.

LECKY-THOMPSON, Ed a Steven D NOWICKI. *PHP 6: programujeme profesionálně* Brno: Computer Press, 2010, ISBN 978-80-251-3127-5.

Lesson 1: Systematically Approaching Design Stages [online] 2016 Microsoft [cit. 2016-12-13]. Dostupné z: https://technet.microsoft.com/en-us/library/cc505843.aspx.

LUBBERS, Peter, Brian ALBERS a Frank SALIM. *HTML5: programujeme moderní webové aplikace* Brno: Computer Press, 2011, ISBN 978-80-251-3539-6.

Návrhové vzory v PHP: [23 vzorových postupů pro rychlejší vývoj]. Brno: Computer Press, 2012, ISBN 978-80-251-3338-5.

Nette REST API, 2016 The OAuth 2.0 Authorization Framework [online] 2016 [cit. 2016-12-13]. Dostupné z: https://github.com/drahak/Restful.

Analysis and Design of Information Systems : Third Edition. London, 2008, ISBN 9781846286551.

OAuth 2.0, 2012 The OAuth 2.0 Authorization Framework [online] 2012 Microsoft [cit. 2016-12-13]. Dostupné z: https://tools.ietf.org/html/rfc6749#section-1.2.

O společnosti it2b s. r. o. [online]. 2016 [cit. 2016-09-19]. Dostupné z: http://www.it2b.cz/o-spolecnosti-it2b/.

PhpStorm [online] 2016 JetBrains [cit. 2016-12-20]. Dostupné z: https://www.jetbrains.com/phpstorm/.

QI Helpdesk [online] 2016. Brno: DC Concept a.s [cit. 2016-12-26]. Dostupné z: https://www.qi.cz/moduly/qi-helpdesk/.

ROCHKIND, M. *Expert PHP and MySQL: Application Design and Development.* 1. vyd. New York: Apress 2013 ISBN 978-1-430-26007-3.

RUBY, Sam, David THOMAS a David Heinemeier HANSSON *Ruby on Rails: průvodce agilním vývojem webových aplikací.* Brno: Computer Press, 2011, ISBN 9788025136478.

SoapUI [online] 2016 SmartBear Software [cit. 2016-12-20]. Dostupné z: https://www.soapui.org/.

Taskpool [online] 2016 [cit. 2016-10-06]. Dostupné z: http://www.taskpool.cz/.

Working with Objects Doctrine [online].Doctrine Team [cit. 2016-12-13] Dostupné z: http://docs.doctrine-project.org/projects/doctrine-orm/en/latest/reference/working-with-objects.html.

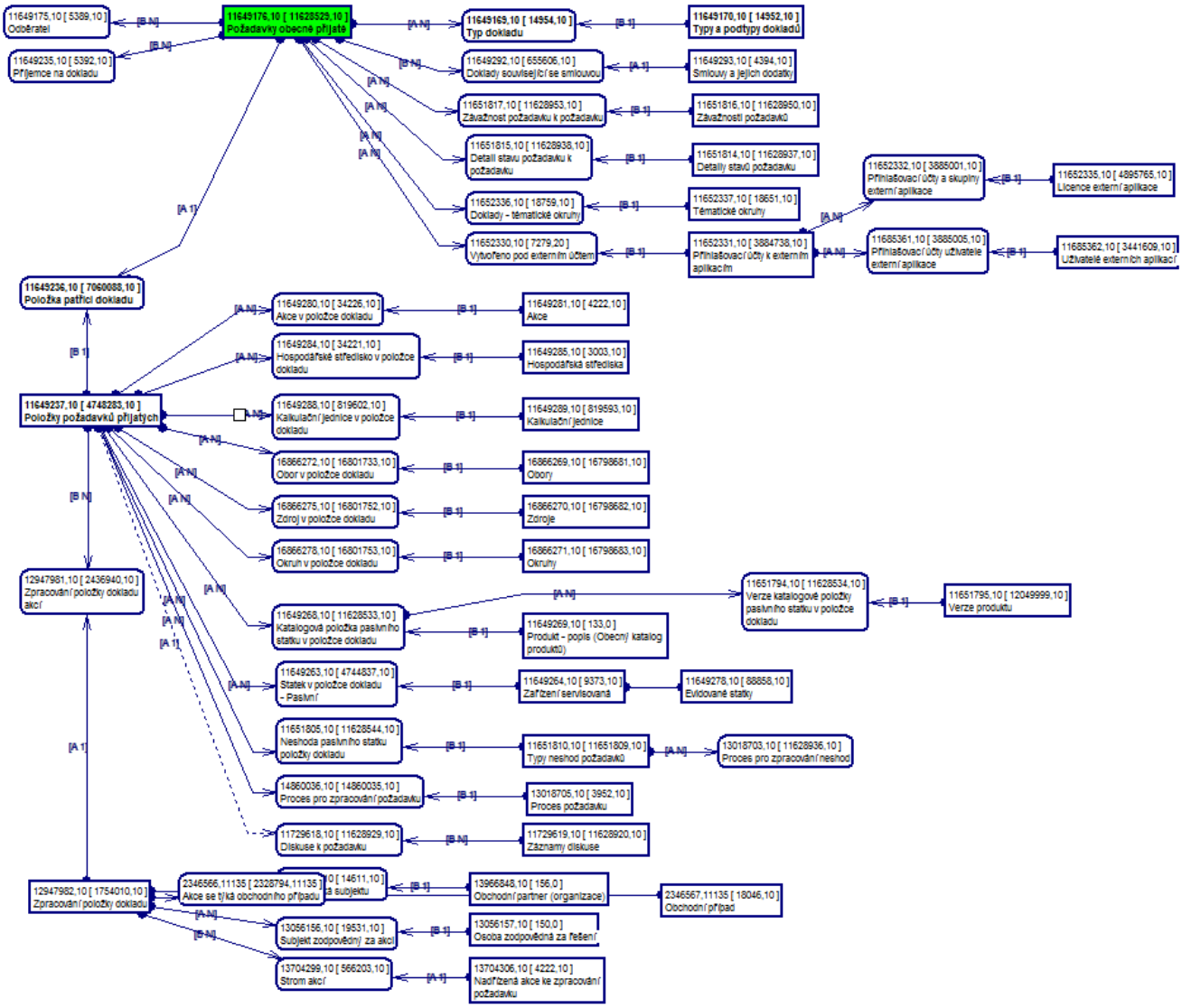
Prílohy

A Priložené CD

Priložené CD obsahuje:

- Zdrojové kódy aplikácie.
- UML diagramy

B Dátový model súčasného riešenia



Obr. 34: Dátový model súčasného riešenia