Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Bakalářská práce

Informační systém pro správu chyb aplikací

Jakub Vévoda

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Jakub Vévoda

Informatika

Název práce

Informační systém pro správu chyb aplikací

Název anglicky

Information system for managing application bugs

Cíle práce

Cílem této práce je navržení informačního systému zajišťující celkovou a efektivní správu chyb v aplikacích prostřednictvím webových technologií.

Metodika

Na základě informací a potřeb získaných od zvoleného IT oddělení firmy, studiu odborných zdrojů a vyhodnocování získaných informací, v souladu se základními principy při tvorbě aplikací, vyhotovit vlastní informační systém za využití technologií jako je PHP 5, PHP 7, Linux Server, MySql 8, Apache, PostgreSQL, JavaScript, CSS3, HTML5. V závěru bude zhodnoceno naplnění stanovených cílů.

Doporučený rozsah práce

30-40 stran

Klíčová slova

systém, web, bug, informační, php, online

Doporučené zdroje informací

BRUCKNER, T. *Tvorba informačních systémů : principy, metodiky, architektury.* Praha: Grada, 2012. ISBN 978-80-247-4153-6.

HYSLOP, B. – CASTRO, E. *HTML5 a CSS3 : názorný průvodce tvorbou WWW stránek*. Brno: Computer Press, 2012. ISBN 978-80-251-3733-8.

KLČOVÁ, H. – SODOMKA, P. *Informační systémy v podnikové praxi*. Brno: Computer Press, 2010. ISBN 978-80-251-2878-7.

POKORNÝ, J. – GILMORE, W J. *Velká kniha PHP 5 a MySQL : kompendium znalostí pro začátečníky i profesionály.* Brno: Zoner Press, 2011. ISBN 978-80-7413-163-9.

SUEHRING, S. *JavaScript : krok za krokem.* Brno: Computer Press, 2008. ISBN 978-80-251-2241-9.

Předběžný termín obhajoby

2018/19 LS – PEF

Vedoucí práce

Ing. Marek Pícka, Ph.D.

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 24. 1. 2019

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 24. 1. 2019

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 12. 03. 2019

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci "Informační systém pro správu chyb aplikací" jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 14.03.2019 _____

Poděkování

Rád bych poděkoval vedoucímu mé bakalářské práce Ing. Marku Píckovi, Ph.D. za odborné vedení práce a za možnost konzultací, kde mi byly poskytnuté cenné rady, které jsem využil pro svou bakalářskou práci .

Dále bych chtěl poděkovat zadavatelům práce za možnost zpracovávat bakalářskou práci v jejich firmě a dalším lidem, kteří mi během vypracovávání práce poskytovali užitečné rady, které jsem využil pro svou praktickou část bakalářské práce.

Informační systém pro správu chyb aplikací

Abstrakt

Tato bakalářská práce se zabývá navržením informačního systému zajišťujícího celkovou a efektivní správu chyb v aplikacích prostřednictvím webových technologií. Systém firma využívá k zaznamenávání kompletní cesty chyby ve firmě, od jejího nalezení testerem, až po vyřešení chyby programátorem. Vysvětluje důležité pojmy přímo spojené s danou problematikou. Dále se bude zabývat vysvětlením informačního systému a jeho částí.

Systém se skládá z několika částí. První částí je samotná webová aplikace, ve které je znázorněn pohyb nalezené chyby. Druhou částí systému je databáze, do které se ukládají záznamy o chybách.

Klíčová slova: systém, web, chyba, informační, online, server, rozhraní, vývoj, PHP, framework.

Information system for managing application bugs

Abstract

This bachelor thesis deals with the design of an information system ensuring overall and effective application bug management through web technologies. The system used by the company to track path of bug from detection by the tester to repair by the programmer. It explains important concepts directly related to the issue. It will also deal with the explanation of the information system and its parts.

The system consists of serval parts. The first part is the web application itself, in which the motion of the detected errors is shown. The second part of the system is the database in which bug records are stored.

Keywords: system, web, bug, information, online, server, interface, development, PHP, framework.

Obsah

Seznam obrázků

Seznam použitých zkratek

HTML	Hypertext Markup Language.
CSS	Cascading Style Sheets.
PHP	Hypertext Preprocessor, původně Personal Home Page.
WEB	World Wide Web.
URL	Uniform Resource Locator.
HTTP	Hypertext Transfer Protocol.
MySQL	My Structured Query Language.
IS	Informační systém.
UI	User interface.
MMDIS	Multidimensional Management and Development of Information Systems.

1 Úvod

V dnešní době je vyvíjena spousta nových aplikací, kterým je při vývoji nutné poskytovat zpětná vazba jak od zákazníků, tak od testerů přímo ve firmě. Tato zpětná vazba, v podobě chyb nebo požadavků na další vývoj aplikace, by měla být někam ukládána. Proto jsme se ve firmě rozhodli pro vývoj vlastní aplikace pro uchovávání této zpětné vazby. Aplikace nabízené na trhu naší firmě nevyhovují cenou a nedostatkem speciálních funkcí, které naše firma požaduje. Firma se rozhodla, že pověří mě, abych se postaral o zpracování a vývoj této aplikace.

Práce je rozdělena na dvě hlavní části – teoretickou část a praktickou část práce.
V teoretické části se budu zabývat rozborem problematiky informačních systémů tvořené pomocí webových technologií, vysvětlením pojmů, metodiky a zásadami informačních systémů a způsobem vývoje. Dále se zaměřím na využití informačních systémů v podniku.

V praktické části se zaměřím na zjištění požadavků a priorit od zástupců firmy. Na základě zjištěných informací bude zpracován návrh funkcí aplikace a vytvořené návrhy budou okomentovány. Dále budou vyřešené speciální funkce, které firma požaduje. Následovat bude návrh databáze a samotný návrh a vývoj informačního systému.

Bakalářská práce je zpracována za použití odborné literatury, dokumentací k příslušným použitým technologiím, internetových zdrojů a osobních zkušeností autora v uvedené problematice.

2 Cíl práce a metodika

2.1 Cíl práce

Cílem práce je navržení informačního systému pro správu chyb v aplikacích pomocí webových technologií. Systém bude navržen zejména pro testery, zákazníky, manažery a programátory aplikací, kteří budou moci prostřednictvím systému sledovat proces řešení chyby nebo požadavku na vývoj. Informační systém bude zobrazovat stav řešení chyby nebo požadavku. Tester bude moci přidávat nové chyby do systému a přidělovat určenému programátorovi na opravu. Manažeři a zákazníci budou přidávat požadavky na další vývoj aplikace. Programátoři uvidí jim přidělené položky. Po dokončení opravy nebo vývoje, programátor přidělí záznam zpět testerovi pro konečné přetestování.

2.2 Metodika

V teoretické části se budu zabývat teoretickými východisky pro informační systém. Nejprve se zaměřím na odbornou literaturu vztahující se k problematice informačních systémů. Budou vysvětleny základní pojmy a principy informačních systémů. Dále bude vysvětlena metodika vývoje a nástroje k tomu vhodné. Studiem se seznámím s programovacími jazyky, které budu používat k řešení. A v práci samotné uvedu jejich základní charakteristiky a k čemu budou využívány.

Praktická část bude obsahovat analýzu a hlavně návrh systému samotného. Dále se bude věnovat vývoji požadovaných částí systému. V analýze se bude zabývat zjištěním požadavků a priorit, které má systém splňovat. V návrhu půjde především o navržení části, která bude realizována v informačním systému. Návrh bude přednesen zástupcům firmy na schválení. Po schválení návrhu se bude věnovat vývoji navržených částí. Informační systém bude vyvíjen pomocí HTML, CSS, PHP, Javascriptu a Frameworky pro PHP.

3 Teoretická východiska

V uvedené kapitole se práce bude zabývat vysvětlením všech základních a důležitých pojmů nutných pro navržení a vývoj informačního systému pro správu chyb. V první kapitole přiblíží co jsou chyby v aplikacích, jak vznikají a jak chyby dělíme z hlediska podniku a z pohledu programátora. V dalších kapitolách se bude věnovat webu, informačnímu systému a nutným technologiím k tvorbě tohoto projektu. Tyto poznatky teoretických východisek budou uplatněny k tvorbě tohoto projektu.

3.1 Chyby v aplikacích

Programátorská chyba se často označuje anglickým výrazem bug. Bug způsobuje nekorektní chybu, která negativně ovlivňuje chod aplikace či programu. Chyby také mohou způsobit úplný pád programu či představovat vážný bezpečnostní problém, mohou se projevovat neobvyklými grafickými prvky nebo neočekáváným chováním aplikace. [1]

Chyby odstraňujeme pomocí odlaďování neboli debugování, často v rámci testování kódu. Další chyby se hledají v testovací verzi programu či aplikace. Bugy v testovací verzi programu hledají testeři. [1]

3.1.1 Vznik chyby

Chyba vzniká velmi jednoduše, programátor chybu v kódu udělá. Někdy použije špatnou proměnnou, někdy okopíruje kus kódu, který nefunguje tak jak má, někdy špatně zapíše do řádku. Pokaždé, když programuje, přidá tyto chyby do kódu. [2]

3.1.2 Dělení chyb z hlediska podniku

Z hlediska podniku dělíme chyby podle priorit jejich řešení. [12]

- Minor
- Normal
- Critical
- Blocker

Chyba typu minor se vyznačuje velmi malou prioritou. Tuto chybu programátor může odložit na později a věnovat se chybám s prioritou větší. Občas se můžeme setkat s případem, kdy chyba typu minor nemusí být opravena.

Chyba typu normal má standardní prioritu řešení. Programátor by chybu měl opravit, ale nemusí ji opravovat ihned.

Chyba typu critical by měla být opravena v nejkratším čase. Často velmi ohrožuje běh programu, tudíž má prioritu nejvyšší.

Chyba typu blocker má specifické vlastnosti. Tento typ chyby většinou brání testerovi v dokončení nějakého důležitého testu. Priorita odladění této chyby závisí na domluvě testera s programátorem.

3.1.3 Dělení chyb z hlediska programátora

Chyby z hlediska programátorů můžeme rozdělit do několika kategorií. [3]

- **Sémantické chyby** odhalí překladač během překladu, tyto chyby se nejjednodušeji hledají i opravují. Překladač programátorovi nabídne dostatek informací na to, aby chybu našel a opravil.

- **Běhové chyby** tyto chyby se hledají hůř. Překladač je neodhalí a za běhu programu se mohou, ale nemusí objevit. Běhové chyby způsobí, že se program nechová tak, jak programátor chtěl. Program se dostane do nestandardní situace, kterou program ohlásí vygenerováním výjimky nebo systémové chyby. Takováto chyba může být zachycena a zpracována programátorem, ale často není zpracování chyby možné.

- **Logické chyby** jsou nejzávažnější chyby, které programátor může udělat. Tyto chyby jsou závažné tím, že většinou nemají žádné symptomy nebo se projevují nepřímo. Z pohledu překladače nedojde k žádné zvláštní situaci, tudíž je programu povoleno pokračovat. Chyba se vyznačuje rozporem mezi tím, co má program dělat a tím, jak byl naprogramován. [4]

3.2 Webová stránka

Protože informační systém bude vyvíjen prostřednictvím webových technologií, nejprve musím vysvětlit, co je to vlastně web. Web a internet jsou často zaměňující se pojmy. Proto nejprve definuji internet a poté web.

3.2.1 Internet

Internet je síť vzájemně propojených počítačů. Je navržen k dopravě dat z jednoho počítače na jiný. Pro dopravu dat existují vždy alespoň dvě komunikační cesty. [5]

3.2.2 Web

Web je cenným článkem současnosti. Znamená především velký informační zdroj. Web je počítačový software, který pracuje na internetu. Jsou to dva počítačové programy. [5]

- **Webový prohlížeč**, který zpracovává a zobrazuje webové stránky, které obdrží od web serveru.

- **Webový server**, který uchovává a spravuje webové stránky v jejich zdrojovém tvaru. Pracuje v online režimu a díky tomu jsou webové stránky ihned k dispozici. Server na žádost posílá stránky prohlížeči.

Web je postaven z několika částí.

- **HTML** je značkovací jazyk pro tvorbu webových stránek.

- **URL** jako efektní způsob adresování webových stránek.

- **HTTP** je protokol, který umožňuje prohlížečům získávat webové stránky.

Dále musím definovat, co jsou to webové stránky. Webové stránky jsou hypertextové dokumenty. Lze je prohlížet a číst v případě zájmu ihned a bez ohledu na to, ke kterému dokumentu patří a na kterém počítači v internetu jsou právě uloženy. [5]

3.3 Informační systém

Pro pojem informační systém existuje již mnoho definic, a proto vyberu tu nejvíce vhodnou. Informační systém je soubor lidí, technických prostředků a metod (programů), zabezpečujících sběr, přenos, zpracování, uchování dat, za účelem prezentace informací pro potřeby uživatelů činných v systémech řízení. Jinak řečeno, systém je množina vzájemně propojených komponent, které musí pracovat dohromady pro celý systém tak, aby tento systém naplnil daný účel. [6]

Informační systém je také definován zákonem o informačních systémech. [Zákon č. 365/2000 Sb.]: „Informačním systémem je funkční celek nebo jeho část zabezpečující cílevědomou a systematickou informační činnost. Každý informační systém zahrnuje data, která jsou uspořádána tak, aby bylo možné jejich zpracování a zpřístupnění, provozní údaje a dále nástroje umožňující výkon informačních činností."

Ještě než přejdu k architektuře informačního systému, musím definovat, co je to informace a data. Data obsahují nějaká sdělení, a jestliže budou tato data někým využita a tím pro něj vznikne nějaký užitek, stanou se z dat informace. Takže teprve v okamžiku smysluplného užití dat vzniká informace. Informací tedy rozumíme data, kterými jejich uživatel přisuzuje nějaký význam a uspokojí konkrétní objektivní informační potřebu svého příjemce. [6]

3.3.1 Architektura informačního systému

Architektura informačního systému je pro vývoj velmi významná. Význam by se dal přirovnat například k plánům pro stavbu domu. Při vývoji informačního systému je často viděno, že je komplikovaný systém vyvíjen bez jakékoliv architektonické představy. V dnešní době patří kvalitní informační systém k hlavním faktorům, které významně ovlivňují úspěšnost, kvalitu a výkonnost firmy. Architektura systému je prostředkem, který umožňuje navrhnout a jasně popsat vztahy všech zainteresovaných pracovníků. [11]

3.3.1.1 Přístupy k tvorbě architektur

V následujících kapitolách stručně popíši, z mého pohledu, nejdůležitější přístupy k tvorbě architektury informačních systémů.

3.3.1.1.1 Podniková architektura

Podniková architektura je přístup, který vyjadřuje fundamentální uspořádání vztahu mezi informačním systémem a byznysem. Toto uspořádání vede k naplnění poslání organizace a zároveň respektuje okolní prostředí a dodržuje principy návrhu a rozvoje systému. Hlavním cílem této architektury je zvládnout zvětšující se komplexitu distribuovaných systémů a lepší sjednocení projektů s byznys potřebami. [11]

3.3.1.1.2 Modelem řízená architektura

Modelem řízená architektura je založena na myšlenkách, které nejsou nijak nové. Tato architektura prosazuje možnost oprostit se od technologických problémů a zaměřit se na věcné problémy. Nástroje podporující modelem řízenou architekturu umožňují zpětné inženýrství. Díky tomu je možné vytvořit modely již existujících systémů, například pro účely integrace aplikací. [11]

3.3.1.1.3 Architektura MMDIS

Poslední typ architektury, který uvedu je MMDIS. MMDIS určuje, z jakých konkrétních komponent se bude informační systém skládat a jaké budou jejich vzájemné vazby. Také určuje principy vývoje a provozu informačního systému. Cílem těchto principů je dosáhnout požadovaných vlastností systému. Těmito vlastnostmi je myšleno dostupnost, včasnost, správnost a důvěryhodnost funkcí a informací, uživatelská přívětivost, bezpečnost a pokrytí požadovaných funkcionalit. [11]

3.3.2 Vývoj informačního systému

V současné době existuje mnoho variant vývoje systému. Pro tvůrce i uživatele informačních systémů je velice důležité tyto varianty znát. Dále je potřeba umět posoudit jejich výhody a nevýhody a na základě těchto znalostí poté rozhodnout jak postupovat.

Vývoj informačního systému je proces, který má dosáhnout plánované změny informačního systému. Změna se může týkat všech komponent obsažených v systému.

Podstatné změny se realizují jako projekt. Ukončením tohoto projektu vzniká nová verze informačního systému. Z hlediska byznysu je nejpodstatnější změna taková, při které dochází ke změně softwarové aplikace ovlivňující průběh byznys procesu nebo dat uvnitř tohoto procesu. Existují 2 základní metody pro vývoj aplikace a to IASW, což je aplikace vytvořená na míru podle potřeb podniku. Druhým tipem je TASW. TASW je typový aplikační software. Specifickým případem TASW je open-source software. Pro naše účely budeme potřebovat metodu ISAW.

ISAW je individuální aplikační software. Jak jsem již zmiňoval, aplikace tvořené touto metodou jsou tvořené na míru podniku. Je přesně dáno, na jakých komponentách informační systém poběží. Funkcionalita aplikace je navržena tak, aby optimálně podporovala činnosti podniku, pro který je určena. Podnik díky aplikaci vyvinuté pomocí této metody podpoří svoje specifické procesy a dosáhne tak specifických cílů na trhu. [11]

3.4 Technologie tvorby informačních systémů na webu

Jak jsem již psal, web je složen z několika částí. Nyní představím všechny technologie, které jsou potřeba k tvorbě informačního systému a rozdělím je na technologie na straně klienta, serveru a na technologie databázové. Dále představím editor, ve kterém budu pracovat.

3.4.1 Technologie na straně klienta

Základní technologie na straně klienta jsou HTML, CSS, JavaScript. Tyto technologie v následujících kapitolách stručně představím a uvedu příklad kódu.

3.4.1.1 HTML

HTML je standardní značkovací jazyk využíván k tvorbě webových aplikací a webových stránek. Tento jazyk má svou přesnou syntaxi. Syntaxe je od vzniku HTML neustále upravována a vylepšována. Nyní máme verzi HTML 5. Pozoruhodnost a unikátnost syntaxe spočívá v tom, že je sice exaktně definována, ale současně je velice flexibilní.

Jazyk HTML je pouze a vždy textového formátu. Pokud se na stránce vyskytují binární data v podobě obrázků nebo animací, je na ně odkázáno, tudíž binární data nejsou uložena do

souboru se základním popisem stránky. Jakmile je vytvořena html stránka v textovém editoru, je vytvořen její finální tvar, který přímo čte a interpretuje prohlížeč.

HTML příkazy jsou spolu s parametry uzavírány do špičatých závorek. Jakmile se v HTML souboru objeví výrazy uzavřené do špičatých závorek, jedná se o příkaz, který nějakým způsobem definuje formátování elementů na stránce. Vždy hned za otevírací závorkou je jméno příkazu, dále jsou pak jeho parametry. [7]

Příkazy neboli tagy se dělí na dva druhy. [7]

- **Párové příkazy** slouží k formátování elementu. První díl párového příkazu je před formátovým elementem a druhý díl se nachází těsně za formátovaným elementem. Tím je vymezena oblast, na kterou se formátování aplikuje. Základní párové tagy jsou <DOCTYPE>, <html>, <title>, <head> a <body>.

- **Nepárové příkazy** se vztahují na celý dokument. Vztahují se například na element , který už je sám o sobě přesně vymezen. Tím je třeba obrázek <img>.

Nejnovější verze HTML je HTML 5.3. HTML 5 přineslo spoustu novinek. Například nová sémantika včetně příkazů jako jsou <header>, <nav>, <article>, <section>, <footer> a další. Zápis HTML kódu je na obrázku č.1. [7]

Příklad kódu HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Hello world</title>
</head>
<body>
<p>
    My first web page.
</p>

</body>
</html>
```

Obrázek 1 - Příklad kódu HTML

3.4.1.2 CSS

Další technologií na straně klienta jsou CSS. Kaskádové styly změnily stylování HTML stránky. HTML počítalo s tím, že text bude formátován přímo na místě pomocí formátovacích tagů bez jakéhokoliv předdefinování. Nyní můžeme definovat styl přímo v hlavičce dokumentu nebo jej lze importovat z externího dokumentu. Další možností je zápis přímo u prvku, který chceme stylovat. Při importování CSS souboru je potřeba svázat soubor CSS se souborem HTML pomocí odkazu na CSS soubor v hlavičce HTML. Vzor příkazu pro svázání souborů <link href="style.css" type="text/css" rel="stylesheet">.

Kaskádové styly s sebou nesou mnoho výhod. Zmenšují objem HTML souboru, což je pozitivně vnímáno uživateli. Další výhodou je, že jediný text může ovlivňovat několik stylů, které jej formátují. Nenastává chyba konfliktu, protože přednost má ta definice, která přichází jako poslední.

Je zde ale i nevýhoda, a to komplikace v případě importování stylů z externího souboru. Jelikož externích souborů můžeme importovat více. Zde opět platí pravidlo, že později importovaný styl má přednost. Ovšem v celkovém seznamu priorit mají přednost lokální definice. Importy mají nejnižší důležitost. Další nevýhodou je, že každý prohlížeč na CSS může reagovat jinak. Vzor zápisu CSS přímo k prvku a do hlavičky souboru je zobrazen na obrázku č.2. Na obrázku č.3 je ukázán příklad importu CSS. [7]

Příklad kódu CSS přímo v HTML souboru

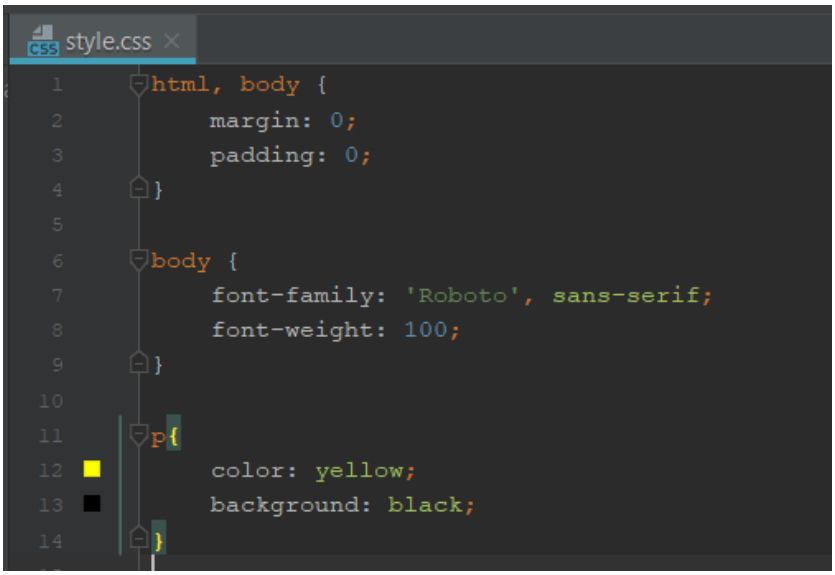
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Hello world</title>
    <style>
        p{color: yellow}
    </style>
</head>
<body>
<p style="background: black" >
    My first web page.
</p>

</body>
</html>
```

Zápis přímo do hlavičky souboru HTML

Zápis přímo u prvku

Obrázek 2 - Příklad kódu CSS přímo v HTML souboru

Příklad CSS kódu importem.



Obrázek 3 - Příklad kódu CSS importem

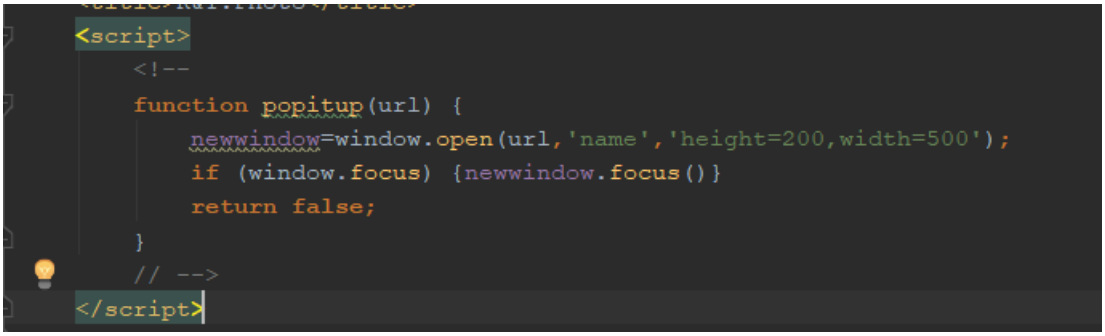
3.4.1.3 JavaScript

JavaScript je další technologií na straně klienta. Pomocí JavaScriptu se zabezpečuje, aby stránka byla dynamická. Dynamickou stránku lze definovat tak, že podle reakce uživatele a na základě jím zadaných údajů server dynamicky odpoví. Server vygeneruje obsah stránky a pošle ho přes síť na počítač uživatele. Počítač uživatele je vhodné používat nejen k zobrazení stránek dokumentů, ale i ke změně obsahu dokumentu, formy jeho zobrazení. Dále je vhodné užívat počítač uživatele i ke kontrole zadaných údajů. Díky těmto využití uživatelského počítače dojde k redukci množství přenášených dat a výrazně se zrychlí odezva systému na podněty uživatele.

JavaScript je závislý na prohlížeči. Každý uživatel má možnost ho vypnout. Díky tomu vzniká problém při vytváření stránky, jelikož programátor musí ověřit funkčnost stránky s vypnutým a zapnutým JavaScriptem.

Zápis probíhá stejně jako u kaskádových stylů. Můžou se zapisovat přímo do HTML pomocí párového tagu <script>. Dále mohou mít zápis v externím souboru, na který odkážeme. Nebo zápis provádíme jako atribut jiného tagu. Použití JavaScriptu v HTMl je dost podobné jako CSS, a proto uvedu pouze jeden příklad syntaxe JavaScriptu. Příklad je uveden na obrázku č.4. [8]

Vzor kódu JavaScriptu v hlavičce HTML souboru



Obrázek 4 - Vzor kódu JavaScriptu v hlavičce HMT souboru

3.4.2 Technologie na straně serveru

Jedinou technologií na straně serveru, kterou využívám je PHP a dále Frameworky vycházející s PHP. Tyto technologie představím v následující kapitole.

3.4.2.1 PHP

PHP je nejrozšířenější skriptovací jazyk pro web. Tento program typicky běží na nějakém webovém serveru a prostřednictvím webového prohlížeče k němu přistupuje spousta lidí. PHP engine se stará o zpracovávání programů napsané právě v PHP a generuje webové stránky. Díky tomuto programovacímu jazyku je možno tvořit dynamické webové stránky. Obsah stránky je přizpůsoben konkrétní osobě. Nevýhoda dynamičnosti stránky je, že nejde upravit v momentě, kdy si ji příjemce čte. Pokud ale příjemce postupuje podle pokynů, které obdržel, lze upravit i celou stránku. Pomocí různých přesměrování, vyskakovacích oken atd. Vzor kódu PHP je zobrazen na obrázku č.5. [8]

Vzor kódu PHP

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>PHP says hello</title>
</head>
<body>
<?php
print "Hello, World!";
?>
</body>
</html>
```

Obrázek 5 - Vzor kódu PHP

3.4.2.2 Framework

V této části práce uvedu, které frameworky jsem použil při vývoji informačního systému, jakou přinášejí pro vývoj výhodu a jejich hlavní využití. Vybral jsem si framework Bootstrap s kombinací frameworku Nette a s editorem PhpSrom kvůli snadnému použití a hlavně podpoře těchto frameworků PhpStormem.

3.4.2.2.1 Bootsrap

Bootsrap je jedním z nejpoužívanějších frameworků pro práci s HTML, CSS a JavaScriptem. Tento framework se stará hlavně o vizuální stránku systému a responzivitu stránky. Pro to, aby Bootstrap fungoval, je potřeba nejnovější HTML 5. Bootstrap přináší velkou úlevu pro programování v HTML a CSS. Na oficiálních stránkách je k dispozici spoustu vizuálních podob webu. Programátor si vybere podobu odpovídající jeho požadavkům a nakopíruje kód do svého kódu. Vizuální vzhled se dá samozřejmě upravit pomocí vlastních css stylů, které umístí tak, aby byly nadřazené stylům od bootstrapu.[13]

3.4.2.2.2 Nette

Framework Nette je několik vyspělých a samostatně použitelných komponent pro Php. Nette zajišťuje bezpečnost programu, na kterou se v poslední době klade velký důraz. Obsahuje ladící nástroj nazvaný Tracy, díky kterému lze pohodlně odhalit většinu chyb. Další skvělou výhodou je šablonovací systém zvaný latte. Tento šablonovací systém má stejný

zápis jako Php a díky tomu se programátor nemusí bát, že by neznal syntax. Výhodou je jeho rychlost. Latte kompiluje šablony za běhu, takže se v rychlosti oproti php neliší. [14]

3.4.3 Technologie databázové

PHP se může připojovat až k několika databázím najednou. MySQL ve spojení s PHP je nejpopulárnější kombinací k tvorbě webu. V následující kapitole MySQL stručně představím.

3.4.3.1 MySQL

MySQL je relačně databázový systém, který běží na serveru. Je také velmi rychlý a lehce použitelný. Spolu s PHP funguje na více platformách. MySQL je ideální pro malé i velké aplikace a tudíž jsem se ho rozhodl pro svou práci využít. [9]

3.4.4 Editor

Jako editor pro svůj projekt jsem si vybral PhpStorm. Právě toto vývojové prostředí umožňuje pracovat s většinou frameworků. Dále umožňuje lehce odlaďovat a testovat. V případě potřeby refactoringu lze snadno a bezpečně nahrazovat a přejmenovávat.[10]

4 Vlastní práce

V následujících kapitolách se práce bude věnovat popisu průběhu vývoje informačního systému počínaje konzultací se zadavatelem projektu, kde získám požadavky, které IS musí splňovat. Následuje vypracování návrhu. Po schválení návrhu informačního systému zadavatelem začne vývoj. Vývoj informačního systému se rozdělí do několika vývojových částí.

4.1 Konzultace požadavků

Moje práce začíná konzultací s firmou. Konzultaci jsem provedl se všemi, kteří informační systém budou využívat. Od uživatelů jsem získal požadavky na informační systém, které jsem prokonzultoval s vedením. Vedení chtělo vyhovět zaměstnancům a umožnit jim, aby systém co nejvíce odpovídal požadavkům uživatelů. Dále jsem dostal na výběr, jestli se pokusím refaktorovat jejich starý systém nebo navrhnu úplně nový. Všechny požadavky uživatelů byly zadavateli úspěšně schváleny. Nyní je očekáváno předložení návrhu řešení získaných požadavků. Až bude návrh získaných požadavků schválen, očekává se návrh architektury a UI. Požadavky na informační systém uvedu v další kapitole.

4.1.1 Požadavky na informační systém

Všechny požadavky získané od zadavatelů by měly být splněny. Byly zde ale i požadavky s menší prioritou, které není nutné splňovat ihned v první verzi systému. Je možné je dodělat v dalších verzích.

Získané požadavky:

- Zjednodušit IS.
- Zvýšit přehlednost systému.
- Rozlišovat prioritu opravy pro chyby a požadavky.
- Možnost přidat nový požadavek na systém zákazníkem.
- Přidat testerovi možnost znovu otevřít chybu.

Nyní získané požadavky upřesním. Jedním z nejdůležitějších požadavků je zjednodušit systém. Uživatelé si stěžují na příliš mnoho zbytečných akcí i při triviální práci se systémem.

Další požadavek by měl ulehčit a zkrátit komunikaci a práci se zadáváním nalezené chyby. Jedná se o přidání možnosti rozlišit prioritu její opravy.

Firma chce, aby zákazník mohl přidávat požadavky na zlepšení systému, popřípadě doplnění nějakých vhodných funkcí.

Další požadavek se týká změn v systému. Jestliže tester nalezne chybu nebo programátor něco opraví, nikdo se to nedozví, dokud neaktualizuje stránku se záznamy.

Pokud programátor opraví chybu a tester po přetestování opravené chyby zjistí, že je opravená chybně, musí tester založit nový záznam o chybě. Toto chování systému je zbytečně složité, a proto ho firma chce odstranit.

4.1.2 Řešení požadavků

V této části uvedu možná řešení požadavků, které jsem získal od zadavatelů a uživatelů systému.

Pro řešení prvního požadavku jsem firmě nabídl vytvoření akčního menu s nejčastějšími úlohami, které uživatelé vykonávají.

Návrh na řešení problému s přehledností je uživatele omezit právy na projekt, kde by se ke každé chybě při vyplňování záznamu umožnilo vložit specifický projekt. Poté by tuto chybu viděl pouze uživatel přiřazený k projektu. Další možné řešení je vytvořit filtr těchto záznamů, kde by si každý uživatel mohl odfiltrovat chyby a požadavky zvlášť.

V systému, který teď firma používá, není rozlišeno, jakou má chyba či požadavek prioritu řešení. Řešením by byla jednoduchá značka, která by záznam označila prioritou.

Zákazníkovi chce firma umožnit přidávat požadavky do systému. Řešením tohoto problému bude tlačítko v navigačním menu, které umožní zákazníkovi přidat nový požadavek.

Požadavek na změny v systému bych řešil automaticky generujícím emailem. V případě, že tester vytvoří nový záznam o nalezené chybě, systém automaticky odešle email osobě, které je záznam přidělen.

Řešení posledního požadavku je velmi triviální a souvisí s řešením předchozím. Do záznamu o stavu chyby přidám položku reopened, což znamená, že tester záznam neschválil jako opravený a znovu jej otevřel do řešení.

Dále jsem zadavatelům navrhnul jaké stavy a priority, chtějí rozlišovat. Priority řešení záznamu budou stejné, jako jsem uvedl v teoretické části práce. Stavy pro chybu a požadavek budou následující:

- **New –** nově vytvořená chyba nebo požadavek.
- **Accepted –** požadavek nebo chyba přijatá programátorem k opravě.
- **Solved –** po dokončení opravy programátor změní značku stavu na solved a tím dává najevo testerovi, že je možné provést přetestování.
- **Approved –** po úspěšném přetestování tester změní značku na approved a tím dává signál manažerovi, že je chyba úspěšně opravena.
- **Reopened –** v případě, že přetestování opravené chyby nedopadne úspěšně, tester změní značku na reopened a dává chybu nebo požadavek zpět programátorovi k další opravě.
- **Done –** done je značka, kterou záznamu přiděluje manažer projektu. Tuto značku manažer dává záznamu v případě, že oprava i přetestování dopadlo úspěšně.

Dále jsem uvedl informaci o uchovávání opravené chyby v systému. Navrhl jsem mazat opravené chyby ručně. Jelikož se build testovaných aplikací nevydává po každé opravě chyby, ale až po několika různých úpravách, bude lepší záznamy uchovávat do té doby, než se vydá changelog k nově vydané verzi. Po vydání changelogu manažer projektu opravené chyby ručně vymaže.

Zadavatelům se řešení těchto požadavků líbilo, a proto mi řešení bylo schváleno. Nyní se přesouvám na další část a to je navržení architektury informačního systému. Dále budu pokračovat návrhem UI pro nový systém.

4.2 Návrh informačního systému

Návrh informačního systému rozdělím do několika částí. Jednotlivé části budou uvedeny v dalších kapitolách.

4.2.1 Návrh architektury

Při návrhu architektury jsem vycházel z požadavků firmy. Jeden z hlavních požadavků firmy byl, aby informační systém běžel na vnitřní síti firmy z hlediska bezpečnosti a citlivosti informacích uváděných v systému. Na web se přistupuje z počítače uvnitř firmy nebo přes VPN. Připojení přes VPN není z pohledu firmy tak bezpečné, a proto chce firma využívat přístup hlavně z vnitřní sítě. Přes počítač a jeho webový prohlížeč se připojíme na webový server, na kterém běží informační systém. Webový server komunikuje s mysql databázovým serverem, ze kterého získává informace o uživatelích a záznamech.

4.2.2 Návrh databáze

Pro zadaný informační systém budeme potřebovat několik tabulek. První tabulka by měla obsahovat uživatele. Pro náš účel bude stačit, když v této tabulce bude informace o loginu a hesle, které se z důvodu bezpečnosti bude hashovat. Tabulka bude dále obsahovat práva uživatele. Práva uživatele budou uložena ve sloupečku role. Dále tabulka obsahuje jméno, příjmení a email uživatele.

V další tabulce budou uloženy informace o záznamu chyby. Tabulka bude obsahovat datum, kdy záznam vznikl, přiděleného uživatele, status a prioritu.

Dále budu potřebovat tabulku s informacemi o požadavku. Tabulka bude vypadat stejně jako tabulka s informacemi o chybě, jen se bude týkat požadavků. Požadavky budou mít také přiděleného svého uživatele, datum a čas vzniku, projekt, status a prioritu. Tyto dvě tabulky se mohou spojit do jedné tabulky s názvem report. V této tabulce se bude rozlišovat typ záznamu. Typem záznamu je chyba a požadavek.

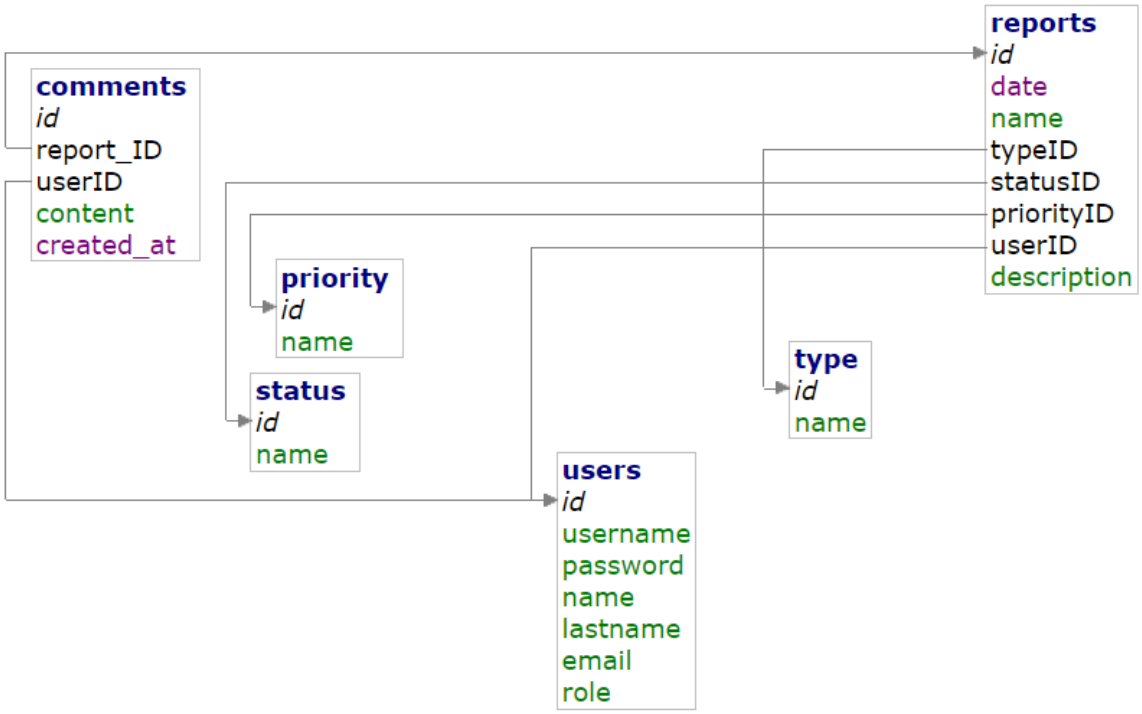
Tabulky o chybě a záznamu budou také obsahovat datum a čas poslední změny, v případě, že některý uživatel na záznamu udělá nějakou změnu. Bylo by také dobré vypisovat uživatele, který změnu provedl. Tento záznam ale z pohledu firmy není nutný.

Další tabulka bude zobrazovat komentáře k daným záznamům, takže bude svázána s tabulkou záznamů a s tabulkou uživatelů. Další položka této tabulky bude časové razítko.

Další databázové tabulky budou spojené s tabulkou o záznamu, ve kterých bude priorita, stav a typ. Tyto 3 tabulky budou již naplněny hodnotami. Tyto hodnoty pak uživatel bude vybírat při tvoření nového záznamu v systému.

Jelikož systém nechci dělat zbytečně komplexní a složitý, tak si myslím, že pro účely systému tyto tabulky stačí. Na uvedeném obrázku je znázorněno spojení všech tabulek, které jsem pro tvorbu systému využil. Schéma databázových tabulek a jejich propojení je vyobrazeno na obrázku č.6.

Schéma databáze



Obrázek 6 - Schéma databáze

4.2.3 Návrh UI

Uživatelské rozhraní webové aplikace bude tvořeno pomocí HTML a CSS. Nejprve UI popíšu a poté přidám nějaké obrázky z návrhů. Návrh by se od konečné aplikace neměl funkčně lišit. Změna bude pouze ve vizuální stránce.

Aplikace nebude graficky nijak složitá, jelikož ji budou používat pouze osoby seznámené s danou problematikou. Složitý grafický vzhled by pro vykonávání práce v aplikaci mohl být na obtíž. Proto volím jednoduchý a přehledný styl.

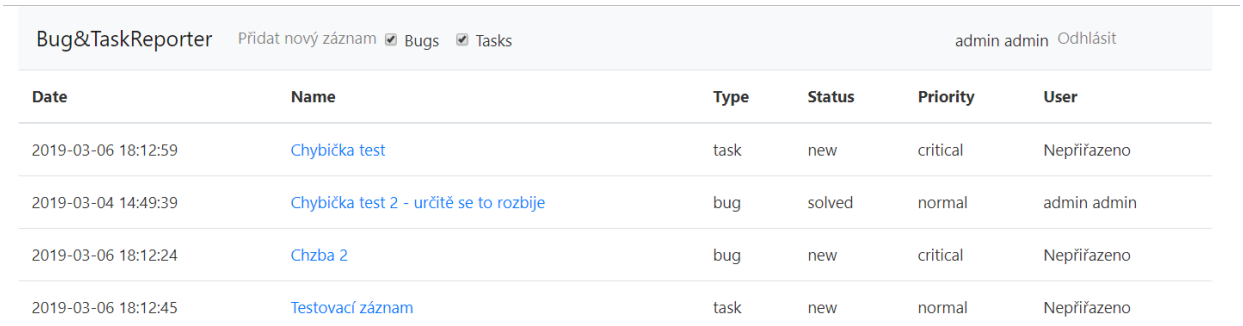
Každý uživatel se do aplikace musí přihlásit, aby ji mohl používat, tudíž první obrazovka bude přihlašovací okno. Do aplikace se bude logovat pomocí uživatelského jména a hesla. Ukázka přihlašovací obrazovky je na obrázku č.7.

Přihlašovací obrazovka

Obrázek 7 - ukázka přihlašovací obrazovky

Po přihlášení do aplikace se zobrazí hlavní obrazovka viz. obrázek č.8. V pravém horním rohu aplikace bude vidět přihlášený uživatel a možnost odhlášení ze systému. Na horním panelu bude možnost přepínat mezi chybami a požadavky od zákazníků nebo manažerů. V základním zobrazení budou na hlavní obrazovce vypsané chyby. Záznamy na hlavní stránce budou zobrazovat pouze nejdůležitější informace, a to stav, prioritu řešení a přiděleného uživatele. Vedle možnosti přidání nového záznamu bude umístěno filtrování chyb a požadavků, aby si uživatel mohl zobrazit pouze chyby nebo požadavky.

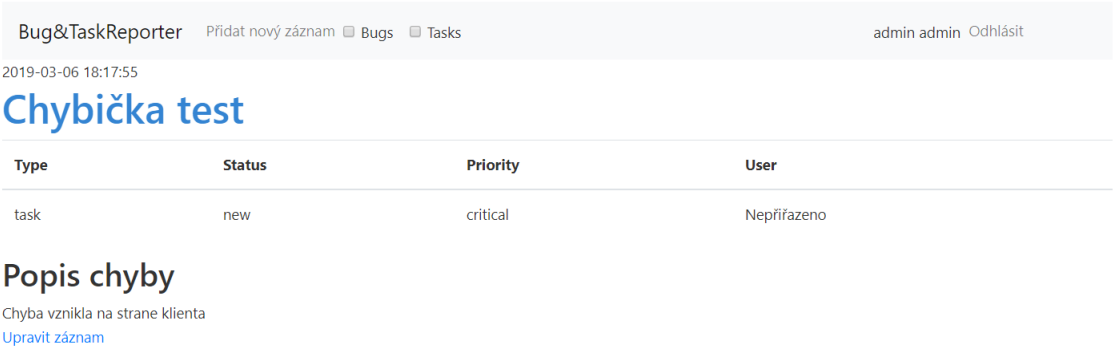
Domovská obrazovka s přehledem chyb a požadavků



Obrázek 8 - Ukázka domovské obrazovky

V seznamu na hlavní stránce vybereme záznam, který chceme otevřít. To nás přesměruje na stránku s vybraným záznamem, kde budou zobrazeny podrobnosti o záznamu. Dále zde bude možnost editace záznamu. Stránka z vybraným záznamem je na obrázku č.9.

Vzor stránky s chybou



Obrázek 9 - stránka s chybou

Nabídka menu s nejčastějšími akcemi bude umožňovat přidání nového záznamu. Tuto funkci chci nechat oddělenou, jelikož si uživatelé systému, který teď firma používá, stěžovali právě na složitost vytvoření nového záznamu. V případě, že by firma chtěla nový systém dále rozšiřovat, bude lepší nechat tyto akce oddělené. Po vybrání možnosti vytvořit nový záznam, budeme přesměrováni na novou stránku, kde vytvoření nového záznamu dokončíme. Zde bude možnost přidat název záznamu, možnost určit, zda se jedná o chybu nebo o požadavek, prioritu řešení záznamu a stav, v jakém se záznam zrovna nachází. Další možností bude přidělení řešitele dané chyby nebo požadavku. Vzor stránky pro vytvoření nového záznamu je na obrázku č.10.

Ukázka stránky nového záznamu

Obrázek 10 - Stránka s novým záznamem

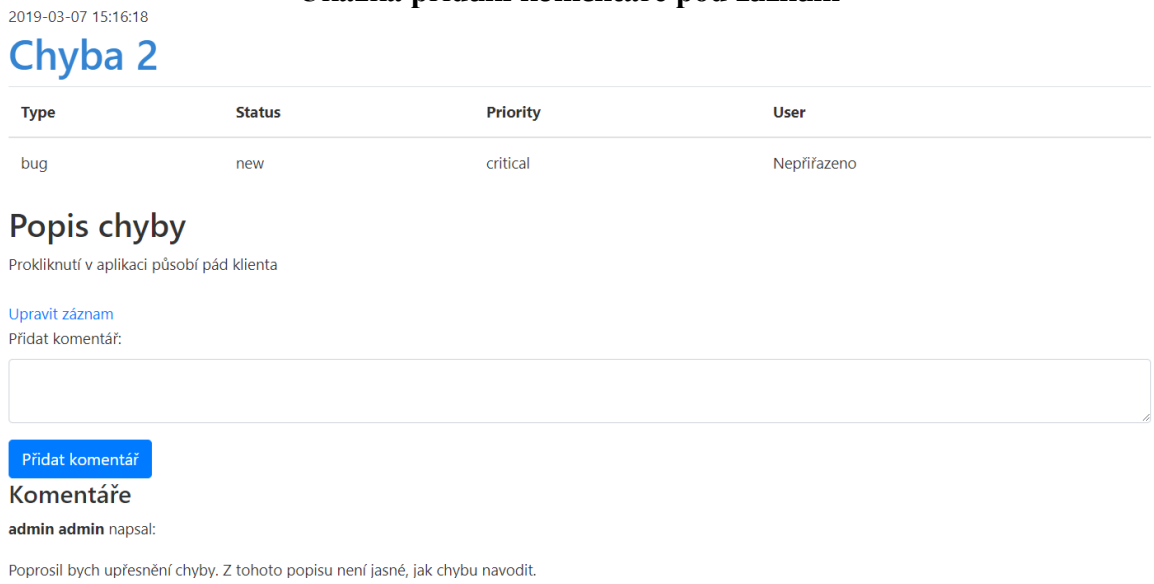
Každý záznam, jak již bylo zmíněno, jde editovat. Editace probíhá následujícím způsobem. Nejprve si v seznamu vybereme záznam, který chceme editovat. Po kliknutí na záznam se zobrazí stránka s vybraným záznamem. Po kliknutí na tlačítko upravit záznam se zobrazí stránka určená pro editaci záznamu viz. obrázek č.11.

Stránka pro editaci příspěvku

Obrázek 11 - Editace příspěvku

Další možnost, kterou systém umožňuje, je přidávání komentářů pod jednotlivé záznamy. Komentář se zobrazí se jménem a příjmením uživatele, aby bylo jasné, kdo záznam komentoval. Vzor stránky s komentáři je znázorněna na obrázku č.12.

Ukázka přidání komentáře pod záznam



Obrázek 12 - Vzor komentáře

4.3 Vývoj informačního systému

V této části práce popíši, jak probíhal vývoj aplikace. Ukáži nejdůležitější části kódu a popíši, co dělají. Budu se věnovat hlavně vývoji webové aplikace a komunikaci s databází.

4.3.1 Vytvoření databáze a serveru

Vývoj informačního systému jsem započal zprovozněním webového serveru a databázového serveru. Dále jsem pokračoval vytvořením databázových tabulek. K databázi jsem přistupoval přes adminer, který mi umožnil lehký návrh databázových tabulek a vytvoření vazeb mezi nimi. Dále jsem tabulky type, status, priority a users naplnil položkami. Do tabulky type jsem přidal položku bug a task. Tyto položky definují, zda se jedná o chybu nebo o požadavek. Do tabulky status jsem přidal 5 položek. Tyto položky znázorňují aktuální stav řešení záznamu. Do tabulky priority jsem vložil položky, které označují závažnost záznamu. Do tabulky users jsem ručně přidal prvního uživatele s názvem admin. Přes tohoto uživatele probíhá veškerá správa informačního systému. Heslo je samozřejmě hashováno. Z hlediska bezpečnosti ukáži pouze tabulky priority, status a type. Naplnění tabulky users vynechám. Tabulky jsou na obrázcích č. 13,14,15.

Tabulka priority

```
SELECT * FROM `priority` LIMIT 50
```

Změnit	id	name
upravit	1	critical
upravit	2	blocker
upravit	3	normal
upravit	4	minor

Obrázek 13 - Naplnění tabulky priority

Tabulka status

```
SELECT * FROM `status` LIMIT 50
```

Změnit	id	name
upravit	1	new
upravit	2	in-progres
upravit	3	solved
upravit	4	approved
upravit	5	done

Obrázek 14 - naplnění tabulky status

Tabulka type

```
SELECT * FROM `type` LIMIT 50
```

Změnit	id	name
upravit	1	bug
upravit	2	task

Obrázek 15 - Naplnění tabulky type

4.3.2 Tvorba webové aplikace

Pro tvorbu webové aplikace jsem použil vývojové prostředí PhpStorm. Pracoval jsem s HTML, CSS a PHP. Dále jsem využil sadu nástrojů Bootstrap. Pomocí Bootstrapu jsem pracoval hlavně s designem systému. Požadavek byl jednoduchost, a proto jsem nedělal design systému složitý a držel jsem se základních barev. Dále jsem využil framework Nette, který mi výrazně zjednodušil práci s PHP.

První a nejdůležitější věc bylo spojení webové aplikace s databází. Tento krok se provádí v konfiguračním souboru config.neon. Do konfiguračního souboru jsem vložil tuto část kódu, která zajišťuje spojení s databází. Z hlediska bezpečnosti citlivé informace nahradím symbolem hvězdičky.

```
database:
   dsn: 'mysql:host=*********.net;dbname=********vevoda'
   user: a136563_vevoda
   password: ********
   options:
      lazy: yes
```

Nyní mám spojenou aplikaci s databází a jako další věc potřebuji ošetřit přihlašování do aplikace. Pro přihlašování do aplikace musím nejdříve vyřešit načítání uživatele z databáze. Pro načítání uživatele z databáze v nette slouží php soubor UserManager.php. V tomto souboru jsou předdefinované možná nastavení uživatelů. Pro načítání uživatele vytvořím třídu UserManager.

```
final class UserManager implements Nette\Security\IAuthenticator
{
    use Nette\SmartObject;

    const
        TABLE_NAME = 'users',
        COLUMN_ID = 'id',
        COLUMN_NAME = 'username',
        COLUMN_PASSWORD_HASH = 'password',
        COLUMN_EMAIL = 'email',
        COLUMN_ROLE = 'role';


    /** @var Nette\Database\Context */
    private $database;
```

Načítání uživatelů z databáze je již hotové. Dále je potřeba ošetřit přihlašování uživatelů. Z toho důvodu si vytvořím třídu SignPresenter a funkci signInFormSucceeded. Ve třídě SignPresenter vytvořím formulář pro načítání uživatelského jména a hesla. Ve funkci signInFomrSucceeded ověřuji, jestli je zadané uživatelské jméno a heslo správné. Vstupy z formuláře poté naplním do formuláře, o který se postará bootstrap a vykreslí je v mnohem hezčí podobě.

```php
class SignPresenter extends Nette\Application\UI\Presenter
{
    protected function createComponentSignInForm()
    {
        $form = new Form;
        $form->addText('username', 'Uživatelské jméno:')
            ->setRequired('Prosím vyplňte své uživatelské jméno.');

        $form->addPassword('password', 'Heslo:')
            ->setRequired('Prosím vyplňte své heslo.');

        $form->addSubmit('send', 'Přihlásit');

        $form->onSuccess[] = [$this, 'signInFormSucceeded'];
        return $form;
    }



    public function signInFormSucceeded(Form $form, \stdClass $values)
    {
        try {
            $this->getUser()->login($values->username, $values->password);
            $this->redirect('Homepage:');

        } catch (Nette\Security\AuthenticationException $e) {
            $form->addError('Nesprávné přihlašovací jméno nebo heslo.');
        }
    }
```

V dalších krocích je potřeba zvládnout načítání záznamů z databáze a vykreslení je na úvodní obrazovku aplikace. Vykreslení záznamů požadujeme pouze, když je uživatel přihlášen. Proto načítání úvodní obrazovky omezím jednoduchou funkcí. Tato funkce přesměruje uživatele na přihlašovací obrazovku, pokud uživatel není přihlášen.

```php
protected function startup()
{
    parent::startup(); // TODO: Change the autogenerated stub
    if (!$this->getUser()->isLoggedIn() && $this->presenter->name !=
'Sign') {
        $this->redirect('Sign:in');
    }
}
```

Funkce startup se spouští při každém zapnutí webové aplikace a je umístěna v souboru BasePressenter.php, který je nadřazen všem ostatním stránkám. Tudíž není možné, aby nepřihlášený uživatel zobrazil stránky aplikace. Nepřihlášenému uživateli se vždy zobrazí pouze stránka s přihlášením.

Po vyřešení přihlašování mohu řešit načítání záznamů z databáze. Načítání probíhá pomocí funkce renderShow. Získané informace vypíši na úvodní stránce a pomocí bootstrapu upravím vzhled.

```
{foreach $reports as $report}
   <tr class="report">
    <td>{$report->date}</td>
    <td><a n:href="Report:show $report->id">{$report->name}</a></td>
    <td>{$report->type->name}</td>
    <td>{$report->status->name}</td>
    <td>{$report->priority->name}</td>
    <td>
        {if $report->user}
            {$report->user->name} {$report->user->lastname}
        {else}
            Nepřiřazeno
        {/if}
    </td>
</tr>
{/foreach}
```

Nyní mám na hlavní stránce zobrazené záznamy z databáze. Dalším požadavkem byla možnost editace těchto záznamů. Editace záznamů probíhá na nové stránce. Pro editaci si zjistím identifikační číslo záznamu, na kterém se nacházím a pomocí funkce ReportForm záznam upravím. Zobrazení opět nechám zpracovat bootstrap.

```
protected function createComponentReportForm()
{
    $form = new Form;

    $types = $this->database->table('type')->fetchPairs('id', 'name');
    $statuses = $this->database->table('status')->fetchPairs('id', 'name');
    $priorities = $this->database->table('priority')->fetchPairs('id',
'name');
    $userss = $this->database->table('users')->fetchPairs('id', 'name',
'lastname');

    $form->addText('name', 'Název:')
        ->setRequired();

    $form->addSelect('typeID', 'Typ:', $types)
        ->setPrompt('vyberte typ')
        ->setRequired();

    $form->addSelect('statusID', 'Status:',$statuses)
        ->setPrompt('vyberte status')
        ->setRequired();

    $form->addSelect('priorityID', 'Priorita:', $priorities)
        ->setPrompt('vyberte prioritu')
        ->setRequired();

    $form->addSelect('UserID', 'Uživatel: ',$userss)
```

```
        ->setPrompt('vyberte uživatele');

    $form->addTextArea('description', 'Popis:')
        ->setRequired();

    $form->addSubmit('send', 'Uložit');
    $form->onSuccess[] = [$this, 'reportFormSucceeded'];

    return $form;
}
```

Dále potřebuji vytvořit funkci pro přidání nového záznamu. Možnost přidání nového záznamu jsem umístil na navigační panel hlavní stránky. Navigační panel je zhotoven pomocí bootsrapu. Do navigačního menu jsem umístil odkaz na vytvoření nového záznamu, dále se zde zobrazuje jméno a příjmení přihlášeného uživatele a možnost odhlásit se.

```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
    <a class="navbar-brand" href="#">Bug&TaskReporter</a>
    <button class="navbar-toggler" type="button" data-toggle="collapse"
data-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false"
aria-label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarNav">
        <ul class="navbar-nav">

            <li class="nav-item">
                <a class = "nav-link" n:href="Report:create">Přidat nový
záznam</a>
            </li>
            <li style="margin-top: 10px">

                <div class="form-check form-check-inline">
                            <input class="form-check-input" type="checkbox"
id="inlineCheckbox1" value="option1">
                            <label class="form-check-label"
for="inlineCheckbox1">Bugs</label>
                </div>
                <div class="form-check form-check-inline">
                            <input class="form-check-input" type="checkbox"
id="inlineCheckbox2" value="option2">
                            <label class="form-check-label"
for="inlineCheckbox2">Tasks</label>
                </div>
            </li>
            <li class="nav-item" style="margin-top: 10px; margin-left:
400px">
                {$user->getIdentity()->name} {$user->getIdentity()-
>lastname}
            </li>

            <li class="nav-item">
                <a class="nav-link" n:href="Sign:out">Odhlásit</a>
            </li>
        </ul>
```

Jak jsem již říkal, přes navigační menu je možné se dostat k možnosti vytvořit nový záznam. Nový záznam je opět tvořen funkcí ReportForm. Vstupy této funkce jsou přepsány v souboru create.latte. Soubor create.latte vypadá následovně.

```
{block content}
<h1>Nový záznam</h1>

{form reportForm}
    <div class="form-group">
        {label name}
        {input name, class=>'form-control'}
    </div>
    <div class="form-group">
        {label typeID}
        {input typeID, class=>'form-control'}
    </div>
    <div class="form-group">
        {label statusID}
        {input statusID, class=>'form-control'}
    </div>
    <div class="form-group">
        {label priorityID}
        {input priorityID, class=>'form-control'}
    </div>
    <div class="form-group">
        {label UserID}
        {input UserID, class=>'form-control'}
    </div>
    <div class="form-group">
        {label description}
        {input description, class=>'form-control'}
    </div>
    {input send, class=>'btn btn-primary'}
{/form}
<p class="backtohomepage"><a n:href="Homepage:default">← zpět na výpis
příspěvků</a></p>
```

První řádek {block content} zobrazuje navigační menu i na ostatních stránkách. Na dalších řádcích jsou vstupy a popis vstupu k jednotlivým kolonkám vytváření nového záznamu. Poslední řádek odkazuje zpět na původní obrazovku programu.

4.3.3 Testování informačního systému

Aplikace bude nejdříve otestována pomocí přesně vymezených testovacích scénářů a poté bude nasazena na testovací provoz. Po úspěšném testovacím provozu bude aplikace nasazena na provoz reálný. Nyní ukáži testovací scénáře a poté uvedu něco o testovacím provozu.

- Testovací scénář č.1: Pokus o vstup do aplikace bez zadání hesla.

Předpoklad: Bez zadání hesla budeme pokaždé přesměrováni na obrazovku úvodní.
Postup testování: Na přihlašovací obrazovce přepíšeme adresu z přihlašovací obrazovky na domovskou.
Výsledek testu: Domovská obrazovka se nezobrazila a aplikace byla přesměrována zpět na obrazovku přihlašovací.

- Testovací scénář č.2: Přidání nového záznamu s neplatnými daty.

Předpoklad: Zadání neplatného záznamu se nepodaří z důvodu omezení zadání podmínkami.
Postup testování: Kliknutím na tlačítko přidat nový záznam se otevře možnost přidání nového záznamu do databáze. Vyplníme kolonky neplatnými daty. Například -1.
Výsledek testu: Vytvoření nového záznamu s neplatnými daty se nezdařilo.

- Testovací scénář č.3: Promítnutí smazání uživatele do databáze

Předpoklad: Smazání uživatele se projeví nemožností přihlášení se do systému. Pokud smazanému uživateli byly přidělené nějaké záznamy, přepíše se jejich přiřazení na „Nepřiřazeno“.
Postup testování: Z databáze smažeme jednoho uživatele.
Výsledek testu: Výsledek testu dopadl podle předpokládaného očekávání.

Po úspěšném otestování se přešlo na testovací provoz uvnitř firmy. Program zde byl nasazen cca 10 dní, aby se s ním uživatelé mohli seznámit. Zavedení testovacího běhu bylo úspěšné a přešlo se na reálný provoz. Systém bude dále upravován dle získaných požadavků, které uživatelé systému budou požadovat během jeho užívání, aby se systém stal co nejvíce uživatelsky přívětivý a efektivní.

5 Zhodnocení výsledků

Nově vytvořený informační systém plně nahradil systém předešlý. Podle očekávání nový informační systém zpříjemnil uživatelům práci. Další výhodou je znatelné zpřehlednění a zjednodušení uživatelských akcí. Nejpodstatnější výhodou je bezpečnost systému, kterou nám nette přináší. Systém poskytuje úplnou správu nalezených chyb. Zpřehledňuje cestu, kterou nalezená chyba či požadavek ve firmě podstoupí. Od nalezení až po přetestování chyby testerem.

6 Závěr

V teoretické části bakalářské práce bylo uvedeno vše potřebné a důležité pro tvorbu informačního systému zabývajícího se správou chyb v aplikacích. Byly ukázány druhy chyb z pohledu podniku a z pohledu programátora. Dále byly vysvětleny základní pojmy související s vývojem informačních systémů pomocí webových technologií. Technologie, které byly představeny v dalších kapitolách jsou HTML, CSS, PHP, JavaScript a využité Frameworky. Dále bylo krátce popsáno vývojové prostředí, ve kterém byl systém naprogramován.

V praktické části proběhlo získání požadavků na nový informační sytém od zadavatele. Dále bylo sestaveno možné řešení těchto požadavků, které bylo předneseno zadavateli. Po schválení řešení započal vývoj systému samotného počínaje návrhem databáze a jednotlivých stránek. Po hotovém návrhu systému začalo programování systému samotného. V práci jsem uvedl důležité části vývoje systému a jeho komunikaci s databází. Po dokončení vývoje bylo spuštěno testování. Nejprve pomocí testovacích scénářů, následně testovacím provozem. Nyní je systém úspěšně zaveden v reálném provozu a firma tento systém plně využívá ke správě chyb nebo požadavků, které jsou nalezeny testery v ostatních aplikacích, které firma vyvíjí.

Seznam použitých zdrojů

[1] IT slovník [online]. Dostupné z: https://www.it-slovnik.cz/pojem/bug

[2] THIELEN, David. No bugs!: delivering error-free code in C and C++. Reading, Mass.: Addison-Wesley, c1992. ISBN 0-201-60890-1.

[3] HARRIS, Shon. Hacking: manuál hackera. Praha: Grada, 2008. ISBN 978-80-247-1346-5.

[4] OOP [online]. Dostupné z: http://vyuka.pecinovsky.cz/animace/ladeni/index.html

[5] KLÁN, Petr a Jindřich JINDŘICH. WWW pro zelenáče. Praha: Neocortex, 2002. Bestseller for all. ISBN 80-86330-09-5.

[6] MOLNÁR, Zdeněk. Podnikové informační systémy. Vyd. 2., přeprac. V Praze: České vysoké učení technické, 2009. ISBN 978-80-01-04380-6.

[7] HLAVENKA, Jiří. Vytváříme WWW stránky. 6. aktualiz. vyd. Praha: Computer Press, 2002. ISBN 80-7226-748-5.

[8] SKLAR, David. PHP 7: praktický průvodce nejrozšířenějším skriptovacím jazykem pro web. Přeložil Jan POKORNÝ. Brno: Zoner Press, 2018. Encyklopedie Zoner Press. ISBN 978-80-7413-363-3.

[9] BORONCZYK, Tim a Martin E. PSINAS. PHP and MySQL: create-modify-reuse. Indianapolis, IN: Wiley Pub., c2008. ISBN 978-0-470-19242-9.

[10] PhpStorm:TheLightning-Smart IDE for PHP Programming by JetBrains. JetBrains: Developer Tools for Professionals and Teams[online]. Copyright©2000[cit.28.01.2019].Dostupné z: https://www.jetbrains.com/phpstorm/

[11] BRUCKNER, Tomáš. Tvorba informačních systémů: principy, metodiky, architektury. Praha: Grada, 2012. Management v informační společnosti. ISBN 978-80-247-4153-6.

[12] YouTrack: The Issue Tracking and Project Management Tool for Software Teams. JetBrains: Developer Tools for Professionals and Teams [online]. Copyright © 2000 [cit. 08.03.2019]. Dostupné z: https://www.jetbrains.com/youtrack/

[13] Introduction · Bootstrap. Bootstrap · The most popular HTML, CSS, and JS library in the world. [online]. Dostupné z: https://getbootstrap.com/docs/4.3/getting-started/introduction/

[14] Documentation | Nette Framework. [online]. Copyright © 2008, 2019 Nette Foundation. All rights reserved. [cit. 08.03.2019]. Dostupné z: https://doc.nette.org/cs/2.4/