VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

INFORMAČNÍ SYSTÉM PRO ZPRACOVÁNÍ HLÁŠENÍ O CHYBÁCH

INFORMATION SYSTEM FOR ERROR REPORT MANAGEMENT

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **MARTIN DOMIN**
AUTHOR

VEDOUCÍ PRÁCE **doc. Ing. VLADIMÍR JANOUŠEK, Ph.D.**
SUPERVISOR

BRNO 2017

Zadání bakalářské práce/20137/2016/xdomin04

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav inteligentních systémů Akademický rok 2016/2017

Zadání bakalářské práce

Řešitel: **Domin Martin**
Obor: Informační technologie
Téma: **Informační systém pro zpracování hlášení o chybách**
Information System for Error Report Management
Kategorie: Softwarové inženýrství

Pokyny:

1. Prostudujte problematiku tvorby softwaru, zaměřte se na procesy testování aplikací a zpracování hlášení o chybách. Seznamte se s existujícími dostupnými nástroji, určenými ke zpracování hlášení o chybách, proveďte vyhodnocení a porovnání jejich funkcionality.
2. Prostudujte soudobé technologie tvorby webových aplikací.
3. Na základě výsledků bodu 1 a 2 navrhněte informační systém, který umožní efektivní komunikaci mezi uživateli, testery a vývojáři aplikací. Systém by měl používat pouze otevřené technologie, měl by být optimalizován pro použití v malé softwarové firmě a neměl by uživatele při práci zbytečně zatěžovat.
4. Navržený systém realizujte a proveďte jeho testování v prostředí malé firmy. Vyhodnoťte dosažené výsledky.

Literatura:

- Dle pokynů vedoucího.

Pro udělení zápočtu za první semestr je požadováno:

- První 2 body zadání a část návrhu.

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

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

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

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

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

doc. Dr. Ing. Petr Hanáček
vedoucí ústavu

Abstrakt

Tato bakalářská práce se věnuje problematice testování softwarových aplikací a jejich údržbě. Záměrem této práce je vytvořit aplikaci, která bude pomocným nástrojem nejen při testování softwarových výrobků, ale i při následném provozu a údržbě softwaru. Jedná se o nástroj, jehož hlavním úkolem je umožnit jak samotným testerům, tak i koncovým uživatelům patřičné aplikace evidovat nalezené problémy a chyby. Jednotlivé chyby je následně možné spravovat. Jelikož aplikace bude využívána také veřejnými uživateli testované či používané aplikace, kteří jsou různého věku a schopností, byl by při vývoji kladen důraz na jednoduchost ovládání.

Abstract

This bachelor thesis inquires the problematics of software testing and software maintenance. Motivation is to create an application, that would be a helpful tool not only for testing software applications, but also for future software maintenance. This application is a tool, which main purpose is to allow registration of problems and errors not only to the testers, but also to the public end-users of a particular application. Those individual problems can be managed using this system. Because the application will be used also by public end-users of a particular application who very differ of age and skill, there is an emphasis on ease of use and intuitivness of the system.

Klíčová slova

Software, Testování, Informační systém, Softwarové inženýrství, Kanban

Keywords

Software, Testing, Information system, Software engineering, Kanban

Citace

DOMIN, Martin. *Informační systém pro zpracování hlášení o chybách.* Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Janoušek Vladimír.

Informační systém pro zpracování hlášení o chybách

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana doc. Ing. Vladimíra Janouška Ph.D.. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Martin Domin
17. května 2017

Poděkování

Rád bych poděkoval vedoucímu Doc. Ing. Vladimíru Janouškovi, Ph.D. za vedení této bakalářské práce, společnosti EDIacademy s.r.o. za otestování této aplikace a své rodině za podporu.

Obsah

Kapitola 1

Úvod

1.1 Motivace

Údržba a provoz je etapa životního cyklu softwaru, která spotřebovává nejvíce úsilí (času a peněz). Je tedy důležité, aby aplikace před jejich nasazením byly kvalitně otestovány, a zároveň, aby jejich následná správa vyžadovala co nejmenší úsilí. Je tedy vhodné mít k dispozici takovou aplikaci, která by umožňovala co nejefektivnější správu nalezených chyb v softwarových aplikacích.

Na trhu existuje mnoho systémů tohoto typu, které umožňují firmám vytvářející softwarové výrobky jejich správu. Tyto aplikace spadají do kategorie Bug tracking systémů využívaných zejména v agilních metodikách. Ve většině případů se však jedná o komplexní systémy, které nejsou zcela vhodné pro potřeby malých firem, ať už díky jejich složitosti či ceně.

1.2 Cíle

Cílem této práce je navrhnout a vytvořit takový informační systém, který by umožňoval jednoduché hlášení chyb a jejich následnou správu. Hlavním cílem tohoto systému je zapojení koncových uživatelů testovaných či již používaných aplikací do samotného procesu jejich vývoje, testování či údržby.

Hlášení chyb v tomto systému by mělo být rychlé a přehledné. Například by uživatel neměl být zahlcen všemi možnostmi najednou, ale jednotlivé možnosti by uživateli byly předkládány postupně. Po uživateli by také mělo být vyžadováno při vyplňování formuláře pouze minimum nejnutnějších informací, abychom jej neodradili od užívání tohoto nástroje. Mezi tyto informace můžeme také například zařadit vkládání příloh, které by mělo být uživatelsky co nejpřívětivější.

Jedním z hlavních cílů systému je přehledné zobrazení zaslaných hlášení, které je podmínkou pro efektivní správu. K splnění tohoto cíle je vhodné použití tabule kanban, která dělí jednotlivé položky do kategorií podle jejich stavu. Systém by měl také usnadňovat spolupráci mezi testery a vývojáři.

Kapitola 2

Základní pojmy

V této kapitole si uvedeme základní pojmy z oblastí softwarového inženýrství a informačních systémů, s nimiž budeme dále v textu pracovat.

2.1 Data

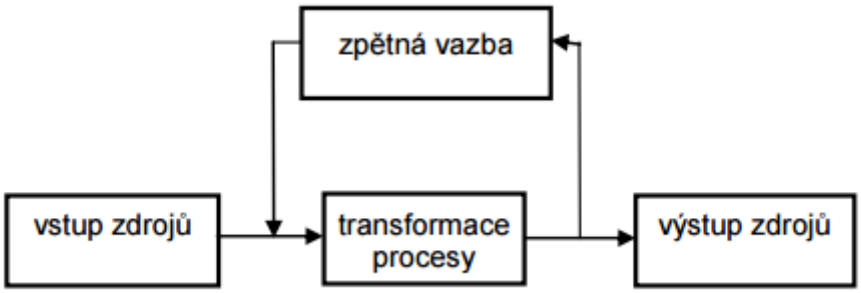
Data jsou jakékoli vyjádření (reprezentace) skutečnosti, schopné přenosu, uchování, interpretace či zpracování. Z hlediska počítačového jsou to pouze hodnoty různých datových typů. Podstatné je, že data nemusejí mít a často také nemají sémantiku (význam). Mohou být zpracovávána čistě formálně pouze na základě syntaxe (např. formální jazyky). [11]

2.2 Informace

Informace jsou data, která mají sémantiku (význam). Informace z dat vytváří uživatel jejich interpretací. Jde tedy o subjektivní proces, který může každý z uživatelů provádět jinak a velmi často tak také činí. Tím mohou vznikat a vznikají problémy. Je třeba zajistit (většinou jinak nežli výpočetní technikou), aby tato shodná interpretace dat na informace u všech uživatelů existovala.[11]

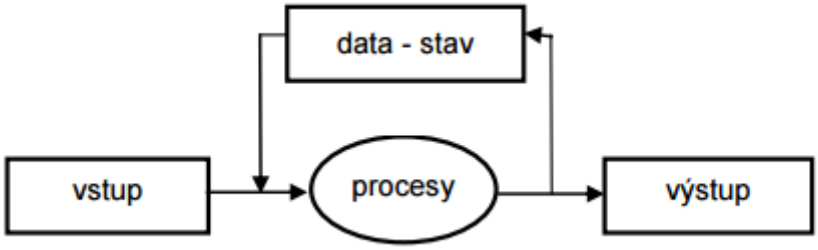
2.3 Informační systém

Obecný systém lze chápat jako množinu prvků a vazeb mezi nimi, které jsou účelově definovány na nějakém nosiči. Nosičem je množina prvků systému ve vzájemných informačních a procesních vztazích. Prvky nosiče nazýváme zdroje. Obecný systém se skládá ze vstupní a výstupní části, kudy se do systému vstupují, resp. se vystupují zdroje. Mezi vstupem a výstupem probíhá transformace těchto zdrojů. Transformace mění zdroje. Typickou (u obecných systémů nikoliv nezbytnou) částí systému je zpětná vazba. Ne všechny výstupy procesů jsou tedy přímo závislé na okamžitých vstupech, ale typicky závisejí na stavu systému. [11]



Obrázek 2.1: Schéma obecného systému [11]

Informační systém je pak obecným systémem pracujícím s konceptuálními zdroji. Proto lze část transformační a zpětnovazební pojmenovat konkrétněji. Informační systém se skládá ze vstupní a výstupní části, kudy se do systému vkládají, resp. se získávají informace. Mezi vstupem a výstupem probíhá transformace těchto dat. Transformační část typicky provádí nad daty různé algoritmy. Probíhají zde procesy a musíme se tedy zabývat způsobem definice procesů, jejich vzájemnou interakcí, paralelním prováděním apod. Typickou částí informačního systému je zpětná vazba, která využívá uloženého stavu systému. Ne všechny výstupy procesů jsou tedy přímo závislé na okamžitých vstupech, ale typicky závisejí na stavu systému. [11]



Obrázek 2.2: Schéma informačního systému [11]

2.4 Issue tracking systém

Issue tracking systém je počítačový software určený pro správu připomínek k systému. Připomínkou máme na mysli požadavek zákazníka například o vytvoření či aktualizaci funkcionality nebo požadavek na opravu chyby.

2.5 Bug tracking systém

Bug tracking systém je softwarová aplikace, která uchovává záznamy o nalezených problémech a chybách v jiných softwarových aplikacích. Můžeme je chápat jako speciální druh issue tracking systémů.

2.6 Agilní metodologie

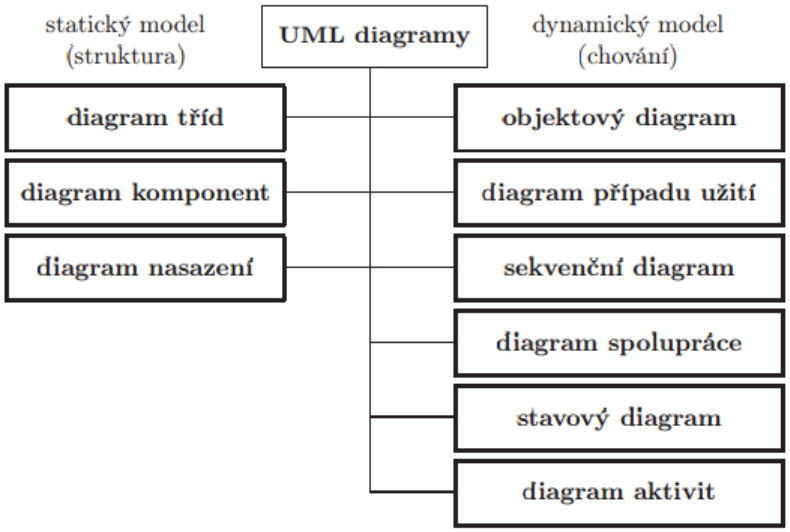
Agilní metodologie jsou odpovědí na problémy spjaté s vytvářením menších softwarových systémů, kdy použití klasických metodologií není vhodné. Agilní metodologie kladou hlavní důraz na člověka jako určující faktor pro kvalitu výsledného produktu. [12] Umožňují rychlý vývoj softwaru a zároveň dokáží reagovat na změnu požadavků v průběhu vývojového cyklu.

2.7 Jazyk UML

Unified Modeling Language (zkráceně UML) je v softwarovém inženýrství grafický jazyk pro vizualizaci, specifikaci, navrhování a dokumentaci programových systémů. UML operuje s pojmem pohled (view). Pohled systému je projekce systému na jeden z jeho relevantních aspektů. Pohledy nad systémem jsou pak modelovány prostřednictvím vhodných nástrojů (modelů) poskytovaných UML. Můžeme rozlišit tyto základní pohledy:

- Strukturální pohled (structural view) popisuje vrstvu mezi objekty a třídami, jejich asociace a možné komunikační kanály.
- Pohled chování (behavior view) popisuje, jak systémové komponenty (objekty) interagují, a charakterizuje reakce na vnější systémové operace.
- Datový pohled (data view) popisuje stavy systémových komponent (objekty) a jejich vazby.
- Pohled rozhraní (interface view) je zaměřeno na zapouzdření systémových částí a jejich potenciální použití okolím systému.

Jazyk UML nabízí několik základních diagramů. [12] Jejich přehled můžeme vidět na obrázku.



Obrázek 2.3: UML diagramy [12]

2.8 Provoz a údržba

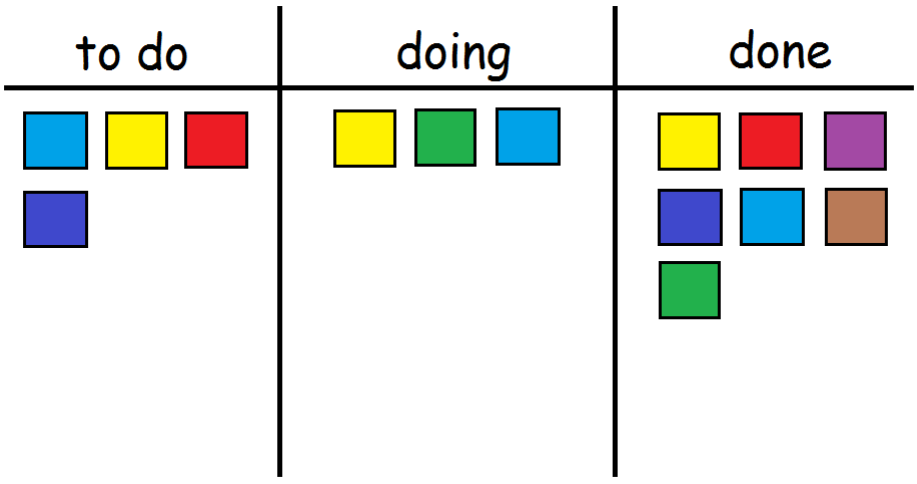
Jedná se o etapu v životním cyklu softwaru, která vyžaduje průběžné řešení problémů, které s nasazením a následným používáním softwaru souvisejí. Tato etapa také zahrnuje opravy nalezených chyb, rozšiřování softwaru o nové funkce či přizpůsobování softwaru měnícím se požadavkům okolí.

2.9 Framework

Softwarový framework může být reálná či konceptuální struktura určena pro podporu vývoje softwarových produktů. Jedná se o znovupoužitelné softwarové prostředí poskytující specifické funkcionality. Framework může obsahovat například podpůrné programy, knihovny či doporučené postupy při vývoji softwaru.

2.10 Kanban

Kanban je plánovací systém obvyklý u metodik jako jsou lean manufacturing (štíhlá výroba) či Just-in-time manufacturing (Just-in-time výroba - JIT). Tyto metodiky pocházejí z Japonské firmy Toyota a jejich cílem je minimalizovat náklady odstraněním plýtvání či rychlou komunikací se zákazníkem a rychlou výrobou. Tabule kanban[1] je nástroj pro vizualizaci postupu práce v čase a měla být brána jako jediný zdroj informací pro celý tým.



Obrázek 2.4: Tabule kanban

Běžně je kanban tabule dělena na tři části. V první části jsou zobrazeny úlohy, které je potřeba vyřešit, avšak práce na těchto úlohách ještě nezapočala (to do). V další části máme zobrazeny ty úlohy, na kterých se momentálně pracuje (doing). V poslední části jsou již ty úlohy, které byly dokončeny (done). Toto rozdělení je však pouze základní. Tabule kanban

[1] Tabulí kanban máme na mysli anglický výraz Kanban board

může být jakkoliv pozměněna aby vyhovovala konkrétním týmům a jejich cílům. V našem případě bychom si mohli přidat další kategorii určenou pro ty úlohy, které se momentálně nacházejí ve fázi testování.

Hlavními výhodami použití kanbanu jsou například snazší spolupráce v týmu či lepší sledovatelnost průběhu práce.

2.11 Cloud computing

Jedná se o model používání počítačových technologií, který charakterizuje poskytování služeb servery dostupnými z internetu, ke kterým uživatelé mohou vzdáleně přistupovat. Principem tohoto modelu je propůjčení výpočetního výkonu serverů vlastněných třetí stranou.

Kapitola 3

Použité technologie

Díky rozšíření internetového připojení v dnešním světě, je velmi vhodné vytvářet informační systémy jako cloudové webové aplikace. Toto umožnuje přístup k systému téměř odkudkoliv a též nezávisle na operačním systému. V této kapitole si tedy popíšeme webové technologie, které byly využity při vývoji této aplikace. Jako jeden z hlavních zdrojů při implementaci byly použity tutoriály ze stránek *World Wide Web Consortium* (dále v textu jako W3C) [18].

3.1 PHP

Jazyk PHP [10] (Personal Home Page) či dnes Hypertext Preprocessor od svého vzniku prodělal značný vývoj a stal se široce využívaným nástrojem pro tvorbu webových aplikací. Jazyk PHP primárně slouží pro vytváření dynamických webových stránek a k jejich propojení s databází. Samotné PHP skripty nejsou prováděny na straně uživatele, nýbrž na straně serveru. Uživateli je zasílán až samotný výsledek jejich činnosti. Mnoho částí systému je vytvářeno dynamicky, a tak tvoří PHP skripty velkou část našeho systému. Při implementaci byly jako zdroj použity oficiální stránky jazyka PHP [17] a kniha *PHP Cookbook* [16].

3.2 UserCake

UserCake [3] je otevřený PHP framework usnadňující vývoj aplikací. Jedná se o poměrně jednoduchý framework implementující primárně správu uživatelů, jednotlivých stránek a povolení přístupu různým uživatelům k datům.

3.3 HTML a CSS

HyperText Markup Language, neboli jazyk HTML [14] je značkovací jazyk, používaný pro vytváření webových stránek. HTML je typem dokumentu SGML (Standard Generalized Markup Language), kde je značkám přiřazena sémantika hypertextového dokumentu v prostředí Webu. V našem systému používáme jazyk HTML pro vytvoření těch částí, které nejsou generovány dynamicky pomocí jazyka JavaScript. Je však nutno podotknout, že téměř všechen HTML kód je vytvářen na straně serveru pomocí PHP skriptů a následně zasílán uživateli. Při vývoji byly informace čerpány kromě stránek W3C také z knihy *HTML & CSS The Good Parts* [8].

CSS (Cascading Style Sheets) či česky kaskádové styly jsou jazyk který popisuje, jak by měl být příslušný HTML dokument zobrazen. Kaskádové styly byly navrženy pro oddělení obsahu a grafické prezentace stránky.

3.4 Foundation

Foundation [19] je front-end framework pro vývoj webových stránek. Jedná se o kolekci HTML, CSS a Javascriptového kódu umožňující rychlejší vývoj responzivních stránek.

3.5 MySQL

Databázový server MySQL představuje jeden z nejrozšířenějších používaných relačních serverů. Na jeho rozšíření má zásluhu jednak fakt, že je vyvíjen jako software s otevřeným kódem a je k dispozici zdarma bez omezení a za druhé jeho poměrně dobrý výkon v porovnání s jinými servery stejné kategorie daný z velké části jeho relativní jednoduchostí. [15]

3.6 phpMyAdmin

Nástroj phpMyAdmin umožňuje jednoduchou, rychlou a přehlednou správu obsahu databáze MySQL prostřednictvím webového rozhraní. Použití tohoto systému značně ulehčilo mnoho úkonů a tak urychlilo vývoj této aplikace.

3.7 Jazyk JavaScript

JavaScript je interpretovaný programovací jazyk se základní objektově orientovanou koncepcí. Klientská verze tohoto jazyka (tj. verze pracující s DOM, pojmy jako událost prohlížeče, dokument, okno apod.) je součástí většiny všeobecně rozšířených prohlížečů jako jsou např. Internet Explorer a Mozilla Firefox [9]. V našem systému se Javascriptu využívá k dynamické zpětné vazbě mezi systémem a uživatelem bez nutnosti znovu generovat nové stránky. Dynamické vytváření kódu dělá stránky interaktivnější a přehlednější. Jako dobrý zdroj informací ohledně jazyka Javascript sloužila kniha *JavaScript The Definitive Guide* [7].

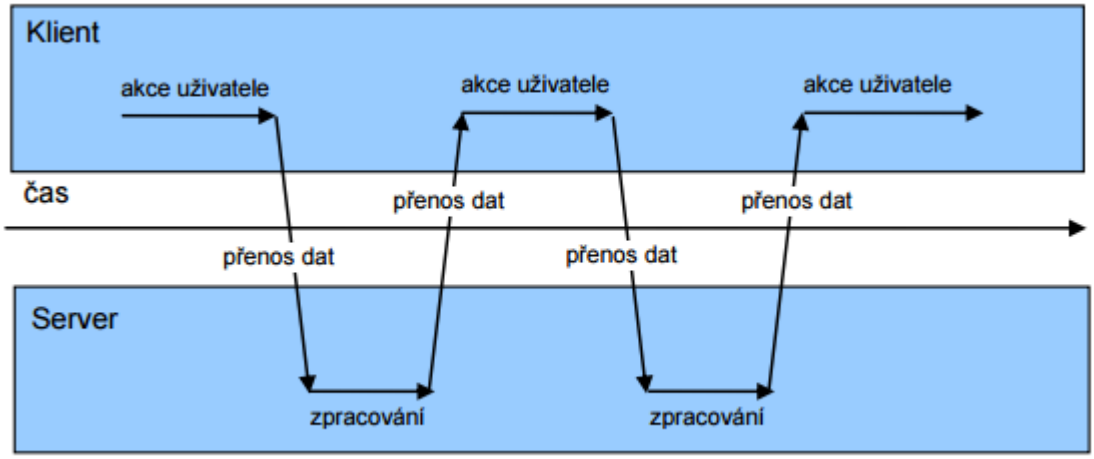
3.8 JQuery

jQuery je otevřená JavaScriptová knihovna která zjednodušuje interakce mezi HTML dokumenem, přesněji řečeno *Document Object Modelem (DOM)* a JavaScriptem [6]. Použití této knihovny nejen usnadnilo a urychlilo vývoj aplikace, ale také jednotlivé dostupné funkce vylepšily celkovou kvalitu systému jak po stránce vizuální, tak funkční.

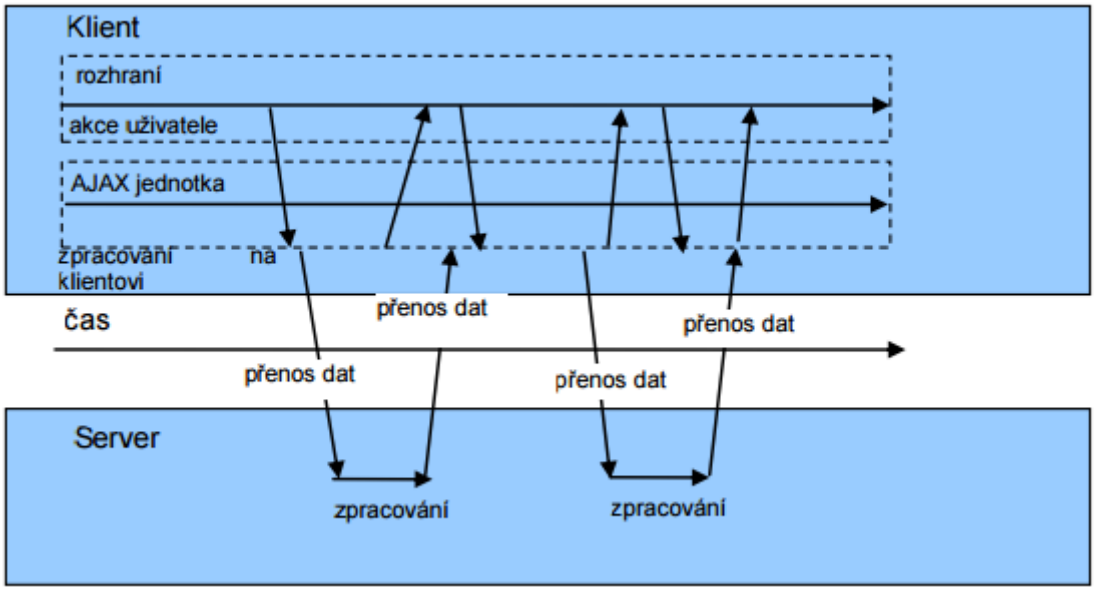
3.9 AJAX

Zkratka AJAX pochází z anglického Asynchronous Javascript And XML. AJAX, či nesoudobý Javascript je webová technologie umožňující lepší a interaktivní komunikaci s uživatelem. AJAX používáme především v případech, kdy potřebujeme změnit pouze část

uživatelského rozhraní. Může se jednat například o změnu obrázku, přepočítání a zobrazení nových hodnot či vytváření nových prvků. Následující dva obrázky demonstrují rozdíl mezi synchronní a asynchronní komunikací mezi klientem a serverem.



Obrázek 3.1: Synchronní komunikace mezi klientem a serverem [13]



Obrázek 3.2: Asynchronní komunikace mezi klientem a serverem [13]

Můžeme tedy vidět, že AJAX tedy nežádá o celou novou internetovou stránku, ale vytvoří datový kanál mezi klientem a serverem, po kterém server pošle příslušná data. Velikost dat, která posílá server, se obecně zmenší, protože nepřicházejí žádná redundantní data. V důsledku toho se také zkrátí doba potřebná pro odpověď serveru.

Kapitola 4

Návrh

Protože hlavním cílem této aplikace je minimalizovat úsilí, které je potřeba vynaložit na údržbu korektního chodu vyvíjených aplikací, je potřeba se při jejím vývoji zaměřit zejména na návrh, aby práce s ní nebyla přítěží.

Budoucí uživatele našeho systému si můžeme rozdělit do tří kategorií. První kategorií uživatelů jsou běžní koncoví uživatelé aplikací, kteří chtějí pomoci s testováním určité aplikace, nebo našli softwarovou chybu v aplikaci, kterou již používají. Tyto aktéry budeme dále označovat v textu jako veřejné uživatele. Vzhledem k faktu, že mezi veřejnými uživateli aplikací se mohou nacházet méně technicky zdatní, je vhodné, aby grafické rozhraní pro ně určené bylo velmi přehledné. To může v praxi znamenat například využití velkých tlačítek, fontů, výrazných barev či implementaci funkcí ulehčující práci s tímto systémem jako například funkce drag and drop. Druhou kategorií uživatelů jsou softwaroví vývojáři zodpovědní za danou aplikaci, tedy její vývoj a údržbu. Poslední kategorií jsou správci systému neboli administrátoři, kteří mají k dispozici všechnu funkcionalitu systému.

Při navrhování aplikace byly vytvořeny v jazyce UML (Unified Modeling Language) dva diagramy. Konkrétně diagram případů užití (Use-case diagram) viz obrázek 4.1 a Entity Relationship diagram (ERD) viz obrázek 4.2 Dále bylo také vytvořeno několik *mockupů* (obrázky 4.3 a 4.4). Mockupem máme na mysli obrázek, sloužící pro předběžnou ukázku vzhledu uživatelského rozhraní. Vytvoření mockupů nám umožnuje ujasnit si požadavky na vzhled grafického uživatelského rozhraní v etapě návrhu softwaru, což může vést k vytvoření kvalitnějšího systému poskytujícího efektivnější práci.

Všichni uživatelé tohoto systému by měli mít k dispozici tabuli Kanban, která bude vizuálně zobrazovat stav jednotlivých problémů a informace o nich. Použití Kanbanu umožní uživatelům systému lepší plánování a tudíž lepší využití zdrojů. Zpřístupnění tabule Kanban umožní běžným uživatelům nejen lepší přehled nad stavem jimi zadaných problémů, ale také lepší pochopení procesu testování a údržby softwaru, které by mělo vést k lepší spolupráci mezi veřejnými uživateli a vývojáři aplikací.

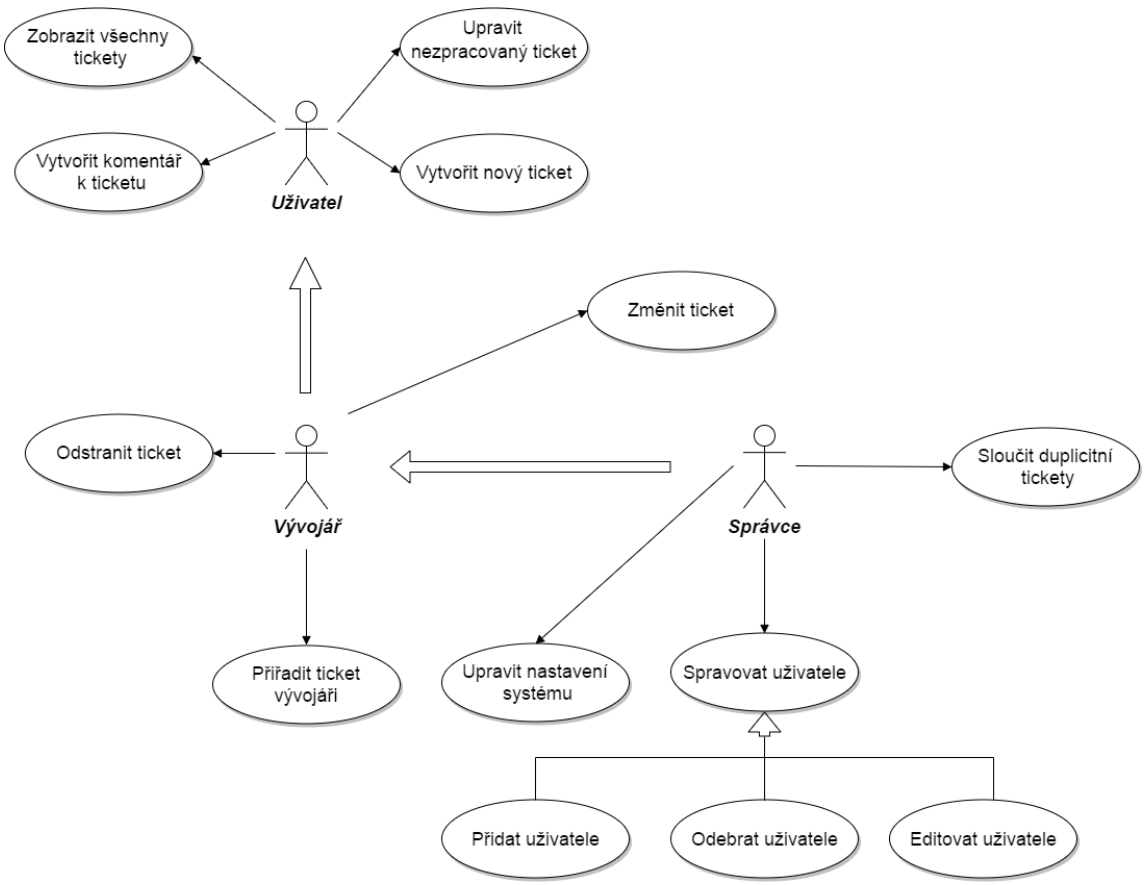
4.1 Use-case diagram

Use-case diagram neboli diagram užití je diagram chování systému definovaný v jazyce UML. Use-case diagram nám znázorňuje tyto věci:

- hranice navrhovaného/zkoumaného systému,
- účastníky analyzovaných případů užití,

- analyzované případy užití a
- interakce mezi aktéry a případy užití.

Na níže uvedeném diagramu (Obrázek 4.1) můžeme vidět akce, které bude systém umožňovat jednotlivým aktérům.

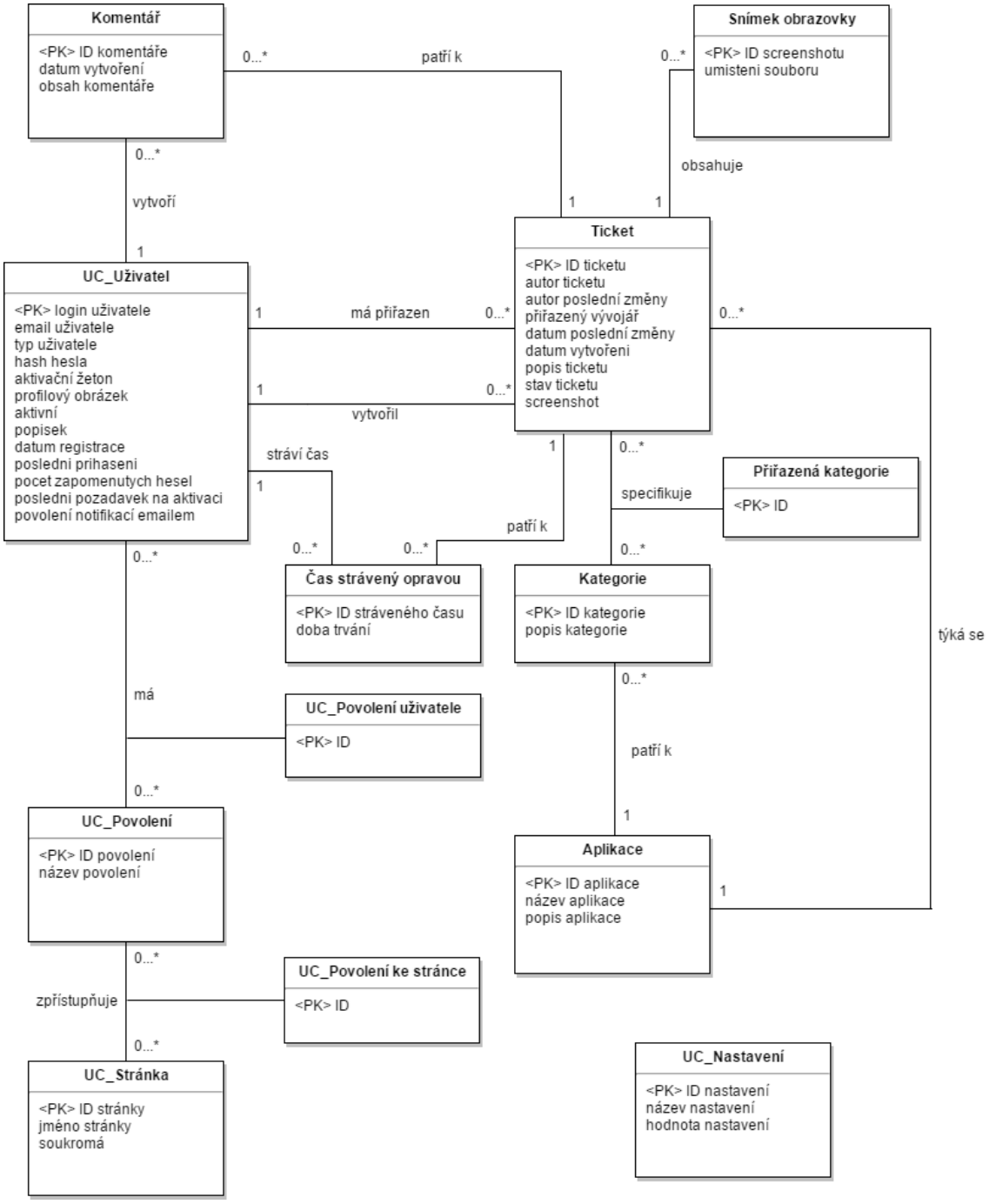


Obrázek 4.1: Usecase diagram

Na výše uvedeném diagramu případů užití máme tři různé aktéry. Těmito aktéry jsou správce *systému, uživatel* a *vývojář*, přičemž aktér *vývojář* je speciální variantou aktéra *uživatel* a aktér *správce* je specialní variantou aktéra *vývojář*. Můžeme vidět, že správce má k dispozici všechny funkce systému. Aktérem *uživatel* je myšlen veřejný uživatel jím testované či používané aplikace, který zadal požadavek na opravu chyby neboli ticket. Uvedený pojem ticket si můžeme definovat jako jedno konkrétní nahlášení chybného chování systému, o kterém máme vytvořený záznam v databázi.

4.2 Entity Relationship Diagram (ERD)

Při návrhu systému bylo také vhodné vytvořit model relační databáze pomocí Entity Relationship Diagramu (zkráceně ERD). ERD nám vizuálně zobrazuje jednotlivé entity v relační databázi a jejich vzájemné vztahy.



Obrázek 4.2: Entity Relationship Diagram

Hlavním prvkem systému tabulka *Ticket*. Řádek této tabulky reprezentuje jeden záznam nahlášení problému uživatelem. Jak vyplývá z diagramu, k ticketu lze díky zvláštní tabulce teoreticky možno přiložit neomezený počet snímků obrazovky. Stejně tak je k jednomu hlášení možné teoreticky přidat neomezený počet komentářů a kategorií. Tyto kategorie, jimiž lze klasifikovat jednotlivé tickety, jsou přiřazeny k určité aplikaci. Kategorizování problémů dá správci systému lepší přehled nad stavem testované aplikace a potencionálně umožní

efektivnější využití aplikace. Vzhledem k možné různorodosti nalezených problémů je toto vhodným řešením. Výše zmíněné komentáře mohou sloužit například pro komunikaci mezi uživatelem či testerem, který nahlásil nález problému v aplikaci, a vývojářem či správcem, kteří chybu řeší. Takováto komunikace může částečně eliminovat riziko vzniku nedorozumění, které by vedlo k plýtvání zdrojů.

Za pozornost také stojí tabulky s předponou *UC_*. Jedná se totiž o tabulky vytvořené použitým frameworkem UserCake. Tabulka *Stránka* reprezentuje všechny stránky systému, na které je možné se dostat. V této tabulce je kromě primárního klíče též uložen její název a informace o tom, zdali je stránka volně dostupná, nebo zdali vyžaduje přihlášení do systému. Samotnou kontrolu pak provádí výše zmíněný PHP framework userCake. Další je tabulka jednotlivých funkcí uživatelů v systému. Tuto tabulku si nazveme *Povolení*. Podle těchto povolení je pak rozhodováno, jaké funkce systému má uživatel k dispozici. K zprovoznění této funkcionality jsou však vyžadovány dvě vazební tabulky. První z nich přiřazuje jednotlivým uživatelům jejich oprávnění. Druhá tabulka definuje, která povolení mají přístup na které stránky. Poslední zatím nepopsanou stránkou je stránka *Nastavení*, do které jsou ukládány nejrůznější informace týkající se nastavení systému.

4.3 Návrh grafického rozhraní

V rámci návrhu bylo vytvořeno také několik mockupů pro lepší vizualizaci budoucího uživatelského rozhraní a upřesnění požadavků na výsledný systém.

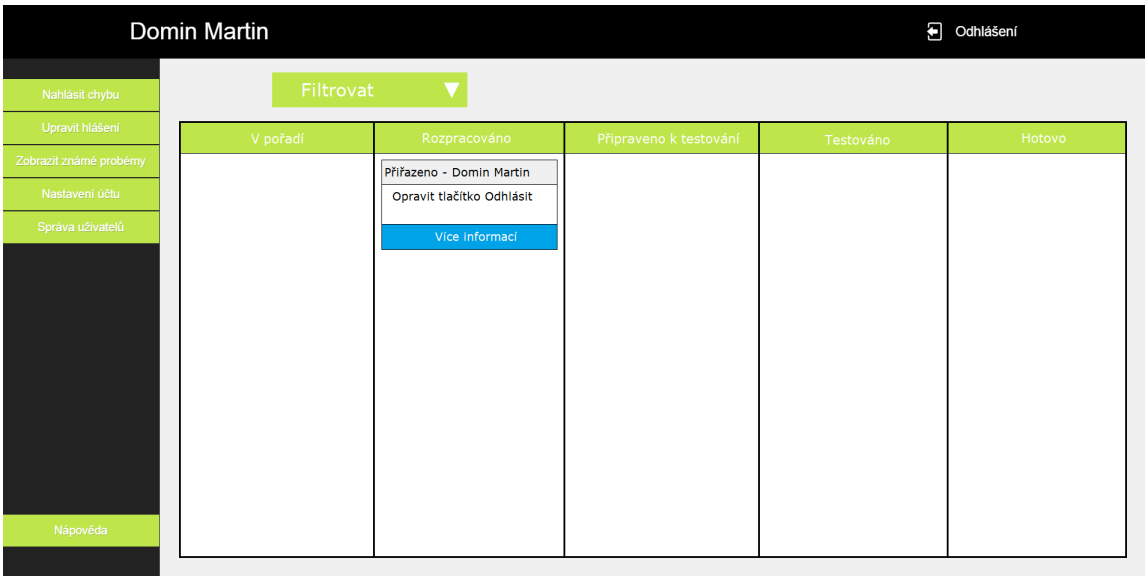
4.3.1 Nahlášení nalezených problémů

Obrázek 4.3: Mockup rozhraní pro hlášení chyb

Při návrhu stránky pro hlášení chyb je potřeba si ujasnit, jaké informace od uživatele vyžadujeme. Velmi důležitý je uživatelův slovní popis problému. Dále je pak také potřeba daný problém kategorizovat. Toto kategorizování problému je velmi užitečné pro vývojáře, protože umožňuje lepší přehled nad problémy. Nemělo by také být opomíjeno přidávání příloh, zejména tedy obrázků, které mohou v mnoha případech objasnit či upřesnit, kde se problém nachází. Samotná stránka by měla být velmi přehledná a proces hlášení co nejsnazší a nejrychlejší.

4.3.2 Tabule Kanban

Při navrhování části systému určené pro správu chybových hlášení je zejména potřeba se zaměřit na její hlavní prvek, kterým je tabule kanban. Pomocí kanbanu budou uživateli zobrazeny tickety podle jím nastaveného filtru. Mohl by si tedy vybrat, zdali chce aby se mu zobrazily všechny tickety, ty které mu byly přiřazeny k opravě či například jen ty, které spadají do určité kategorie, která jej zajímá. Veřejní uživatelé systému by si pak mohli zobrazit jimi zaslané tickety a vidět, jak probíhá jejich oprava. Vývojáři budou mít k dispozici funkce vztahující se k jednomu konkrétnímu ticketu po kliknutí na tlačítko *Více informací* vedoucí k detailu tohoto ticketu. V tomto detailu by vývojáři mohli měnit data náležící k příslušného ticketu, tedy například změnění stavu, jeho přiřazení vývojáři či přidání komentáře. Veřejným uživatelům by pak bylo umožněno pouze prohlížení detailů.

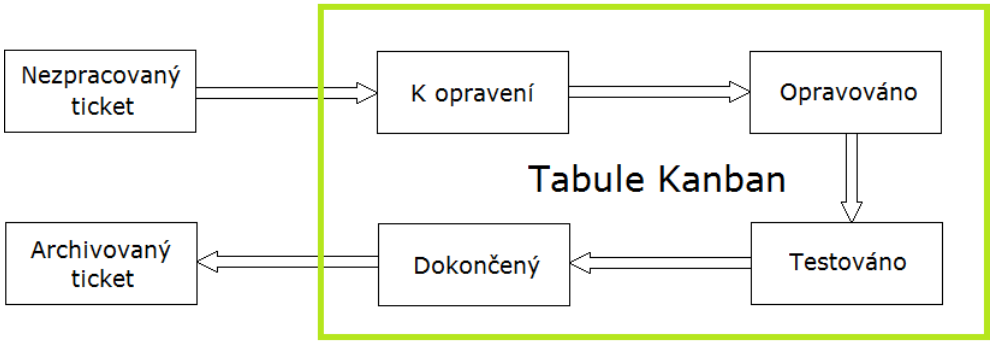


Obrázek 4.4: Mockup kanbanu našeho systému

4.3.3 Životní cyklus ticketu

Životní cyklus ticketu bude rozdělen do šesti částí. Tabule kanban pak bude zobrazovat ty tickety, které se nacházejí ve stavu *K opravení, Opravováno, Testováno a Dokončeno*. *Nezpracované* a *Archivované* tickety si bude možno zobrazit na oddělených stránkách.

1. **Nezpracovaný** - Ticket ještě nebyl zpracován správcem systému.
2. **K opravení** - Ticket byl již zpracován, ale práce na něm ještě nezapočala.
3. **Opravováno** - Chyba je momentálně opravována.
4. **Testováno** - Chyba byla opravena a je právě testována korektní funkcionalita.
5. **Dokončený** - Chyba byla opravena a správná funkcionalita ověřena.
6. **Archivovaný** - Aby se nám v sekci určené pro dokončené tickety nehromadily tickety, je možno je archivovat v případě, že není v našem zájmu je ihned odstranit z databáze.



Obrázek 4.5: Životní cyklus ticketu

Kapitola 5

Implementace

Tato část se věnuje popisu důležitých částí řešení jako jsou například vkládání hlášení do databáze, jejich zobrazení, filtrování a řazení, životní cyklus ticketu, přidávání aplikací, nápověda, rozesílání notifikačních emailů či autentifikace a autorizace.

5.1 Přihlášení a autentizace

Jak již bylo výše v textu zmíněno, autentifikaci a autentizaci implementuje použitý PHP framework UserCake. Uživatelem zadané údaje jsou zpracovány a porovnány se záznamem v databázi. Z bezpečnostních důvodů je ihned po odeslání formuláře použita nad uživatelovým heslem *hašovací funkce*. V databázi máme rovněž uložen pouze *hash*, či česky *otisk* uživatelova hesla, a tak porovnání probíhá pouze mezi samotnými otisky. Jako hašovací algoritmus byl použit algoritmus *MD5* implementovaný v jazyce PHP. MD5 neboli Message-Digest algorithm vytváří otisk o velikosti 128 bitů. Z bezpečnostních důvodu je k tomuto otisku navíc přidáno několik dalších bitů, které slouží jako doplňující vstup zvyšující bezpečnost. Těmto přidaným bitům říkáme sůl. Po úspěšném přihlášení je uživatel přesměrován na úvodní stránku systému a na pozadí jsou mu přidělena oprávnění přístupu k definovaným stránkám systému. Při neúspěšném přihlášení je uživateli oznámeno, že v databázi nebyl nalezen záznam o uživateli s zadaným jménem a heslem a je mu umožněno nové zadání informací.

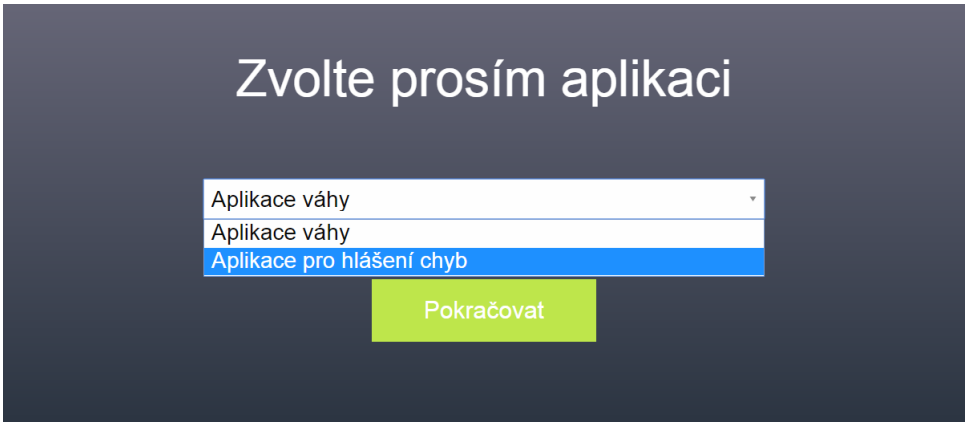
5.2 Uživatelské rozhraní

5.2.1 Hlášení chyb

Proces nahlášení chyby je rozdělen do celkem čtyřech částí, přičemž v každé části tohoto procesu uživatel vyplní určité informace o problému. Jednotlivé části formuláře jsou dynamicky generovány po stisknutí tlačítka *Pokračovat*. Již vyplněné sekce jsou po stisknutí tlačítka přepnuty do neaktivního režimu tudíž s nimi nelze pracovat, a jejich barva je změněna. Pokud se uživatel rozhodne, že by rád změnil již vyplněná data, může se vrátit k editaci předchozích sekcí tlačítkem *Zpět*.

1) **Volba aplikace**

   V první části je uživatel vyzván k tomu, aby určil, ve které aplikaci chybu nalezl. Uživateli se zobrazí seznam všech aplikací, které jsou uloženy v systému.



Obrázek 5.1: Zvolení aplikace, ve které je chyba

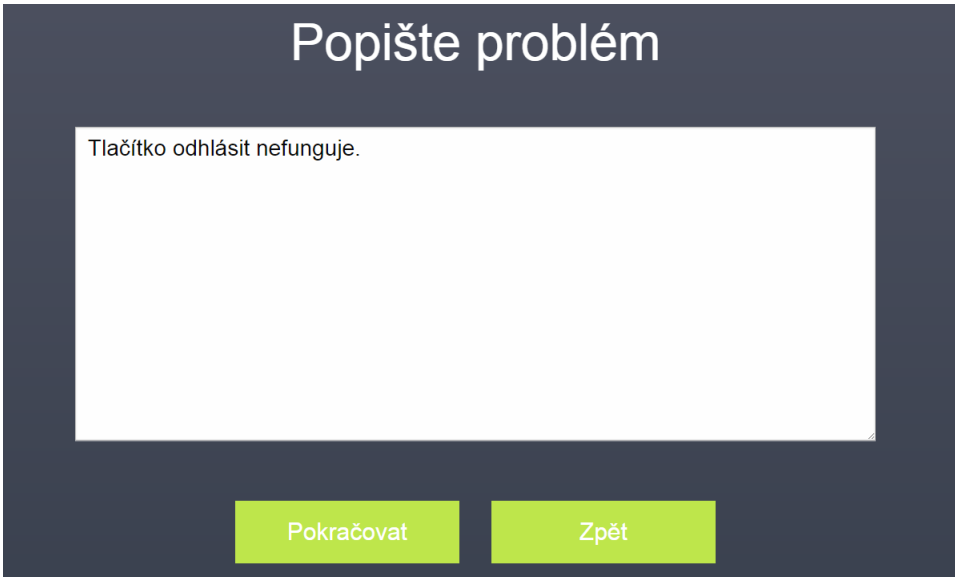
2) Kategorizace

Při pokračování do druhé části je zaslán pomocí technologie AJAX požadavek, který navrátí možné kategorie, kterými můžeme popsat nalezenou chybu. Z těchto kategorií si pak uživatel může vybrat ty, které nejlépe specifikují daný problém.

Obrázek 5.2: Kategorizace problému

3) Slovní popis

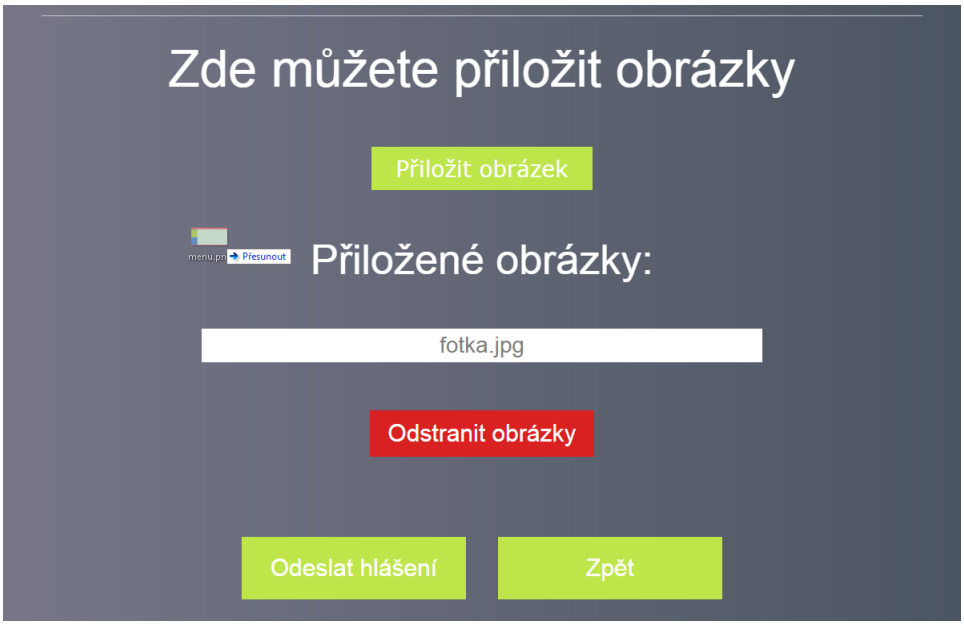
Ve třetí části hlášení je uživatel požádán o vlastní slovní popis problému. Ten je omezen na 1024 znaků.



Obrázek 5.3: Okno pro slovní popis problému

4) Přílohy

Poslední částí formuláře je sekce, kde může uživatel přiložit obrázky demonstrující nekorektní funkčnost systému. Také je uživateli umožněno v této sekci použít funkce drag and drop, tedy možnost přidání souborů přetáhnutím z počítače do prohlížeče. Samozřejmostí je pak také možnost odstranění přidaných obrázků.



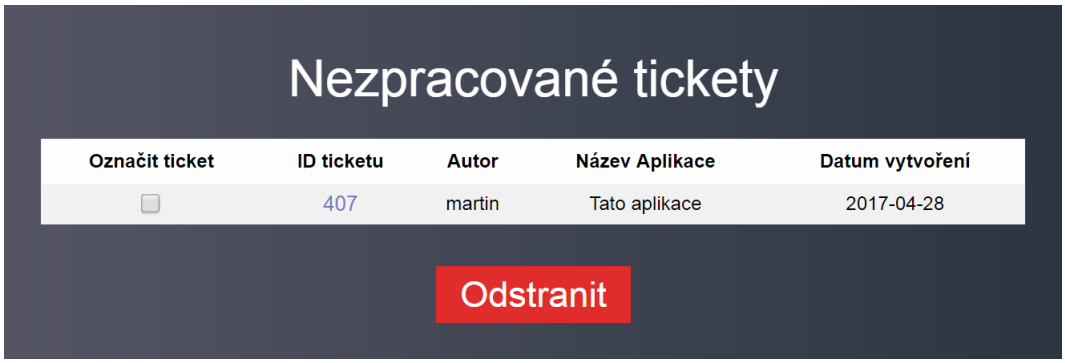
Obrázek 5.4: Přidávání obrázku pomocí funkcionality drag and drop

Pro standardní vkládání souborů jako příloh slouží *file input* element HTML formuláře. Jedná se o speciální tlačítko, po jehož stisku lze ručně vkládat do formuláře soubory, resp. cesty k souborům z lokálního úložiště klienta. Z bezpečnostních důvodů je toto jediný způsob vkládání souborů s použitím HTML formuláře a nelze jej dynamicky programově ovládat, aby nebylo možné získávat soubory klienta na pozadí bez jeho vědomí. K tomuto klasickému způsobu byla přidána ještě možnost vkládat soubory *drag and drop* způsobem. Využívá se zde *onDrop* událost formuláře, která reaguje na upuštění souboru do jeho plochy. Jeden z parametrů události je také odkaz k souboru. Z důvodu výše zmíněného bezpečnostního omezení, není možné tento odkaz programově přidat do stávajícího formuláře. Proto jsou takto vkládané soubory přidávány do dočasné proměnné a nahrány na pozadí použitím AJAX technologie až ve chvíli odeslání formuláře (*onSubmit* událost).

Před uložením obrázků do databáze je zkontrolován typ a velikost obrázku. Uživatelům je povoleno přidat pouze obrázky typu jpg, png a gif. Maximální velikost obrázku je omezena na 10MB a jejich počet je rovněž omezen, aby nedošlo k zahlcení serveru.

5.2.2 Nezpracované tickety

Všechny zaslané tickety jsou ihned po odeslání zařazeny do sekce *nezpracované*. Veřejní uživatelé mají možnost si prohlížet a případně upravit jimi vytvořené tickety, které se nacházejí v této sekci. Toto řešení tedy umožní uživatelům opravit své případné chyby a předejít nedorozumění. Jakmile však dojde ke zpracování ticketu administrátorem, tedy dojde ke změně stavu z *Nezpracovaný* na stav *K opravení* či *Opravováno*, není již veřejným uživatelům umožněno provádět jakékoliv změny nad tickety. Není žádoucí, aby byl ticket při své opravě libovolné pozměňován, jelikož by toto také mohlo vést ke vzniku dalších nedorozumění a nesrovnalostí. V případě, že uživatel chce upřesnit informace po zpracování ticketu, může k tomuto účelu použít diskuzi.

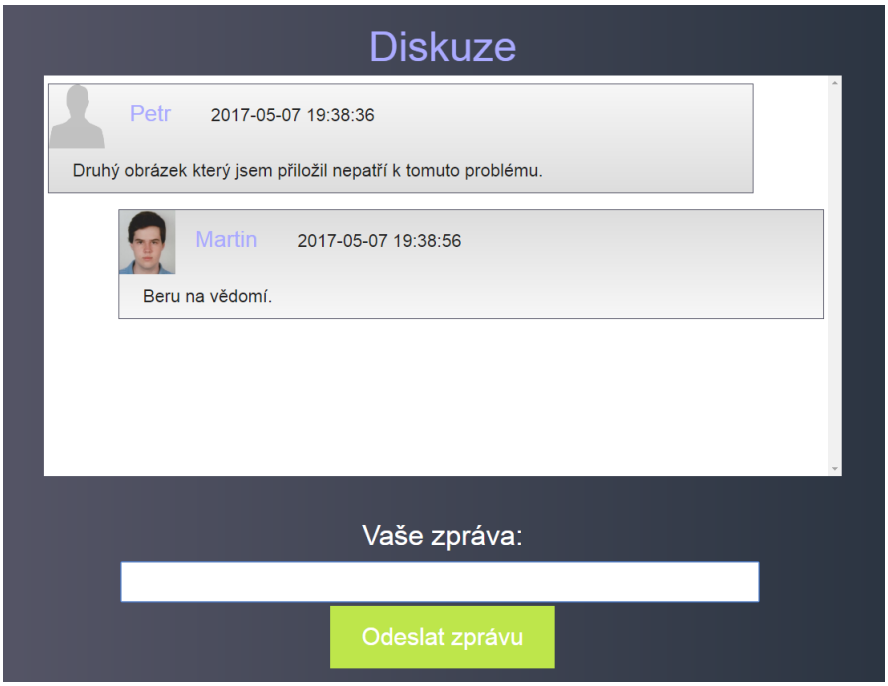


Obrázek 5.5: Seznam nezpracovaných ticketů

5.2.3 Diskuze

Ke každému ticketu je vytvořena diskuze. Pomocí ní mohou jednotliví uživatelé mezi sebou komunikovat. Zprávy jsou okamžitě po stisku tlačítka *Odeslat* odesílány na server, uloženy do databáze a znovu načteny pomocí asynchronního Javascriptu. Pro zobrazení uživatelem

právě odeslané zprávy pak není nutné načítat znovu celou stránku. Toto diskuzní okno lze najít v detailu ticketu.

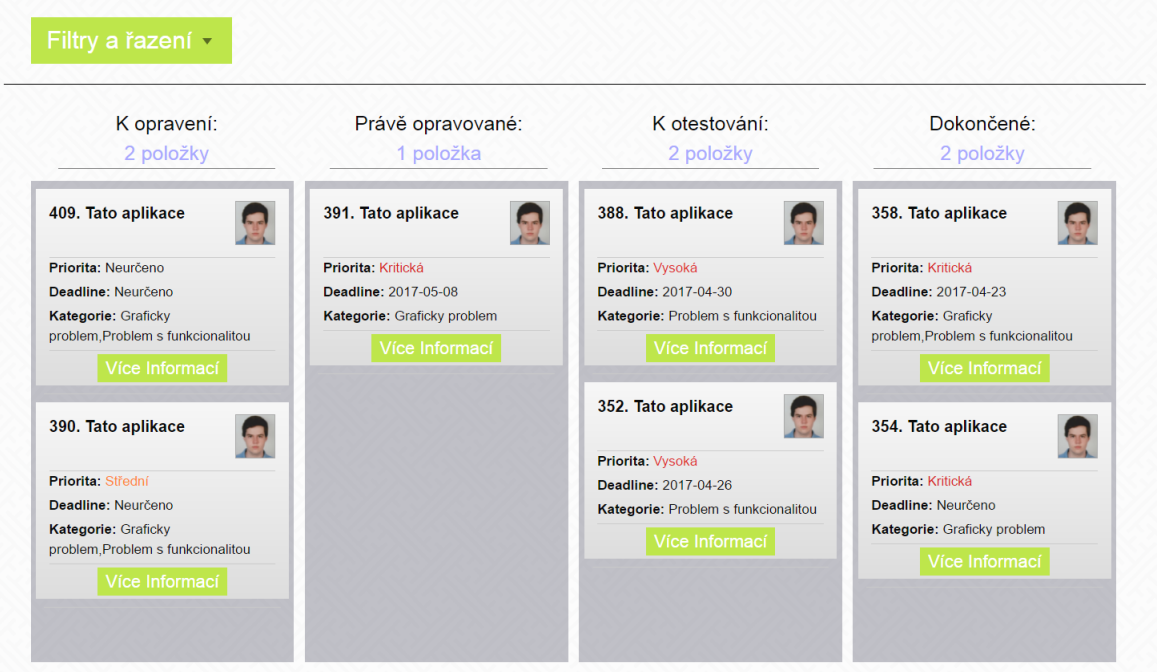


Obrázek 5.6: Diskuze k ticketu

5.2.4 Tabule kanban

Ty tickety, které jsou správcem systému zpracovány, jsou pak zobrazeny na tabuli Kanban. V našem případě je tabule kanban rozdělena do čtyř sloupců. V prvním sloupci jsou tickety, které již byly zpracovány, avšak práce na nich ještě nezapočala. V druhém sloupci se pak nacházejí ty tickety, na kterých se momentálně pracuje. Třetí sloupec obsahuje tickety, které jsou určeny k otestování. V případě, že by v této fázi byl nalezen další problém, lze ticket přeřadit zpátky do předchozí kategorie, dokud není řádně opraven. V posledním, tedy čtvrtém sloupci jsou pak vyřešené problémy. Aby se však v tomto sloupci nehromadily, je možno tickety v této kategorii přeřadit do kategorie *archivované*, či ze systému úplně odstranit.

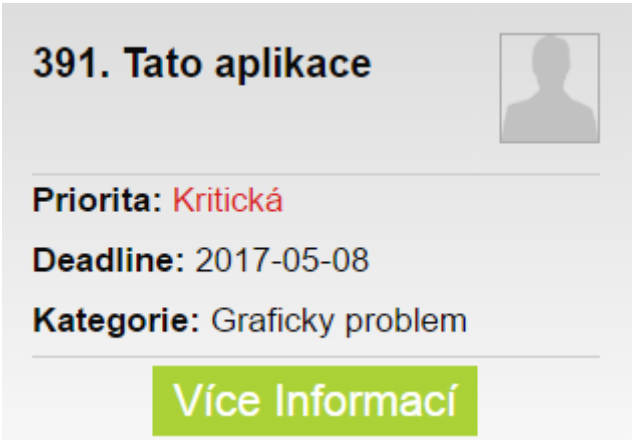
Při načtení stránky jsou z databáze načteny informace o všech uložených ticketech. Tyto tickety jsou pak dynamicky vytvářeny podle načtených informací pomocí Javascriptu a rozděleny do příslušných sloupců podle jejich stavu.



Obrázek 5.7: Demonstrace vzhledu tabule kanban

5.2.5 Náhled ticketu

V náhledu ticketu, který se nachází na tabuli kanban, jsou obsaženy základní informace. V levém horním rohu je vypsáno ID příslušného ticketu. Toto identifikační číslo je pak následováno názvem aplikace, ke které dané hlášení patří. V pravém horním rohu je pak zobrazen profilový obrázek vývojáře, jemuž bylo řešení přiřazeno. Níže jsou pak uvedeny základní informace, které je vhodné zobrazit již v náhledu. Ty informace, které se do náhledu nevejdou, jsou zobrazeny až po přechodu na detail ticketu. Tento detail je zobrazen po stisknutí tlačítka *Více informací*.



Obrázek 5.8: Vizualizace ticketu na tabuli Kanban

5.2.6 Detail ticketu

Prohlížení ticketů je umožněno všem přihlášeným uživatelům. Kromě režimu prohlížení je však pro vývojáře a správce systému dostupný i režim úprav. Tito uživatelé mají možnost libovolně přepínat mezi těmito režimy.

Režim úprav umožňuje nejen upravit informace zadané uživatelem, tedy například přiřazené kategorie či obrázky, ale také přiřazovat ticket vývojáři, změnit stav či nastavit prioritu nebo nejzazší datum opravy.

Systém také umožňuje nastavit časový odhad na opravu problému. Jednotliví uživatelé mají také možnost evidovat čas, který strávili opravováním tohoto problému. K dispozici je pak také Google graf, který ukazuje rozdíl mezi časovým odhadem na dobu potřebnou k opravě a reálnou hodnotou, vypočítanou sumou všech zaevidovaných strávených časů zaslaných jednotlivými uživateli.

U detailu každého ticketu je k dispozici galerie, obsahující uživatelem přiložené obrázky. Tato galerie se aktivuje po kliknutí na libovolnou miniaturu obrázku. Systém využívá otevřeného softwaru prettyPhoto. Jedná se o jednoduchou, rychle nastavitelnou galerii podporující nejen prohlížení obrázku, ale také videí. V našem systému však přikládání videí implementováno nebylo, a tak je využíváno pouze prohlížení obrázků. Galerie umožňuje jak manuální, tak automatické listování mezi obrázky či maximalizaci na plnou velikost.

Obrázek 5.9: Galerie prettyPhoto

5.2.7 Filtrování a řazení ticketů

Ve výchozím nastavení jsou na tabuli kanban vypsány všechny tickety, které jsou seřazeny sestupně podle data vytvoření. Toto nastavení však nemusí byt vždy nejvhodnější. Z tohoto důvodu byla do systému přidána možnost filtrování a řazení.

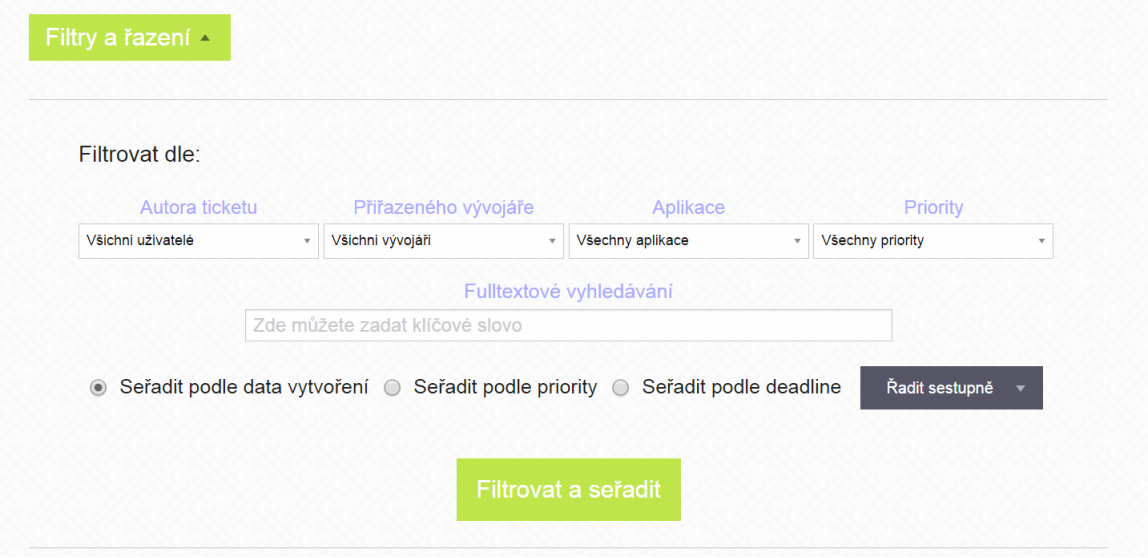
Systém umožňuje použití několika základních přednastavených filtrů, které je filtrovat dle následujících kritérií.

- Jejich autora
- Přiřazeného vývojáře
- Aplikace jejíž chybné chování dokumentují
- Priority

Krom těchto filtrů je též možno použít filtrování pomocí klíčového slova. Klíčové slovo, číslo či řetězec, který uživatel zadá, je pak vyhledáván mezi:

- Identifikátory ticketů
- Názvy aplikací
- Daty vytvoření
- Daty nejpozdějšího termínu opravy
- Uživateli, kteří jsou odpovědní za opravu
- Autory ticketů
- Kategoriemi
- Autorově popisu problému

Tyto filtry je též následně možno kombinovat s vybraným řazením. Tickety lze řadit podle data vytvoření, priority či data nejpozdějšího termínu opravy. Uživatel si také může vybrat, zdali chce, aby se tickety podle seřazené hodnoty řadily vzestupně či sestupně.



Obrázek 5.10: Filtrování a řazení

5.2.8 Správa aplikací

Při vyplňování hlášení je vyžadováno, aby uživatel specifikoval, ke které aplikaci se nalezená chyba vztahuje. Správce systému tedy musí vytvořit v systému seznam těch aplikacích, jejichž chyby chce pomocí tohoto systému spravovat. Při vytváření nového záznamu o aplikaci je potřeba vyplnit název aplikace, její popis a také vypsat jednotlivé kategorie problému, kterými může uživatel klasifikovat problém. Položky kategorií lze podle potřeby libovolně přidávat či odebírat. Při odeslání vyplněného formuláře jsou odstraněny případné duplicitní kategorie a je vytvořen záznam v databázi. Takto vytvořené aplikace lze pak následně libovolně upravovat.

5.2.9 Emailové notifikace

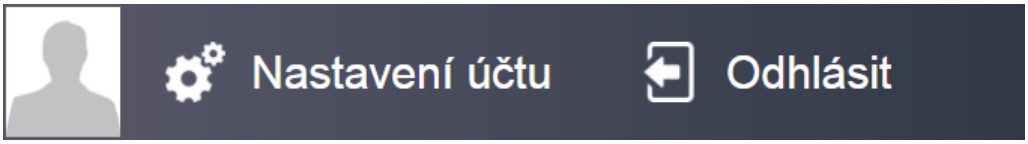
V systému byly implementovány notifikace přes elektronickou poštu. Jednotliví uživatelé si mohou vybrat, jestli chtějí tyto notifikace přijímat či nikoliv. Uživateli je zaslána notifikace v následujících případech.

- Byly pozměněny údaje jeho účtu
- Byl změněn stav ticketu který vytvořil
- Byl změněn obsah ticketu který vytvořil
- Byl mu přiřazen k vyřešení ticket
- Byl změněn stav ticketu který mu byl přiřazen k vyřešení
- Byl změněn obsah ticketu který mu byl přiřazen k vyřešeníl
- Byl k ticketu který vytvořil přidán komentář
- Byl k ticketu který mu byl přiřazen k vyřešení přidán komentář
- Uživatel je správce systému a byl vytvořen nový účet vyžadující aktivaci
- Ticket který vytvořil byl sloučen s jiným ticketem
- Ticket který mu byl přiřazen byl sloučen s jiným ticketem

5.2.10 Účty a jejich správa

Uživatelé se mohou libovolně registrovat, avšak přihlášení do systému je možné až po aktivaci účtu správcem. Tímto zajistíme, že budou se systémem pracovat pouze ti uživatelé o kterých víme, že se chtějí podílet na vývoji a opravách aplikací. Účty však může správce systému vytvářet i sám.

Při registraci si uživatel zvolí své uživatelské jméno, heslo a zadá kontaktní email. Nově vytvořené účty získávají po aktivaci oprávnění *Veřejný uživatel*, které povoluje hlášení a zobrazení ticketů a správu vlastního účtu. Uživatelům je umožněno si měnit heslo, kontaktní email či profilový obrázek. Pro jakékoliv změny je však potřeba zadat aktuální heslo. Profilový obrázek je uživatelům zobrazen na horní liště. Ta obsahuje přechod na správu účtu a odhlášení. Tuto lištu je možno skrýt kliknutím na profilový obrázek.



Obrázek 5.11: Horní lišta

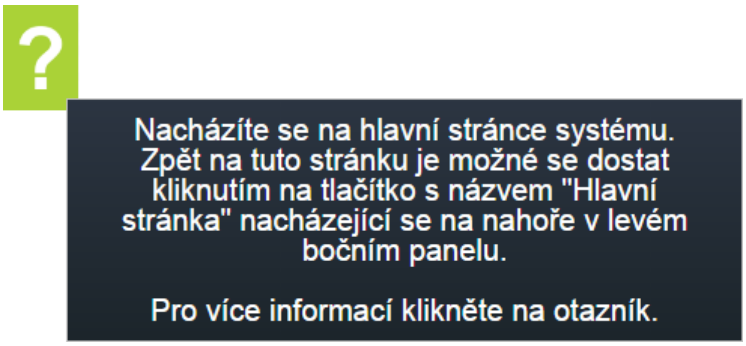
Správce systému může také uživatele libovolně odstraňovat. Může také upravit uživatelův kontaktní email či upravit jeho přístupová práva.

5.2.11 Odstranění duplicitních ticketů

Administrátor systému má možnost odstranit duplicitní tickety. Tento proces je rozdělen do dvou částí. V první části vybere, které tickety by rád sloučil. Počet ticketů ke sloučení může být libovolný, musí však být alespoň dva. V druhé části tohoto procesu je pak vyzván aby určil, který ticket by chtěl z duplicitních ponechat. Ostatní tickety jsou poté smazány a jejich autoři a přiřazení vývojáři o této změně notifikováni.

5.2.12 Nápověda

Každému uživateli je na každé stránce k dispozici nápověda, která se zobrazí po přejetí kurzorem nad speciálním tlačítkem nacházejícím se v levém horním rohu obrazovky. Uživateli jsou pak zobrazeny základní informace o aktuální stránce, tedy jaké akce je na ní možno provést a jak. Po kliknutí na toto tlačítko je uživatel přesměrován na stránku nápovědy pro celou aplikaci.



Obrázek 5.12: Zobrazení nápovědy

Kapitola 6

Porovnání s dostupnými issue tracking systémy

Issue tracking systémy se v dnešní době nedílnou součástí životního cyklu softwaru. Z tohoto důvodu je jich dnes k dispozici velké množství. Bohužel však mnoho těchto systémů není implementováno jako webové aplikace. Z tohoto důvodu pak aplikace vyžadují instalaci a nastavení, což může mnohé uživatele od hlášení chyb odradit. Velké množství issue tracking systémů je také velmi složitých, poskytujících mnoho funkcionalit, které nemusí být pro malé firmy vůbec potřebné. Z tohoto důvodu jsou pak tyto systémy velice drahé. Veliké množství dostupných funkcí může být navíc pro mnoho uživatelů matoucí a odrazující. Níže jsou uvedeny některé příklady issue tracking systémů a výhody či nevýhody jejich použití oproti této aplikaci.

6.1 Atlassian JIRA

Software *JIRA* vyvíjený společností Atlassian je pravděpodobně dnes nejrozšířenější issue tracking systém. Jedná se o velice kvalitní systém poskytující jeho uživatelům mnoho funkcionalit. Systém *JIRA* lze také velmi dobře upravit tak, aby vyhovoval specifickým požadavkům jednotlivých firem. Hlavní nevýhodou tohoto systému je však jeho cena. Ta se odvíjí od počtu uživatelů, kteří budou tento produkt používat. V případě tedy, že bychom do procesu vývoje chtěli zapojit co nejvíce veřejných uživatelů, byl by provoz této aplikace velmi drahý. Pro ilustraci uvedeme ceník dostupný na oficiálních stránkách. [4]

JIRA Software

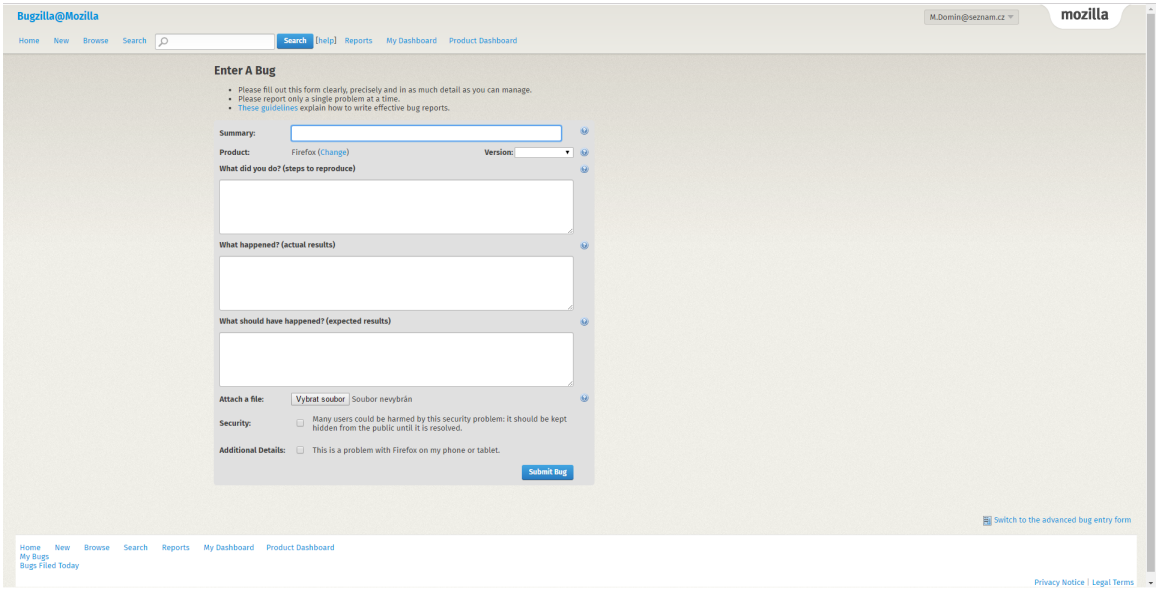
Počet uživatelů	Cena (USD)
1 - 10	10
11 - 25	1 800
26 - 50	3 300
51 - 100	6 000
více	kontaktujte nás

Obrázek 6.1: Ceník softwaru JIRA [4]

K systému *JIRA* je možno navíc dokupovat rozšiřující moduly, které dále rošiřují funkcionalitu systému. Například je k dispozici možnost napojení systému na Git, což u daného ticketu zobrazí provedené změny v kódu.

6.2 Bugzilla

Bugzilla [5] je systém původně vyvinutý a používaný organizací *Mozilla*, dostupný pod licencí *Mozilla Public Licence*. Tento systém se do jisté míry podobá systému vytvořeném v této práci. Oba systémy se primárně zaměřují na to, aby jeho uživatelům poskytly základní funckionalitu, tedy hlášení a správu chyb. Vzhledem k dlouholetému vývoji je velmi pravděpodobné, že tento systém je již dobře odladěn. Na systému je však patrné, že jeho grafické uživatelské rozhraní je zastaralé a nedodržuje moderní zásady webdesignu. Toto lze vidět na obrázku Bugzilla 6.2, kde je ukázáno hlášení problému v aplikaci *Firefox* od společnosti *Mozilla*. Systém také bez rozšíření nepodporuje například funkci drag and drop či zobrazení ticketů pomocí tabule Kanban.



Obrázek 6.2: Hlášení chyb v systému Bugzilla [5] (rozlišení monitoru 1920 x 1080 px)

Kapitola 7

Testování

Součástí vývoje tohoto systému bylo také jeho otestování v prostředí malé firmy. Toto testování bylo však možné provést až tehdy, kdy byla aplikace téměř dokončena a poskytovala většinu své konečné funkcionality. Bylo tedy potřebné aplikaci řádně testovat ještě před jejím nasazením, tedy během jejího vývoje.

7.1 Vlastní testování

Díky zaměření této aplikace bylo možné velmi brzy v procesu jejího vývoje ji využít samu na sebe. Jakmile tedy bylo zprovozněno základní hlášení chyb a jejich správa, byly nejen jednotlivé nalezené problémy, ale také potřebné funkcionality potřebné k doplnění ukládány do systému. Toto umožnilo přehledné řízení jejího vývoje. Takovéto užívání navíc odhalilo mnoho problémů, které bylo možno následně spravovat. V rámci vývoje této aplikace bylo vytvořeno přes 400 ticketů, které dokumentovaly chybnou či chybějící funkcionalitu.

7.2 Testování v malé firmě

Vytvořený systém byl nasazen ve firmě EDIacademy, s.r.o. [1]. V této firmě byl použit k otestování právě dokončovaných aplikací *Váhy* a *Zvířátka dědy Lesoně*.

7.2.1 EDIacademy s.r.o.

EDIacademy, s.r.o. je společnost, která byla založena v roce 2015. Zaměřuje se na vývoj didaktických materiálů pro výuku matematiky. Společnost spolupracuje s Ostravskou Univerzitou a Univerzitou Komenského v Bratislavě, kde za spolupráce s fakultními základními školami provádí výzkum a testování nově vznikajících elektronických pomůcek pro výuku. Projekt si klade za cíl rozvíjení klíčových kompetencí žáků základních škol k řešení problémů v oblasti kritického a logického myšlení a matematiky. K rozvoji jedince a jeho schopností se používají moderní technologie, které žáka vedou k samostatnému řešení problémů, užívání logického myšlení, práci s chybou, ověřování správnosti řešení a dalším dovednostem, které na konci základního vzdělání dosáhne. Pomůcky vyvíjené EDIacademy, s.r.o. využívají Hejného výukovou metodu [2]. Hejného výuková metoda je netradiční způsob výuky matematiky. Metodu zavedl profesor Milan Hejný za účelem vytvoření experimentálních úloh, které pomáhají studentům lépe porozumět matematickým a logickým problémům. M. Hejný založil společnost H-mat, o.p.s., se kterou společnost EDIacademy, s.r.o. od srpna 2015 spolupracuje.

7.2.2 Zhodnocení testování

Na testování se podíleli dva vývojáři a jeden tester. Tester kontroloval aplikace *Váhy* a *Zvířátka dědy Leoně*, jestli odpovídají navržené specifikaci a zda neobsahují funkční chyby. Nalezené nedostatky pak zadával do systému jako tickety. Ty poté přiřazoval buďto vývojáři zodpovědnému za aplikaci *Váhy*, nebo vývojáři zodpovědnému za aplikaci *Zvířátka dědy Lesoně*. Vývojáři měli nově zadané tickety k dispozici na kanbanu v kategorii *K opravení*. Vývojáři buď dle zadání chyby opravili, nebo dále diskutovali se zadávajícím v diskuzi na daném ticketu. Po dokončení opravy byly tickety označeny jako *K otestování* a testerem zkontrolovány. V případě úspěšného testu tester ticket uzavřel jako *Dokončený*. V případě neúspěchu, ticket vrátil do stavu *K opravení* a tedy ticket vrátil vývojáři.

Využití této aplikace usnadnilo a urychlilo poslední, testovací fázi na projektech. I díky tomu byly aplikace dokončeny včas. Zpětná vazba byla pozitivní a se systémem se údajně snadno pracovalo a byl považován za intuitivní. Nicméně bylo vzneseno také několik požadavků na dodání nových funkcionalit nebo vylepšení stávajících. Například jedním z požadavků bylo připravení emailových notifikací při vytvoření a přiřazení nového ticketu nebo při změně jeho stavu. Tato funkcionalita byla následně dodána. Testování systému bylo úspěšné, ale bohužel ho nebylo z časových důvodů možno otestovat tzv. veřejnými uživateli, kterými měli být v tomto případě učitelé. Tato fáze testu je naplánovaná až na měsíc září, a není tedy možno zahrout výsledky tohoto testování do této práce.

Kapitola 8

Závěr

Cílem této bakalářské práce bylo seznámit se s problematikou tvorby softwaru, konkrétně se zaměřením na procesy testování aplikací a zpracování hlášení o chybách a seznámit se s existujícími dostupnými nástroji, určenými pro tuto problematiku. Na základě těchto poznatků pak bylo úkolem navrhnout a vytvořit informační systém pro malou firmu, který by umožňoval hlášení a správu chyb nalezených v aplikacích vyvíjených touto firmou. Důležitým požadavkem bylo umožnit veřejným uživatelům aplikací vyvíjených malou firmou nahlášení jimi nalezených problémů a tak je tedy zapojit do procesu vývoje softwaru.

V rámci této práce byl na základě poznatků získaných při zkoumání dostupných issue tracking systémů vytvořen vlastní informační systém umožňující hlášení a zprávu chyb.

Systém byl během svého vývoje též aplikován na sebe, což umožnilo řízený vývoj samotné aplikace. Posledním bodem zadání byl požadavek na otestování navrženého softwaru v prostředí malé firmy. Tento cíl byl splněn a systém byl nasazen ve firmě EDIacademy s.r.o.. Nasazení systému v této firmě nejen urychlilo a zlepšilo testování právě vyvíjených aplikací, ale také pomohlo vylepšit tuto aplikaci odhalením některých skrytých chyb a přidáním nových funkcionalit. Do tohoto testovaní bohužel však nemohli být zapojeni veřejní uživatelé, a tak byl systém otestován pouze interními pracovníky.

8.1 Možná vylepšení

Byl vytvořen systém umožňující jednoduché a efektivní hlášení a správu problémů. K systému by však bylo možno přidat mnohá vylepšení, která by usnadňovala jeho použití nebo například zlepšila zabezpečení. Pro veřejné uživatele webových aplikací by bylo například vhodné, kdyby se samotnou aplikací byl rovněž dostupný zásuvný modul dostupný pro běžně používané prohlížeče, který by umožnil ještě rychlejší hlášení problémů. Uživatel by pak mohl nahlásit problém nalezený v aplikaci velmi rychle a pohodlně. Dalším možným vylepšením by pak mohlo být například vylepšení filtrování ticketů na tabuli kanban. V průběhu vývoje této aplikace byla též ukončena podpora použitého PHP frameworku userCake, a tak by mohlo být vhodné jej nahradit modernějším a vyspělejším frameworkem.

Literatura

[1] *EDIacademy, s.r.o.* [Online].
URL http://www.ediacademy.cz/

[2] *Hejného metoda.* [Online].
URL http://www.h-mat.cz/hejneho-metoda

[3] *UserCake Framework.* [Online].
URL http://usercake.com/

[4] *JIRA software.* Atlassian, [Online].
URL http://www.myjira.cz/produkty/jira-software.html

[5] *Bugzilla@Mozilla.* Bugzilla, [Online].
URL https://bugzilla.mozilla.org/

[6] jQuery Community Experts: *jQuery Cookbook.* O' Reilly, 2010, ISBN 978-0-596-15977-1.

[7] Flanagan, D.: *JavaScript The Definitive Guide.* O' Reilly, 2011, ISBN 978-0-596-80552-4.

[8] Henick, B.: *HTML & CSS The Good Parts.* O' Reilly, 2010, ISBN 978-0-596-15760-9.

[9] Hruška, T.: *Internetové aplikace Část Programování klienta (JavaScript) Studijní opora.* FIT VUT v Brně, Únor 2012.
URL https://www.fit.vutbr.cz/study/courses/WAP/private/opory/OporaWAP6ProgramovaniKlienta.pdf

[10] Hruška, T.; Burget, R.: *Internetové aplikace Část Programování serveru Studijní opora.* FIT VUT v Brně, Únor 2012.
URL https://www.fit.vutbr.cz/study/courses/WAP/private/opory/OporaWAP4ProgramovaniServeru.pdf

[11] Hruška, T.; Křivka, Z.: *Informační systémy Studijní opora.* FIT VUT v Brně, Únor 2012.
URL https://wis.fit.vutbr.cz/FIT/st/course-files-st.php?file=%2Fcourse%2FIUS-IT%2Ftexts%2FIUS_opora.pdf&cid=10027

[12] Křena, B.; Kočí, R.: *Úvod do softwarového inženýrství IUS Studijní opora.* FIT VUT v Brně, Prosinec 2010.
URL https://wis.fit.vutbr.cz/FIT/st/course-files-st.php?file=%2Fcourse%2FIUS-IT%2Ftexts%2FIUS_opora.pdf&cid=10027

[13] Máčel, L.; Kuželka, A.; Hruška, T.: *Internetové aplikace Část AJAX Studijní opora*. FIT VUT v Brně, Únor 2012.
URL https://www.fit.vutbr.cz/study/courses/WAP/private/opory/OporaWAP5Ajax.pdf

[14] Máčel, L.; Kuželka, A.; Hruška, T.: *Internetové aplikace Část SGML, HTML, CSS, DOM Studijní opora*. FIT VUT v Brně, Únor 2012.
URL https://www.fit.vutbr.cz/study/courses/WAP/private/opory/OporaWAP2SGMLHTMLCSSDOM.pdf

[15] Schwarz, B.; Zaitsev, P.; Tkachenko, V.: *High Performance MySQL*. O' Reilly, 2012, ISBN 978-1-449-31428-6.

[16] Sklar, D.; Trachtenberg, A.: *PHP Cookbook*. O' Reilly, 2014, ISBN 978-1-449-36375-8.

[17] *PHP Documentation*. W3schools, [Online].
URL http://php.net/docs.php

[18] *W3Schools*. World Wide Web Consortium, [Online].
URL https://www.w3schools.com

[19] *Foundation framework*. ZURB, [Online].
URL http://foundation.zurb.com/

Přílohy

Příloha A

Zprovoznění systému

Ke zprovoznění systému je potřeba mít k dispozici server. Testování systému proběhlo na operačních systémemech Widnows s HTTP serverem Apache 2.4.23 a Linux Gentoo s HTTP serverem Apache 2.4.10. Je tedy doporučeno použít stejný operační systém a HTTP server, avšak aplikace by měla fungovat i na kompatibilních platformách. Také je potřeba mít k dipozici databázový server typu MySQL 5 InnoDB a podporu PHP ve verzi alespoň 5.3.

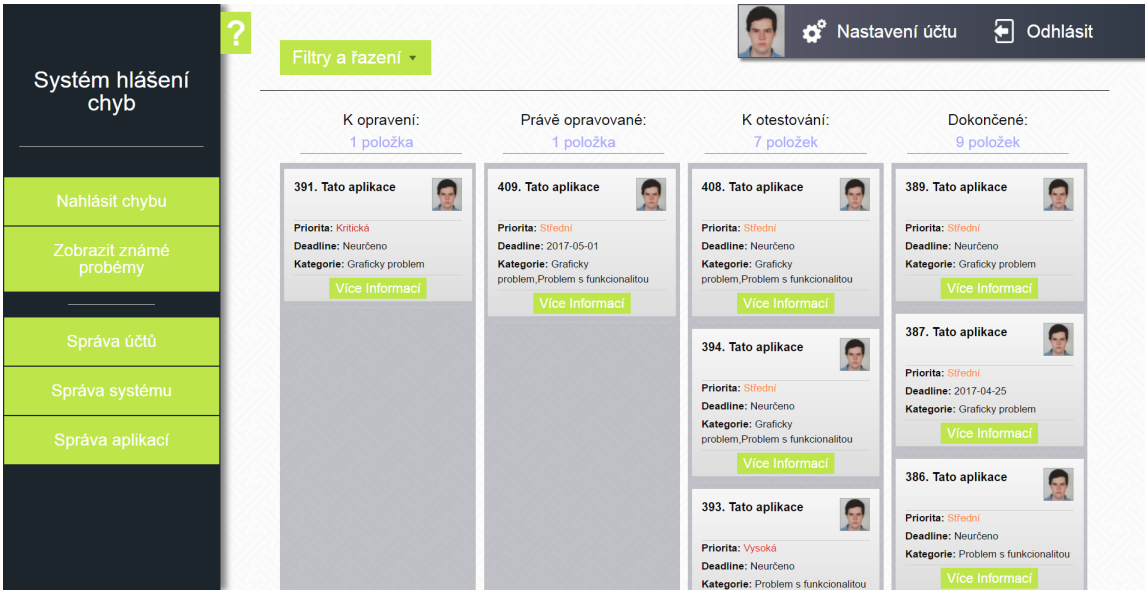
Při zprovozňování systému je potřeba nejprve vytvořit novou databázi. Údaje potřebné k přihlášení do databáze je pak potřeba vyplnit do souboru db-settings.php, který se nachází ve složce models na přiloženém CD. Spuštění či import SQL skriptu *createDatabase.sql*, který se nachází ve složce *SQL*, zajistí vytvoření tabulek a dat potřebných k správnému chodu systému. Poté je potřeba vytvořit DNS záznam směruje dotazy do složky s aplikací. Posledním krokem je nastavení MX záznamu na emailový sever, který bude používán k odesílání notifikačních emailů.

Skript vytvoří jeden správcovský účet s jménem *admin* a heslem *admin123*. Po přihlášení je pak možno změnit heslo, přidat nové uživatele či pozměnit nastavení systému jako například emailovou adresu či jiné.

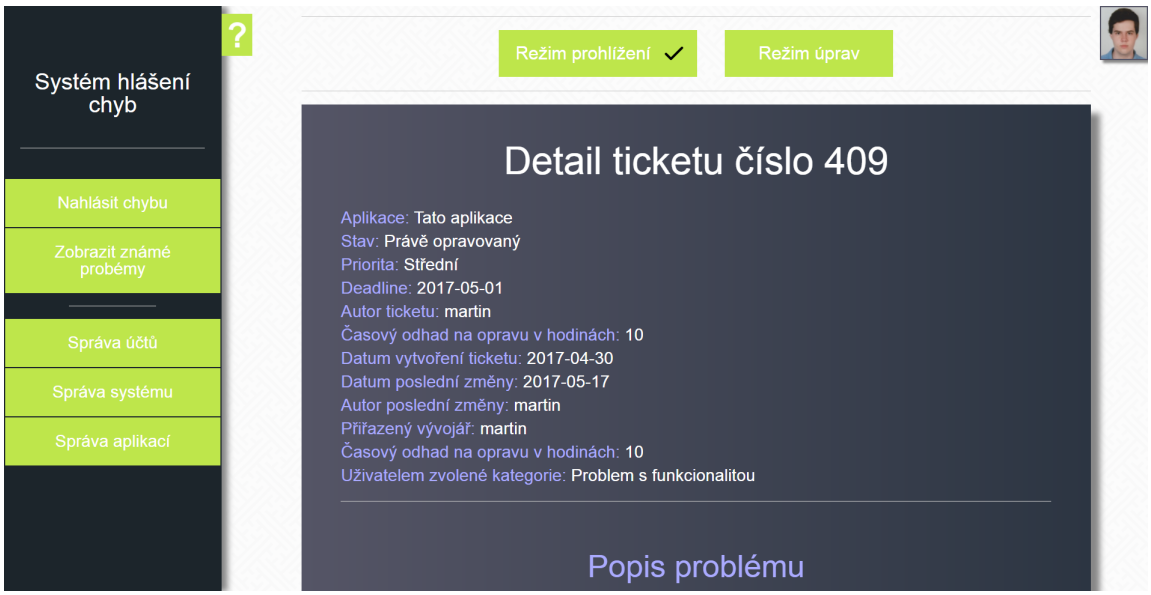
Příloha B

Další ukázky vzhledu

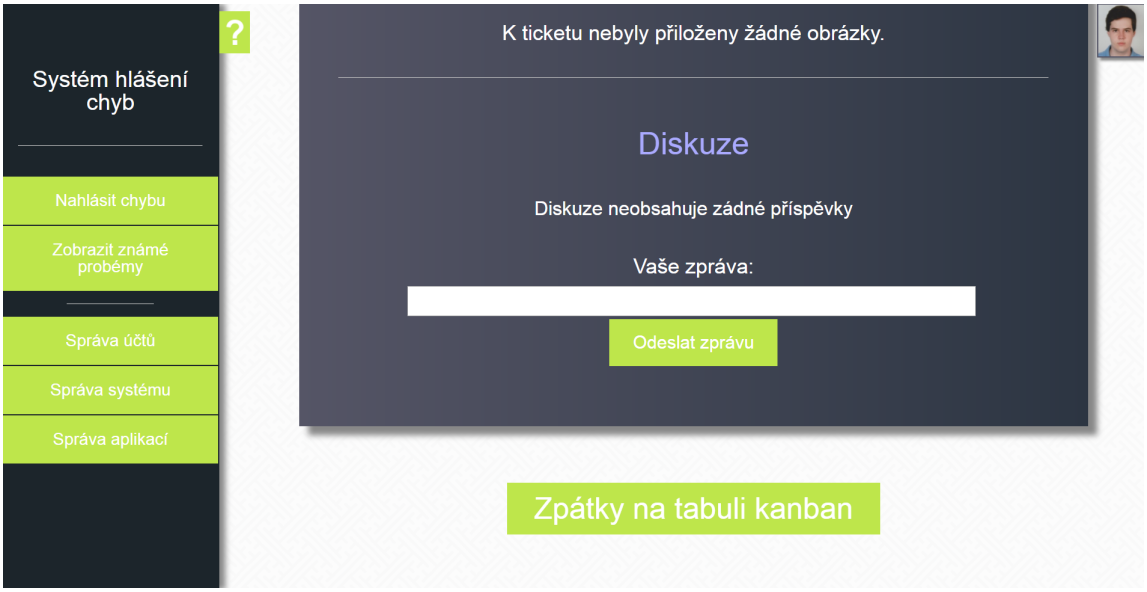
Obrázek B.1: Přihlášení do systému



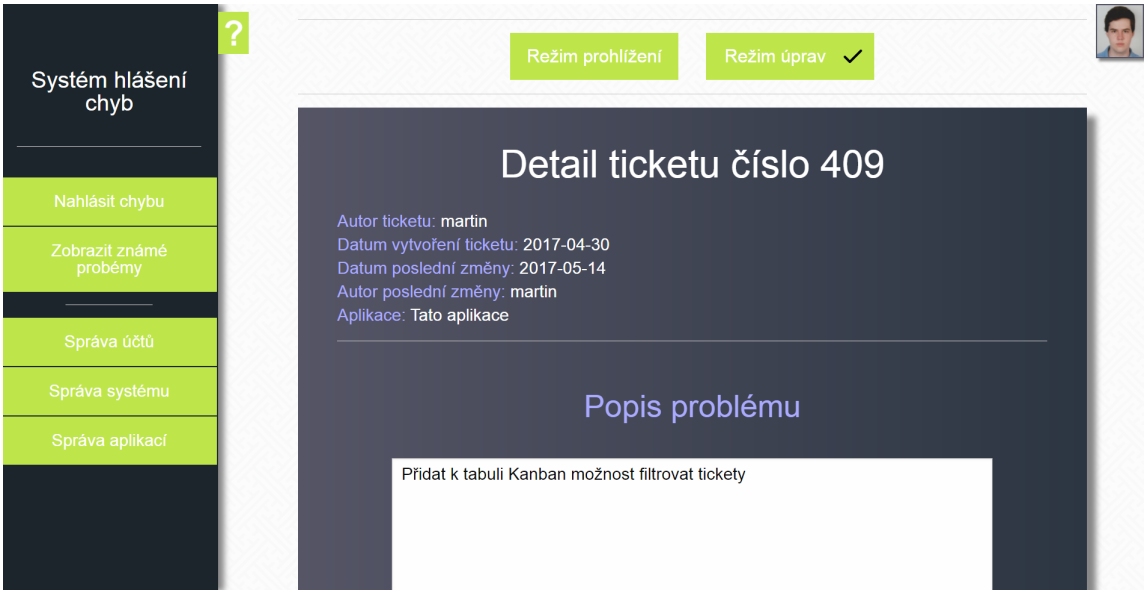
Obrázek B.2: Tabule Kanban



Obrázek B.3: Detail ticketu s režimem prohlížení část první



Obrázek B.4: Detail ticketu s režimem prohlížení část druhá

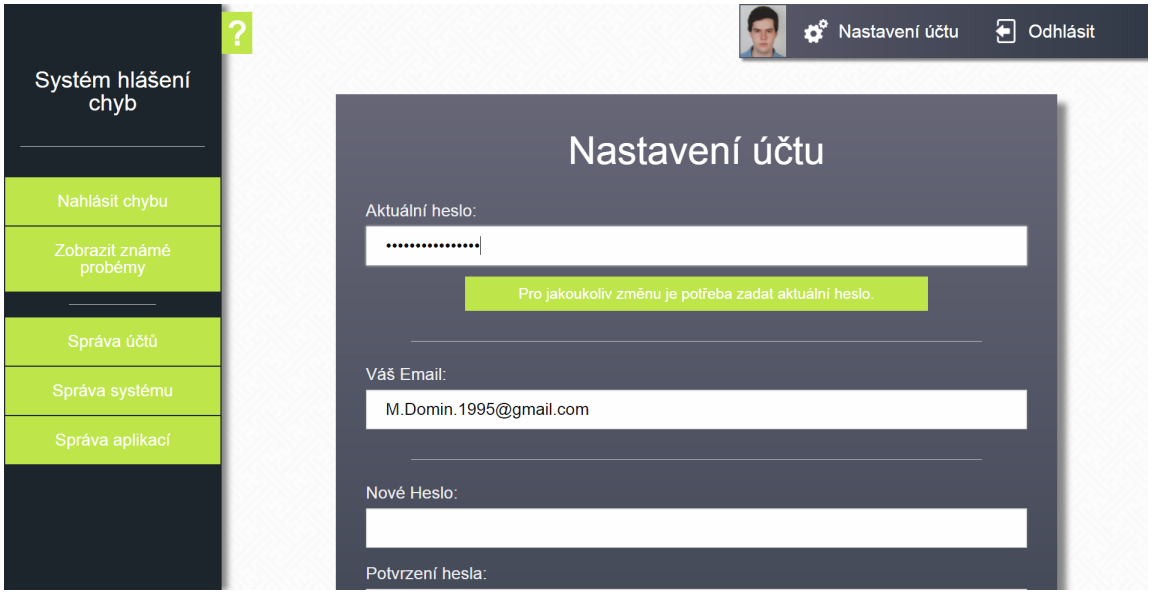


Obrázek B.5: Detail ticketu s režimem úprav část první

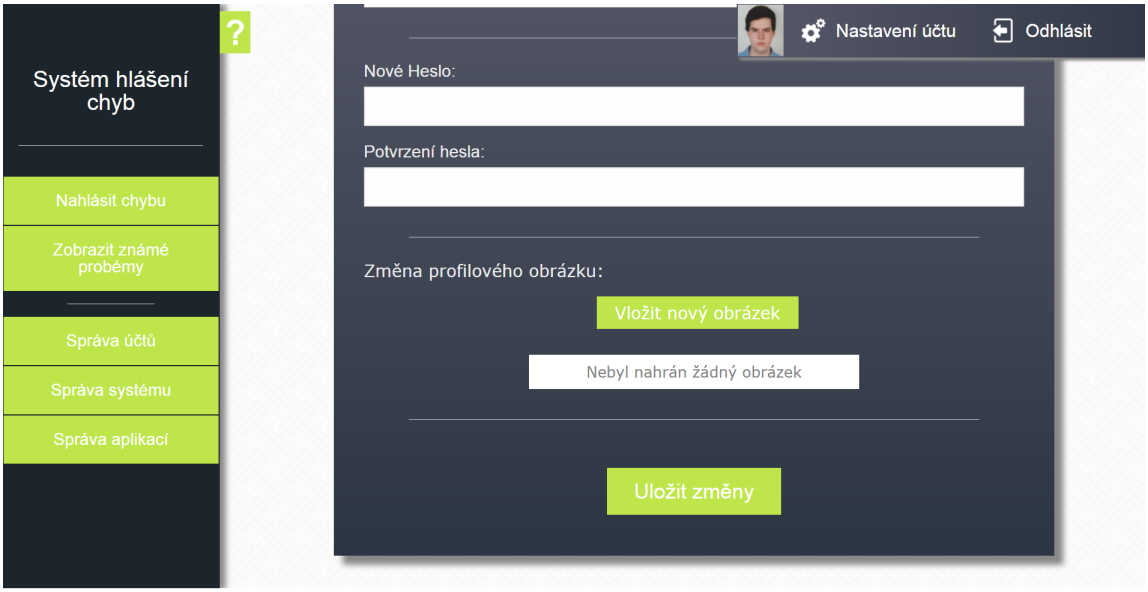
Obrázek B.6: Detail ticketu s režimem úprav část druhá

Obrázek B.7: Detail ticketu s režimem úprav část třetí

Obrázek B.8: Detail ticketu s režimem úprav část čtvrtá

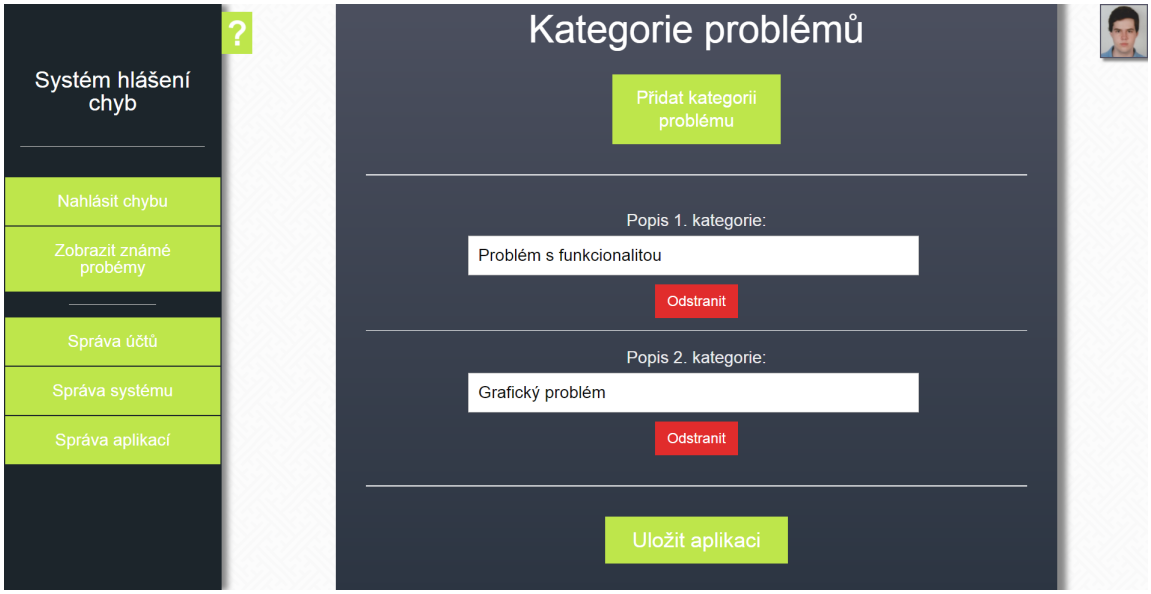


Obrázek B.9: Nastavení účtu část první



Obrázek B.10: Nastavení účtu část druhá

Obrázek B.11: Vytváření záznamu o nové aplikaci část první



Obrázek B.12: Vytváření záznamu o nové aplikaci část druhá

Obrázek B.13: Odlášení ze systému