České vysoké učení technické v Praze
Masarykův ústav vyšších studií

DIPLOMOVÁ PRÁCE

2017 Boris Nikolov

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

MASARYKŮV ÚSTAV VYŠŠÍCH STUDIÍ

Název diplomové práce:

Návrh a tvorba webové aplikace pro správu úkolů

Studijní program: Podnikání a komerční inženýrství v průmyslu

Studijní obor: Podnikání a management v průmyslu

Autor diplomové práce: Boris Nikolov

Vedoucí diplomové práce: Jiří Kaiser

Praha 2017

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
MASARYKŮV ÚSTAV VYŠŠÍCH STUDIÍ
a
VYSOKÁ ŠKOLA EKONOMICKÁ V PRAZE

Zadání diplomové práce

Školní rok: 2014/2015

Jméno a příjmení: Boris Nikolov

Studijní program: Podnikání a komerční inženýrství v průmyslu

Obor studia: Podnikání a management v průmyslu

Forma studia: kombinovaná

Téma práce: Návrh a tvorba webové aplikace pro správu úkolů

Téma práce v anglickém jazyce: Design and Development of web task management application

Zásady pro vypracování práce

Cíl práce (stručné vymezení zkoumaného problému):

Úkolem této práce je analyzovat dosavadní postupy organizace úkolů, používané ve společnosti Litea Solution s.r.o. Dále navrhnout a implementovat webovou aplikaci pro jejich správu.

Teoretická východiska:

Práce je zaměřena na analýzu stávajícího řešení organizace úkolů v týmu spolupracovníků ve společnosti Litea Solution s.r.o. Cílem je identifikovat slabá místa v managementu úkolů, přidělených jednotlivým zaměstnancům. V praktické části se budu zabývat návrhem UML schémat, potřebných k následující implementaci webové aplikace. Výsledná aplikace má za cíl, pomoci sledovat aktuální stav úkolů přerozdělených mezi zaměstnance.

Metody práce:

Analýza informačního systému a firemních procesů. Optimalizace řešení. Návrh aplikace a změna firemních procesů.

Rámcová osnova:

1. Analýza používaných metod
2. Přehled existujících SW nástrojů
3. Výběr vhodných funkcí pro potřeby společnosti Litea Solution s.r.o
4. Analýza a výběr použitelných technologií
5. Návrh aplikace
6. Postup implementace webové aplikace
7. Ekonomické zhodnocení

Základní odborná literatura:

DEACON, J.: Model-view-controller (mvc) architecture. [online] Dostupné z: http://www.jdl.co.uk/briefings/MVC.pdf, 2009

SCHWALBE, K., Řízení projektů v IT, Computer Press, 2011, Brno, 632s, ISBN 978-80-25-12882-4

NEUSTADT, I., ARLOW, J., UML 2 a unifikovaný proces vývoje aplikací, 2. vydání. Computer Press, 2007, Brno, 568s, ISBN 978-80-25-11503-9.

KANISOVÁ, H., MÜLLER, M., UML srozumitelně, 2. Aktualizované vydání. Computer Press, Brno, 2007, ISBN 80-251-1083-4.

PONKRÁC, Miloslav. PHP a MySQL : bez předchozích znalostí. Vydání první. Brno: Computer Press, 2007. 221 s. ISBN 978-80-251-1758-3, K1294.

OTTO, M.; THORNTON, J.: Bootstrap Sleek, intuitive, and powerful frontend framework for faster and easier web development. [online] Dostupné z: http://twitter.github.io/bootstrap/

ALLEN, David. Getting things done: the art of stress-free productivity. New York, N.Y.: Penguin books, c2001, xiv, 267 s. ISBN 06-708-9924-0.

Vedoucí práce: Ing. Jiří Kaiser, Ph.D.

Podpis vedoucího práce: Kaiser

Datum odevzdání zadání: 4.12.2014

Datum odevzdání práce: 6.4.2017 Schváleno na základě žádosti ze dne 14.10.2016

Podpis studenta stvrzující přijetí zadání práce: Nikolov

Toto zadání platí tři po sobě jdoucí semestry od data odevzdání zadání.

Schválení zadání DP

15.12.2014 Šimoník

Datum a podpis vedoucího programu

Kučera

Podpis ředitele MÚVS

PROHLÁŠENÍ

Prohlašuji, že jsem diplomovou práci na téma „Návrh a tvorba webové aplikace pro správu úkolů“ vypracoval/a samostatně. Veškerou použitou literaturu a podkladové materiály uvádím v přiloženém seznamu literatury.

V Praze 6. 1. 2017 …..……..……...

Bc. Boris Nikolov

PODĚKOVÁNÍ

Chtěl bych poděkovat vedoucímu této práce za jeho trpělivost a užitečné rady a kolegům ze společnosti Litea Solution s.r.o., bez jejichž pomoci by tato práce nikdy nevznikla.

Název diplomové práce:

Návrh a tvorba webové aplikace pro správu úkolů

Abstrakt:

Předmětem diplomové práce Návrh a tvorba webové aplikace pro správu úkolů je analýza stavu interních procesů, spojených s každodenními úkoly a navržení nejvhodnějších metod jejich optimalizace. Teoretická část se zabývá aktuálním stavem procesů. Další část obsahuje definice požadavků na informační systém. Dalším krokem byla rešerše existujících systémů podobného charakteru. V praktické části je návrh samotného systému. Výsledkem práce je prototyp aplikace.

Klíčová slova:

Návrh aplikace, projektové řízení, změna procesů, GTD

Master´s Thesis title:

Design and Development of web task management application

Abstract:

Subject of diploma thesis called Design and Development of web task management application is analysis of actual state of internal processes concerning everyday's tasks and proposition of the most suitable solution methods. Theoritical part of describes definitions of internal system requirements. Next part contains selection of existing implementations. Practical part describes the design of new internal system. The result of this thesis is prototype of application.

Key words:

Application design, project management, process change, GTD

Obsah

Seznam symbolů a zkratek

PHP – Hypertext Preprocessor

HTML – HyperText Markup Language

CSS – Cascading Style Sheets

HTTP – Hypertext Transfer Protocol

HTTPS – Hypertext Transfer Protocol Secure

SSL – Secure Sockets Layer

JS – JavaScript

UML – Unified Modeling Language

UX – User Experience

JQ – jQuery

FTP – File Transfer Protocol

NPV – Net Present Value

IRR – Internal Rate of Return

IT – Informational Technology

ISO – International Organization of Standardization

CMS – Configuration Management System

BPI – Business Process Improvement

BPML – Business Process Modelling Language

BPMN – Business Process Modelling Notation

1 Úvod

Společnost Litea Solution s.r.o., pro kterou je tato práce vypracována, se dlouhodobě potýká s problémy s interními procesy. Za častý problém lze považovat nekonzistentní vývojové prostředí, kdy každý z vývojářů využívá trochu jiných metodik a vývoj tak není sjednocen. Dochází také k častému opakování banálních chyb, kterým lze zamezit pomocí automatizovaných procesů. Nejedná se však pouze o vývoj samotný, ve společnosti zaznamenáváme problém především při předávání interních informací, přístupových údajů k projektu nebo kontaktních informací.

Cílem této práce je optimalizovat některé procesy tak, aby nedocházelo ke zbytečnému navyšování časové náročnosti jednoduchých úkolů, snížení nepotřebné komunikace a zefektivnění procesu zadávání úkolů, jejich kontroly a následné fakturace celého projektu.

V práci se zaměřím na změnu často se opakujících procesů vývoje webových aplikací. Především každodenních úkonů, které se opakují u většiny projektů, bývají časově náročné a lze je automatizovat a tím snížit časovou náročnost a náklady na celý projekt.

Pro úkoly a interní informace hledá společnost vhodné řešení, jakým je spravovat. Společnost se rozhoduje, zda využít existujícího řešení nebo implementovat si vlastní interní systém, který by jí pomohl zefektivnit výše uvedené problémy. Pro vybrání vhodného řešení je nutno definovat základní požadavky na systém. Především bychom měli určit, jaké vlastnosti a funkce má mít systém implementované. Ze shromážděných informací porovnáme požadavky s existujícími aplikacemi a vybereme vhodné řešení.

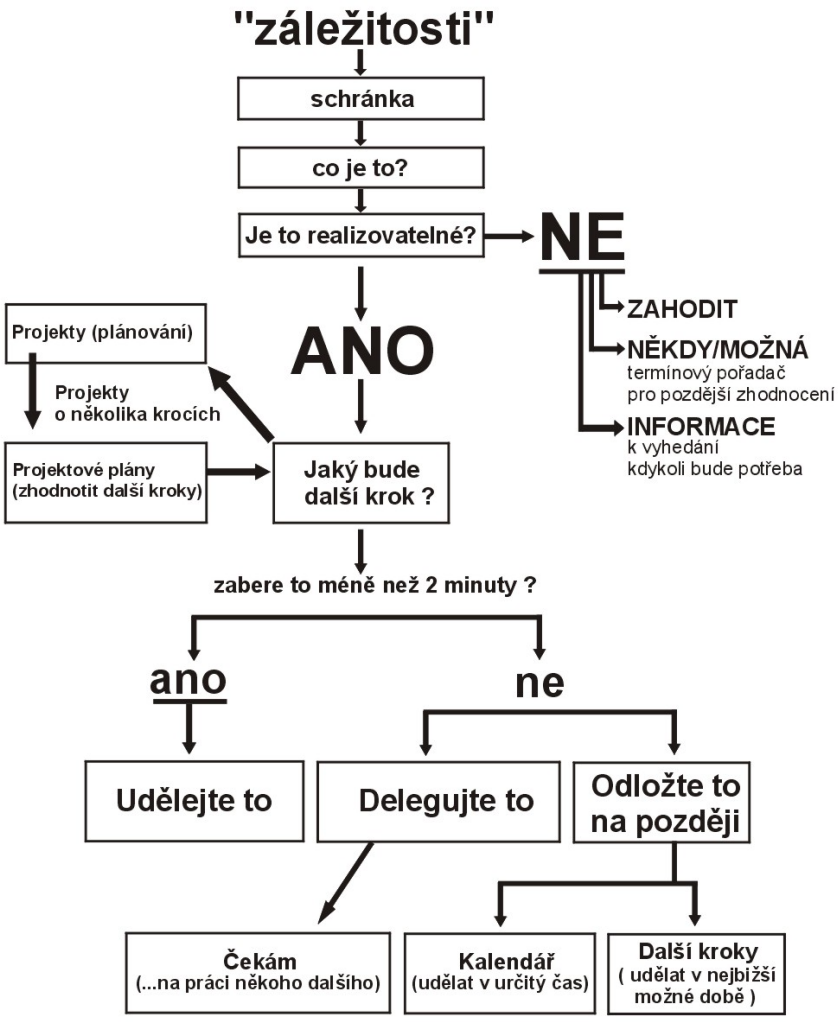
V této práci se budeme zabývat problematickými procesy, jejich optimalizací, analýzou potřebných funkcí systému a porovnáním s existujícími aplikacemi na trhu. Dle zvoleného řešení buď společnost využije již hotového produktu, nebo skončí u návrhu vlastního prototypu aplikace pro správu úkolů a vnitropodnikových informací. V takovém případě bude výsledkem této práce prototyp velmi specifické aplikace postavené na míru společnosti, který však půjde využít a aplikovat i pro účely jiné společnosti s podobným zaměřením.

2 Vymezení základních pojmů

Pro realizaci jakéhokoli projektu je důležité vymezit základní pojmy definující co možná nejpřesněji jednotlivé pojmy. V této kapitole se budeme zabývat definicemi pojmů využitých v této práci.

2.1 Get Things Done

Get Thing Done je metoda organizace práce a času vytvořená Davidem Allenem, popsaná ve stejnojmenné knize Mít vše hotovo [1]. Dle pana Allena uzpůsoben k tomu, aby si pamatoval všechny potřebné úkoly a závazky. Ve své knize, pan Allen navrhuje využití seznamů, ve kterých jsou uloženy úkoly. Díky této metodě člověk nemusí nosit všechny informace v hlavě, ale existuje nástroj, který tyto úkoly automaticky připomíná, pokud se blíží jejich termín dokončení. Cílem je, aby člověk nemusel myslet.



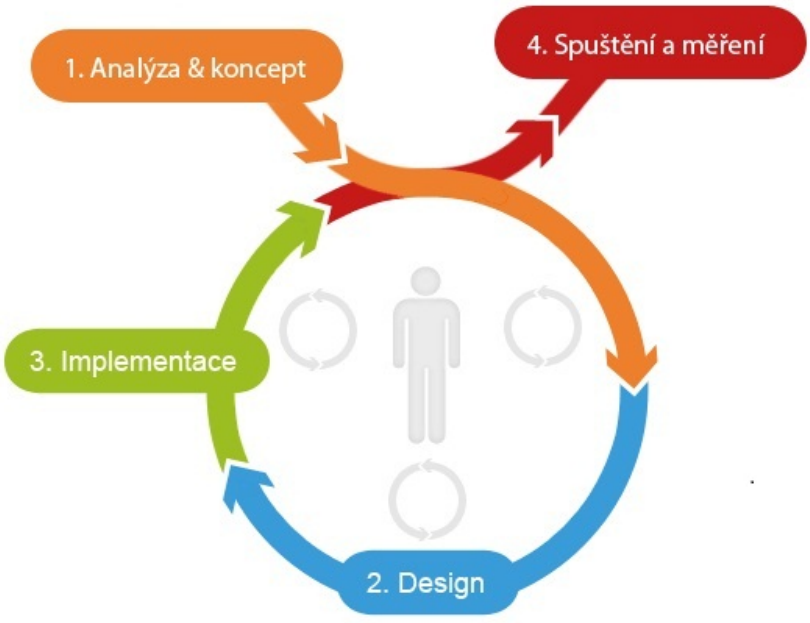
Obrázek 1: Diagram metodiky GTD [1]

2.2 Projekt

Podle definice normy ISO 10006 odst. 3.1 je *"**Projekt** je jedinečný proces sestávající z řady koordinovaných a řízených činností s daty zahájení a ukončení, prováděný pro dosažení cíle, který vyhovuje specifickým požadavkům, včetně omezení daných časem, náklady a zdroji."* [2], [3]
Směrnice IPMA ho specifikuje následovně: *„Projekt je definován jako časově, nákladově a zdrojově omezený proces realizovaný za účelem vytvoření definovaných výstupů (rozsah naplnění projektových cílů) co do kvality, standardů a požadavků."* [4]
Projekt má právě jeden začátek a jeden konec.

I přes fakt, že každý z projektů je charakteristický svými vlastnostmi, dílčími částmi, mají podle definice projektového řízení všechny projekty charakteristické společné znaky. Přestože je každý z projektů jiný, skládá se z různých detailů, většina projektů se shoduje v základních čtyřech částech.

- Analýza
- Návrh
- Implementace
- Spuštění



Obrázek 2: Cyklus projektu (zdroj internet) [5]

2.2.1 Cíle projektu

SMART

Správná definice cílů dle Doležala a kolektivu „*je jedním z klíčových faktorů úspěchu projektu. Čím vágněji je cíl definován, tím nejistěji projekt zřejmě dopadne*". SMART [6] je soubor pravidel pro navrhování cílů v řízení a plánování. Název SMART vznikl jako

akronym z počátečních písmen anglických názvů atributů cílů. V různých zdrojích je možné se dočíst několika možných překladů atributů, významově se však liší minimálně.

S – Specific – Specifický: je potřeba vědět, CO udělat.

M – Measurable – Měřitelný: abychom byli schopni určit, zda jsme dosáhli stanoveného.

A – Achievable/Acceptable – Dosažitelný: aby úkol dostala na starost osoba s dostatečnými pravomocemi.

R – Realistic/Relevant – Realistický/relevantní.

T – Timed/Time specific – Časově ohraničený, termínovaný: aby měl úkol přesně daný termín dokončení.

Dle této metodiky by každý z úkolů nebo cílů měl splňovat všechny podmínky **SMART**.

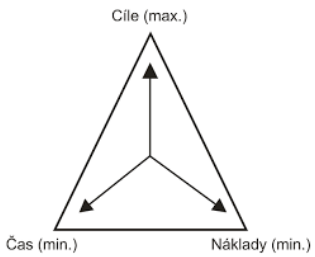
2.2.2 Řízení projektu

Řízení projektu (*Project management*) je možné popsat jako soubor metodik použitých k dosažení efektivního plánování a realizace projektu. Jedná se o osvědčený soubor postupů pro aplikaci znalostí, dovedností a nástrojů pro realizaci stanovených cílů. Nutno podotknout, že se nejedná o pěvně stanovené postupy, spíše o způsob, jakým přistupovat k jejich řešení. [7]

Cílem projektového řízení je **úspěšně realizovat projekt** a je jednodušší využít již existujících metod než znovu vymýšlet kolo. U menších projektů si možná vystačíme s tabulkou v Excelu, u komplikovanějších projektů je vhodné využít nástrojů určených k projektovému řízení.

2.2.3 Trojimperativ projektu

Trojimperativ projektu slouží k vymezení nejvýznamnějších vztahů mezi *cíli,* dostupnými *náklady a* plánovanou *časovou náročností* projektu. Základní pravidlo nám říká, že *náklady* a *čas* je potřeba **minimalizovat,** naopak *cíle* projektu **maximalizovat**. Obvykle se dá zakreslit trojimperativ pomocí trojúhelníku (viz. Obrázek 3). Bývá také charakterizován jako *„současné splnění požadavků na věcné provedení, při dodržení časového plánu, v definovaných rozpočtových nákladech“*. [8]



Obrázek 3: Trojimperativ projektu [6]

2.2.4 Fáze projektu

Předprojektová fáze

Předprojektová fáze má za cíl prozkoumat příležitosti projektu, posoudit jeho přínosy a proveditelnost. Součástí této fáze je i základní myšlenka nebo vize realizace projektu. Nejčastěji v této fázi probíhají studie a analýzy hodnotící, zda se vyplatí projekt uskutečnit.

SWOT analýza
Cílem SWOT analýzy je identifikovat silné (*Strengths*), slabé (*Weaknesses*) stránky, příležitosti (*Opportunities*) a možné hrozby (*Threats*) projektu vůči segmentu činnosti projektu nebo společnosti samotné. Jednotlivé části analýzy se zakreslují do tabulky. Cílem je odhalit, proč projekt realizovat, jaké jsou jeho výhody, možná rizika a možnosti. [6]

Je doporučeno, aby tento typ analýzy byl prováděn formou diskuze ve skupině více lidí, například v projektovém týmu. Brainstorming zajistí více pohledů na věc a analýza nebude mít individuální charakter. Na začátku analýzy je důležité stanovit časový rámec, pro který se analýza provádí.

V této fázi je doporučeno provést mnoho dalších analýz například:

- SLEPT analýza
- Analýza nákladů a přínosů (Cost-benefit analýza)
- Studie příležitostí
- Studie proveditelnosti

Na konci této fáze by měl zadavatel projektu znát dostatek informací k posouzení, zda je projekt vhodný k realizaci. V tuto chvíli je možné projekt zastavit a alokovat zdroje jiným směrem.

Zahájení/iniciace

Zahájení projektu zpravidla začíná přijetím poptávky od zákazníka. V této fázi by zákazník měl mít všechny potřebné informace k zahájení prací. Součástí zahájení projektu by mělo být upřesnění a definování cílů, procesů a nastavení kompetencí. Výsledkem zahajovací fáze by měl být dokument obsahující soupis všech technologicko-organizačních parametrů projektu. Dle pana Doležala bývá tento dokument nazýván *zakládací listina projektu.* [6]

Příprava projektu
Z předchozích částí projektu by již měly být hotovy podklady určující konkrétní zadání. Nyní je čas určit tým, který bude projekt realizovat. Vytvoří se přesné plány pro řízení projektu a harmonogram prací, podle kterého se po jeho schválení bude postupovat.

Realizace/ Implementace

Realizace projektu by měla začínat zvláštním typem jednání, označovaným jako **kick-off meeting**. Při tomto jednání dochází k rekapitulaci plánu, harmonogramu, probíhá seznámení s týmem, a především se všem oznámí, že **začíná realizace**.

U webových aplikací a prezentací může být definována realizace takto: *„Implementační fáze je zhmotnění původních myšlenek a návrhů ve fyzický produkt. Vlastní proces výroby může být u každého dodavatele jiný a může se lišit i projekt od projektu. Obecně lze do implementační fáze zahrnout programování, grafický design a kódování, testování funkčnosti apod."* [9]

Uzavření/předání

Ve fázi ukončení dochází k fyzickému i smluvnímu dokončení prací. Dochází k předání výstupu práce, podpisu a předání akceptačních protokolů a následné fakturaci.

U projektů, které vytvářejí službu, bývá tato část často prodloužena díky pracím na implementaci do nekonečně dalších funkcí. Z hlediska projektu však musí existovat fáze definující konec a u většiny projektů to bývá předání. Vývoj dalších funkcí je proto doporučeno brát jako nový projekt s vlastními fázemi.

Poprojektová fáze

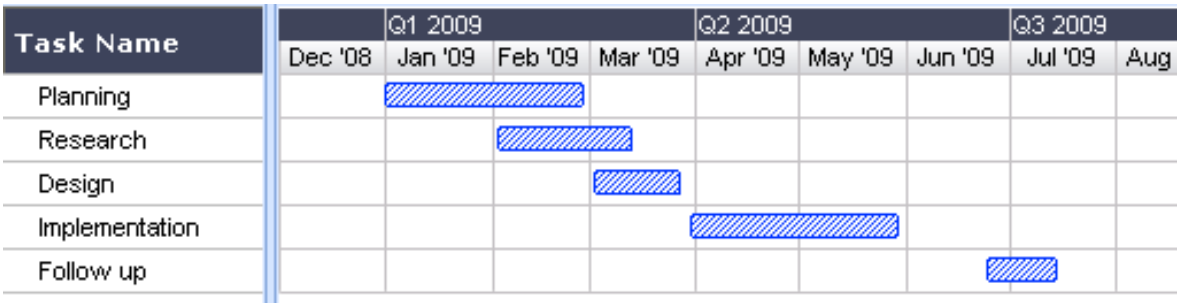
Dokončení projektu ve většině případů neznamená jeho úplný konec, při realizaci se tým setkal s novými poznatky a získal zkušenosti, ať již dobré nebo špatné. Důležité je se z každého projektu poučit a pokud možno neopakovat vzniklé chyby.

2.2.5 Plánování projektu

U každého projektu je důležité včasné dokončení dle smluvených termínů. Existují různá řešení, mezi která patří například oblíbený Gantt diagram. Ten slouží ke grafickému znázornění plánovaných aktivit. Tento diagram umožňuje uživatelům lépe si představit návaznost jednotlivých úkolů a díky zobrazení na reálné časové ose i termíny jejich potřebného dokončení.

Podle délky projektu se ve sloupcích zobrazuje časový údaj odpovídající rokům, měsícům, dnům, týdnům, nebo kratším časovým jednotkám. V řádcích jsou zobrazovány jednotlivé úkony nebo aktivity. Délka každého úkolu je znázorněna pomocí čáry odpovídající délky.

Pro jednoduchý Ganttův diagram lze využít například programu Excel. U náročnějších projektů se vyplatí sofistikovanějších aplikací vytvořených přímo pro tento účel. Ty dokážou rozlišit kapacitu, různé druhy úkolů nebo milníky projektu.



Obrázek 4: Příklad Ganttova diagramu [10]

2.3 Ekonomické zhodnocení

Pro každý projekt je důležité, aby se investice do něj vyplatila. V této podkapitole se zaměříme na základní ukazatele výnosnosti projektů.

2.3.1 Net Present Value (NPV)

Metoda požívaná pod českým názvem *čistá současná hodnota* je jedním z nejvhodnějších a nejpoužívanějších metod pro porovnávání investic. Důvodem je srozumitelný výstup a jasná hodnotící kritéria. Tato metoda reflektuje faktor času, počítá s příjmy po celou dobu životnosti projektu a užívá celkových peněžních příjmů, nejen účetního zisku.

NPV vyjadřuje, kolik peněz investor získá nad investovanou částku. Investici je vhodné realizovat, pokud hodnota $NPV > 0$. V případě $NPV < 0$ nedochází k navrácení investovaného kapitálu a investice je prodělečná.

Pokud vybíráme z více možných investic, zvolíme tu, která vykazuje nejvyšší *NPV*.

$$NPV = \sum_{i=0}^{n} \frac{CF_i}{(1+r)^i}$$

Obrázek 5: Vzorec výpočtu NPV

Kde:

NPV – čistá současná hodnota

CF_i – peněžní toky v jednotlivých letech

n – doba životnosti projektu

r – diskontní úroková míra

Slabým místem metody NPV je vysoká závislost na diskontní sazbě, kterou je složité stanovit. Při použití vyšší diskontní sazby bude dosahovat ukazatel NPV nižších hodnot.

2.3.2 Internal Rate of Return (IRR)

Vnitřní výnosové procento je ukazatel relativního výnosu (rentability), kterou projekt během svého životního cyklu poskytuje.

Číselně se rovná diskontní sazbě, při které je NPV rovna nule. [11] Pro jeho výpočet u krátkých investic menších než 2 roky lze použít iteračních metod nebo metody pokus omyl.

$$0 = \sum_{t=0}^{n} \frac{CF_t}{(1 + IRR)^t}$$

Obrázek 6: Vzorec výpočtu IRR

Kde:

IRR – vnitřní výnosové procento

CF_t – peněžní toky v jednotlivých letech

n – doba životnosti projektu

Investice dle tohoto kritéria je přijatelná, pokud je IRR větší než diskontní sazba. Čím vyšší IRR je, tím vyšší je návratnost investice.

2.3.3 Return of Investment (ROI)

ROI neboli výnosnost investice patří do kategorie poměrových ukazatelů výnosnosti. Dle server finance-management.cz porovnává čistý účetní zisk vůči velikosti investice, respektive objemu celkových aktiv nebo pasiv (bilanční suma). Často bývá označováno *výnosností aktiv* nebo *výnosnost investice*. [12]

Jedná se o nespolehlivý, avšak velmi často používaný ukazatel, užívaný v téměř každém projektu. Hodnotu ROI je potřeba brát **orientačně**. Důvodem nespolehlivosti tohoto ukazatele je čitatel zlomku, ve kterém se nachází čistý účetní zisk. Ten se často liší od ekonomického zisku a může být ovlivněn faktory jako např. odpisové metody, vypovídající schopnost účetnictví, metody užité k promítnutí nákladů, samotná délka a životnost investice a mnoho dalších.

Vzorec pro výpočet je následující:

$$ROI = \frac{\check{c}ist\acute{y}\ zisk}{aktiva\ nebo\ pasiva\ (bilan\check{c}n\acute{i}\ suma)}$$

existuje také varianta snažící se přiblížit se výpočtem reálnému peněžnímu toku:

$$ROI = \frac{\check{c}ist\acute{y}\ zisk + \acute{u}roky\ p\check{r}ed\ zdan\check{e}n\acute{i}m}{aktiva\ nebo\ pasiva\ (bilan\check{c}n\acute{i}\ suma)}$$

2.4 Sběr informací

Sběr informací je velmi důležitou částí každého projektu, kdy dochází k formulaci zadání. Vybral jsem několik metod, které jsme při psaní této práce využili.

2.4.1 Afinitní diagramy

Afinitní diagramy, někdy též označované diagramy příbuznosti, jsou nástrojem vhodným pro třídění velkého množství informací získaných z uživatelských výzkumů. Lze s jejich použitím rozdělit informace do logických skupin, což nám pomáhá pochopit jejich strukturu a vzory v názorech uživatelů. [13]

Častým způsobem použití je metoda využívající nalepovacích papírků různých barev. Všechny relevantní podklady od uživatelů jsou zaznamenány na jednotlivé papírky. Tyto podklady mohou popisovat postoje, nápady nebo navrhované funkce zabývající se zkoumanou problematikou návrhu aplikace nebo daného tématu. Po sesbírání informací se všechny papírky přilepí na připravenou plochu, nejlépe stěnu nebo tabuli, kde jsou všem členům týmu přístupné. Nápady se nechají několik dní uležet a následně může začít jejich třídění do skupin. Při rozdělování se hledají podobnosti, stejné vzory nebo témata. Pro přehlednost je každá skupina pojmenována. Vytvořené skupiny se následně strukturují tak dlouho, dokud nedojde k jejich odsouhlasení většinou členů. Tato metoda přináší možnost, jak seznámit celý tým s budoucími funkcemi, nutnými uživateli a jejich rolemi. Díky afinním diagramům se může také udávat, jakým směrem se bude aplikace vyvíjet. Tým získá zpětnou vazbu, která je velmi cenná a snižuje se šance nepochopení funkčnosti jednotlivých členů týmu. [14]

2.4.2 Myšlenkové mapy

Pod pojmem myšlenková mapa si můžeme představit vizuální reprezentaci klíčových slov, doplněných o grafické elementy a vazby mezi jednotlivými klíčovými slovy. Tato metoda je vhodná pro plánování nebo řešení problémů, ale také při třídění informací. Myšlenková mapa představuje jednoduchý diagram procesu.

Výhodou myšlenkových map je jejich jednoduchost a minimální požadavky na vybavení. K nejjednodušším mapám stačí tužka a papír, v dnešní moderní době je možné využít i aplikací pro mobilní zařízení nebo webové aplikace.

Princip je takový, že do středu mapy zakreslíme klíčové slovo, které má naši pozornost a chceme se mu věnovat. Následně začínáme s hlavními tématy a postupně je přikreslujeme okolo klíčového slova. Každé téma spojíme čárou s klíčovým slovem. Témata lze rozvíjet stejným způsobem, jako klíčové slovo.

Pro lepší vizualizaci je doporučeno využívat grafických prvků, které budou rozvíjet myšlení a asociace s daným tématem.

Mezi hlavní důvody, proč využívat myšlenkové mapy je procvičení mozku, který se tímto způsobem lépe učí a dokáže lépe asociovat. Díky rozkresleným informacím vidíme celý záměr dané problematiky před sebou. Při plánování a brainstormingu může šetřit čas. Jako nevýhodou může být její stručnost, pokud do mapy chceme vkládat dlouhý text, ztrácí svoji výhodu stručnosti.

2.4.3 Specifikace požadavků

Pokud se bavíme o specifikaci požadavků, nejedná se o jednoduchý seznam funkcí v budoucnosti implementovaných do systému. Jedná se o proces definující výsledný projekt před fází návrhu. Tato specifikace by měla popisovat celý řešený problém a způsoby, jak je vyřešit.

Fáze specifikace nastává až po základní analýze, kdy již známe dostatečné množství informací o uživatelích využívajících systém. V průběhu této fáze však stále můžeme doplňovat a upravovat požadavky klienta tak, aby se dosáhlo optimálního cíle. Nejsme tedy omezeni počátečními specifikacemi.

Po sběru informací se požadavkům přidělují priority. Ty se dají určit podle požadavků klienta, afinních diagramů nebo primárních person. Podle velikosti aplikace je potřeba určit, zda se vyplatí vytvářet scénáře. Ty mají často tendenci stát se příliš obsáhlou částí specifikace a mohou se velice rychle prodražit. [15]

2.4.4 Diagram případů užití

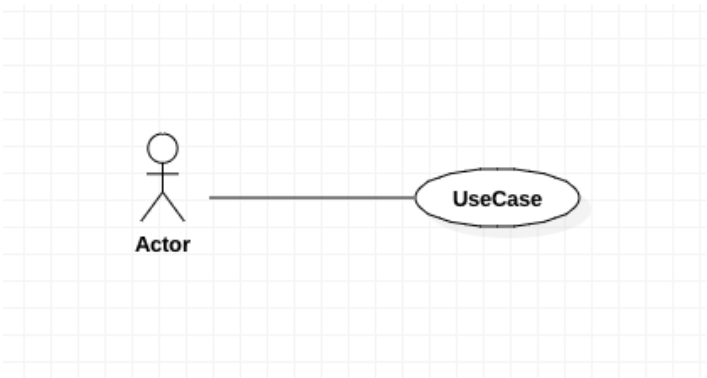
Diagram případů užití (z anglického Use Case Diagram) zobrazuje, jakým způsobem se bude systém chovat z pohledu uživatele. Cílem je popsat funkcionality systému takovým způsobem, aby se shodoval s očekáváním klienta. Důležité je zmínit, že Use Case diagramy zobrazují, co má systém dělat, neříkají nám však, jak to bude dělat. Jedná se o jeden z prvních diagramů využívaných při tvorbě softwaru. Pro vývojáře je důležité nejdříve znát informace o tom, co má systém umět a následně přemýšlet, jak toho dosáhnout.

„Každý *Use Case diagram* se skládá z případů užití *Use Case*, *aktérů* a vztahů mezi nimi.” [16]

Případ užití

Use Case je sada akcí vedoucí k dosažení určitého cíle. [17] Use case může být zobrazení úkolu, přidání komentáře nebo například otevření souboru. Jedná se tedy o definici funkcionalit navrhovaného systému. Každá z těchto akcí v sobě může obsahovat další akce, příkladem může být zobrazení a úprava úkolu. Pro jeho zobrazení musí být uživatel přihlášen, je tedy potřeba zajistit ověření uživatele, jeho oprávnění v případě změn, následné uložení změn do databáze atd. Tyto informace v případu užití nebudou. Na každý Use Case se dá nahlížet jako na černou skříňku (blackbox). Víme pouze, jaký bude vstup a očekávaný

výstup. Jak se výstupu dosáhlo, je nám skryto. Obrázek 7: Jak se zakresluje Use Case elipsa s textem značí samotný případ užití.



Obrázek 7: Jak se zakresluje Use Case

Aktér

Při navrhování Use Case se využívá takzvaných aktérů, jedná se o role komunikující s jednotlivými případy užití. Aktérem jsou jednotlivé role aplikace, může jím být například administrátor, uživatel. Nejedná se však pouze o reálné osoby, mezi role může patřit i server nebo čas, který je velmi často využíván pro automatické úlohy. V případě, že aktér využívá některý Use Case (Uživatel zobrazuje úkol), nazývá se aktivním aktérem. Do pasivního stavu se může dostat aktér v průběhu případu užití, například pokud uživatel nastaví automatické odesílání emailu, čas bude řešit, kdy se má odeslat a server bude odesílat informační email. Oba jsou pasivní uživatelé.

Stavový diagram

Stavový diagram nebo také anglicky *state diagram* je způsob grafického zápisu systému využívaného v jazyce UML. Cílem tohoto diagramu je zobrazení jednotlivých stavů a přechodů mezi nimi. V jazyce UML se dají zakreslovat tři základní prvky, jedná se o stav, událost a přechod. Smyslem stavových diagramů je zmapovat životní cyklus elementů a pochopit jeho chování v jednotlivých stavech. Vytvořené diagramy mohou následně sloužit k vytváření tříd a objektů využívaných v objektovém programování. Dalším možným využitím jsou podklady pro případy užití. Jednotlivé prvky stavového diagramu jsou znázorněny na Obrázek 8.

Stav

Stav definuje aktuální chování nebo situaci, kdy se objekt chová nějakým způsobem nebo čeká na událost. Každý stav může mít v konkrétním bodu uchované vlastnosti. Může se jednat o hodnotu, relace mezi objekty, aktuální aktivitou nebo akcí. Kdy akce je definována jako *„proces, který proběhne rychle a je nepřerušitelný“* a aktivita *„déle trvající proces, který lze přerušit“*.

Kromě počátečního a koncového bodu se v jazyce UML zakresluje pomocí obdélníku se zaoblenými rohy.

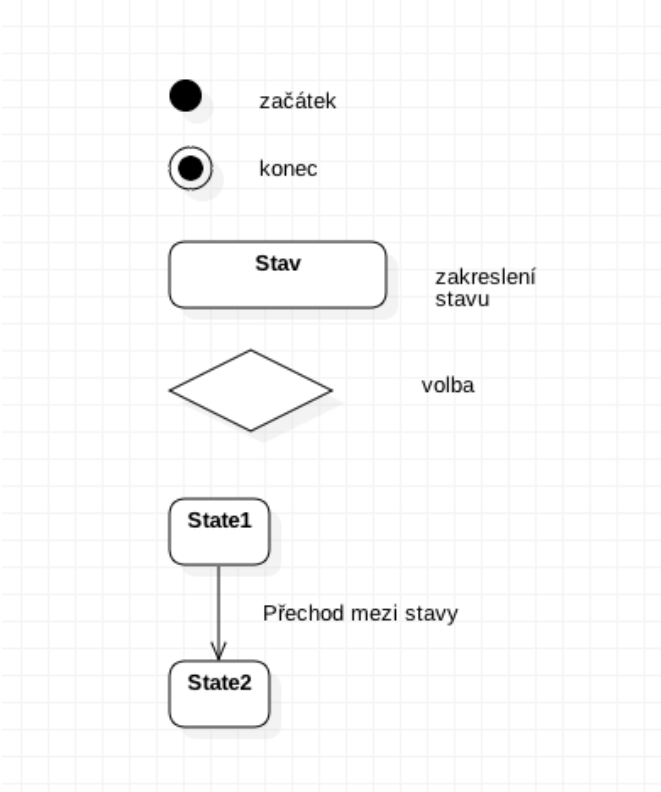
Přechod

Přechodem rozumíme propojení jednotlivých stavů. Vždy směřuje od zdrojového stavu k cílovému a jedná se o reakci na vznik nové události nebo ukončení události vykonávaného stavu.

Zakreslují se jako popisek na šipkou určující zdrojový a cílový stav.

Událost

Dle specifikace UML je událost definována následovně: *„Specifikace něčeho významného, co se stane v určitém čase a prostoru a co nemá trvání.“*



Obrázek 8: Základní prvky stavového diagramu (zdroj Autor)

3 Analýza problému a současné situace

Zadavatel diplomové práce společnost Litea Solution s.r.o. se v posledním roce začala oproti předchozím letem rozrůstat. Přibývají noví kolegové a společnost častěji využívá externích zaměstnanců pro menší úkoly, na které v daný moment nemá kapacity, dostatečné know-how, nebo se jí vyplatí daný úkol outsourcovat. S přílivem nových zaměstnanců dochází čím dál častěji k opakujícím se otázkám ohledně vnitropodnikových informací. Klasické otázky jsou: jaké jsou přístupové údaje, kde najít kontaktní informace na daného klienta, kam si zapisovat odpracované hodiny. Problematika se netýká pouze řadových zaměstnanců, ale i vedení společnosti. Pracovníci ve vedení musí vykonávat opakující se úkony typu měsíční fakturace dílčích projektů nebo stanovení platů zaměstnanců s variabilním ohodnocením dle odpracovaných hodin. Problém bývá i samotné zjištění, kolik zaměstnanec odpracoval hodin za dané období.

Vzhledem k tomu, že společnost tyto informace neudržuje na jednom místě, každý ze zaměstnanců využívá vlastního systému, který mu vyhovuje, ale není kompatibilní s ostatními zaměstnanci.

V následující části bude popsán aktuální stav procesů ve společnosti. Cílem je najít slabá místa týkající se zadávání úkolů nebo úkolů samotných, pokud je možná automatizace každodenní činnosti. Pro společnost je důležité tyto procesy zoptimalizovat a dosáhnout především snížení časové náročnosti a díky tomu i nákladů.

3.1 O společnosti

Společnost Litea Solution s.r.o. se zabývá vývojem internetových řešení na přání zákazníků. Hlavní náplní je tvorba webových prezentací, internetových obchodů, aplikací a online marketingu. Na trhu se pohybuje již více než 5 let a získala si řadu důležitých klientů, se kterými pravidelně spolupracuje: Škoda Auto, O_2 Telefonica, UPC, ČSOB Penze, Mountfield, Česká spořitelna a další.

Společnost aktuálně sídlí v centru Prahy a zaměstnává 10 zaměstnanců. V blízké budoucnosti plánuje kontinuální nábor zaměstnanců a případné otevření další pobočky v některém z velkých českých měst. Obrat společnosti v roce 2016 dosahoval téměř 4 miliónů korun.

3.2 Role zaměstnanců

V následujících odstavcích budou představeny role zaměstnanců, které aktuálně společnost zaměstnává. Z těchto rolí budou částečně vycházet i role systémové, které jsou popsány v kapitole.

Vedoucí pracovník

Vlastník společnosti je zároveň vedoucí pracovník, řídí celkový chod společnosti. Jeho náplní práce je dohlížet na hladký chod procesů, nábor nových zaměstnanců, účastní se důležitých jednání a částečně zastupuje pozici obchodníka. Shání nové zakázky, případně rozhoduje o výhodnosti příchozích nabídek. Nejčastěji komunikuje s projektovým managerem ohledně jednotlivých projektů, u ostatních zaměstnanců řeší především jejich efektivitu a spokojenost v pracovním prostředí.

Projektový manager

Aktuálně společnost zaměstnává sedm zaměstnanců na plný úvazek. Projektový manager řeší rozdělení práce mezi jednotlivé pracovníky. Kromě samotného zadávání úkolů je nedílnou součástí jeho denního programu komunikace s klienty. Ve společnosti se snažíme eliminovat počet lidí komunikujících s klienty. Tento přístup má výhodu v tom, že klienti znají styčnou osobu, protože komunikují stále se stejnou osobou. Projektový manager má zároveň přehled o všem, co se domluvilo. Tím, že problémy a nové požadavky řeší jedna osoba, nedochází ke komunikačnímu šumu a ztrátě informací, pokud by se jednalo o předávání informací přes více článků týmu. Možnou nevýhodou může být ztráta projektového managera v případě jeho odchodu ze společnosti. Jeho odchodem by společnost ztratila velkou část informací o dílčích projektech.

V případě rozrůstajícího se týmu pracovníků, tím pádem i nárůstu zakázek a klientů, bude potřeba zaměstnat dalšího projektového managera. Následně by každý z nich měl na starost pouze předem určené klienty.

Programátor

Mezi hlavní aktivity patří programování. Firemní programátoři se zaměřují na programovací jazyk PHP a Javascript. Senior programátor přijímá úkoly od projektového managera a následně je rozděluje mezi ostatní programátory dle jejich schopností a pracovního vytížení. Junior programátoři plní zadané úkoly a následně je kontrolují se senior programátorem, který s nimi provádí tzv. *code review*. Díky této kontrole dochází ke zvýšení kvality kódu a kontrole, zda kód opravdu dělá, co bylo zadáno. V některých případech se dají odhalit chyby, na které by tester neměl šanci přijít.

Grafický designer

Společnost se zabývá kompletním řešením návrhu a implementace webových aplikací, internetových obchodů a prezentací. Tvorba grafického designu UI (User Interface) podle standardů UX (user experience) spadá na grafického designera. Jeho pracovní náplň je návrh vzhledu webových stránek, analýza rozložení prvků, příprava grafických podkladů pro frontend developery a prezentace výsledků klientům. Do jeho náplně patří také návrh logomanuálu společnosti, návrh grafických komponent, jako jsou ikony, tlačítka a efekty. Při tvorbě využívá především své kreativity a grafických nástrojů od společnosti Adobe [18].

Frontend developer

Poté, co je hotov grafický návrh, je potřeba ho přenést do reálné podoby, tedy vytvořit stránky, které vypadají a chovají se podle návrhu. Tento proces má na starost frontend vývojář. Jeho náplní práce je vytvořit obsah webových stránek pomocí kódování v jazyce HTML a stylovacího jazyka CSS. Pro různé efekty využívá jazyka Javascript a jeho rozšíření. Frontend developer by v ideálním případě neměl mít potřebu komunikovat se zákazníkem. Vše by měl řešit pouze v kolektivu týmu, především s projektovým managerem, grafikem a programátory.

Marketingový specialista

Vytvořit webové stránky nebo eshop neznamená konec projektu. Produkt je potřeba rozšířit mezi potenciální zákazníky. Práce marketingového specialisty začíná s každým projektem, podílí se na analýze budoucího řešení. V analýze se zjišťují konkurenční výhody, způsoby, jak se zviditelnit a odlišit od ostatních webů. Cílem je oživit web po jeho nasazení, přilákat nové zákazníky, a především zvýšit konverzi. Jeho práce bývá u většiny projektů dlouhodobějšího charakteru. Pro programátory a designery práce končí nasazením na produkci, marketingový specialista však stále pokračuje. Tvoří například marketingové, v případě nutnosti PPC kampaně (pay per click), propojení se sociálními sítěmi. Za účelem přilákání nových zákazníků a zvětšení povědomí o produktu/značce organizuje pro uživatele soutěže. Často sám nebo za pomoci externích zaměstnanců vytváří texty pro stránky.

Externisté

Čas od času nastane situace, kdy je ve společnosti nadměrné množství práce, nedostatek kapacit, blížící se termíny odevzdání nebo se naskytne možnost zapojit se do projektu, pro nějž není k dispozici dostačující know-how. Pro tyto účely společnost využívá pomoci externistů. Ve většině případů se jedná o krátkodobou spolupráci, avšak společnost se snaží využívat kvalitních pracovníků opakovaně. Externisté jsou nejčastěji programátoři nebo designeři. Jejich komunikace se odvíjí od vykonávané role.

3.3 Používané technologie

Společnost se zabývá především vývojem webových aplikací, internetových obchodů a webových prezentací. Nejčastěji využívaný programovací jazyk je PHP s frameworkem Nette [19] nebo rozšiřování open source systémů, jako je redakční systém Wordpress [20] a internetový obchod Prestashop [21]. Všechny tyto systémy jsou psané v nějaké formě jazyka PHP. Pro správu interních aplikací se využívá několika služeb, mezi které patří malá aplikace vyvíjená samotnou společností zvaná Ježek. Tato webová aplikace slouží ke správě přístupů, klientů a je zároveň místem pro ukládání znalostní báze. Ježek je samostatná

webová stránka, které je oddělena od ostatních funkcí a slouží pouze k jednomu účelu. Jeho idea je velmi dobrá, provedení a aktivní využívání však pokulhává.

Pro větší projekty společnost využívá programu Redmine [22], zabývajícího se správou projektů. Bohužel tento nástroj není využíván pro všechny projekty a to především díky jeho složitému nastavení pro každý projekt a celkově kostrbatému uživatelskému prostředí.

Dokumenty se ve společnosti sdílí pomocí webové služby Google Dokumenty [21].

3.3.1 Vývojové prostředí

Všichni zaměstnanci využívají stejného vývojového prostředí IDE – programu, ve kterém píší kód. Volba padla na výborný nástroj PhpStorm [23] od vývojářů ze společnosti JetBrains. Ve vývojovém prostředí využívají všichni zaměstnanci stejné nastavení, což zaručuje stejné chování na všech počítačích ve firmě. Pokud tedy kolega potřebuje s něčím poradit, kterýkoli zaměstnanec, který přijde k jeho počítači, bude vědět, jak vývojové prostředí využívat.

Využívání stejného nastavení se velice osvědčilo i přes prvotní problém s naučením se jiných klávesových zkratek, než byl pracovník zvyklý.

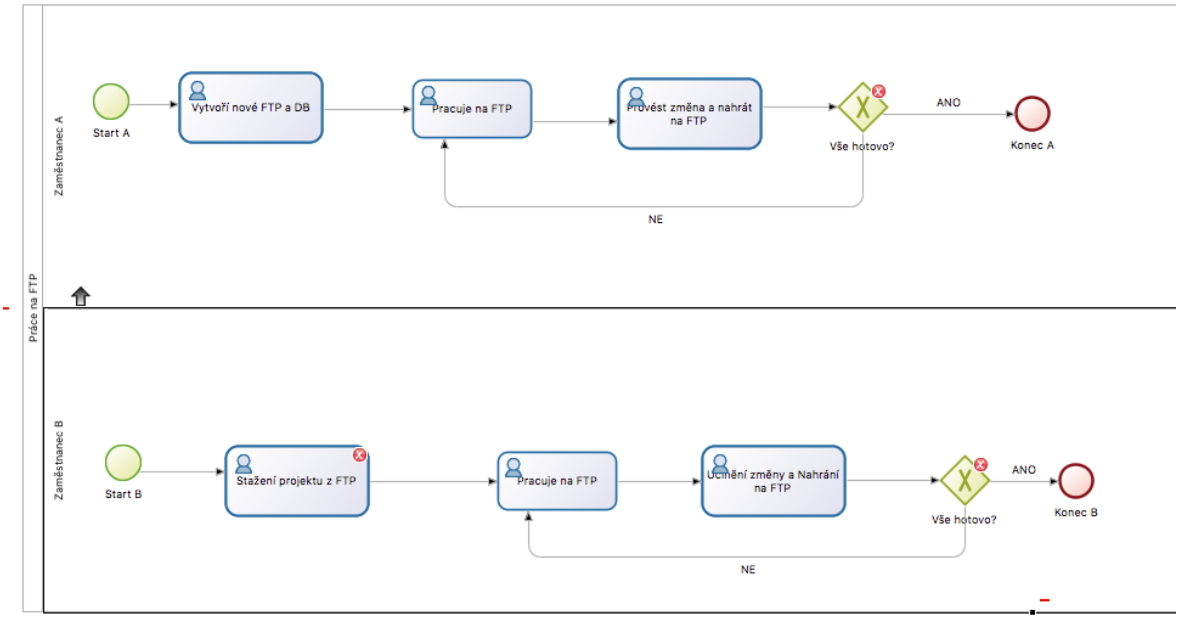
Hlavním problémem v procesu vývoje vidíme v nahrávání souborů na testovací a produkční prostředí. Postup, který se nyní používá, je velmi pomalý a náchylný na chyby lidského faktoru. To je zapříčiněno především využíváním protokolu FTP, který je pomalý, pokud se nahrává nebo stahuje velký počet malých souborů. V následujících odstavcích je popsáno, jak probíhá vývoj a nahrávání souborů.

Testovací prostředí

Společnost většinu projektů vyvíjí na testovacím serveru, ke kterému se zaměstnanci připojují pomocí protokolu FTP.

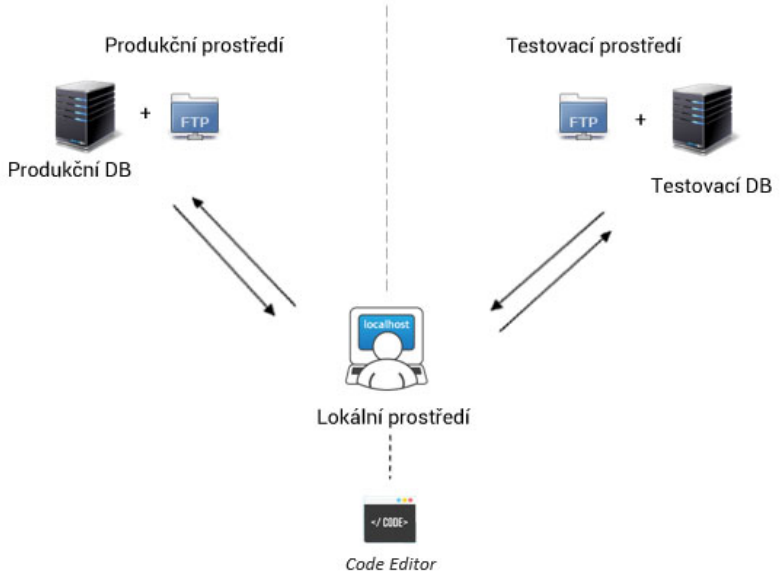
Vývoj vypadá následujícím způsobem:

1. *Zaměstnanec A* vytvoří nové FTP pro *projekt X*.
2. *Zaměstnanec A* vytvoří databázi pro *projekt X*.
3. *Zaměstnanec A* pracuje na *projektu X*, jakákoli změna se nahrává rovnou na FTP. Změny jsou vidět na testovacím serveru a je možné je kdykoli prezentovat zákazníkovi.
4. *Zaměstnanec B* si stáhne aktuální verzi *projektu X* z FTP a pracuje na svém úkolu. Při každé změně dochází k nahrání upravených souborů na FTP. Změny jsou okamžitě vidět na testovacím serveru a je možné je kdykoli prezentovat zákazníkovi.
5. Pokud vše proběhne bez problémů, *zaměstnanec A* a *zaměstnanec B* dokončí celý projekt. Obsah FTP a databáze se nahraje na *Produkční server*, kde je projekt viditelný zákazníkům či cílové skupině.



Obrázek 9: Schéma práce na FTP

Problém nastává, a to ve chvíli, kdy *zaměstnanec A* a *zaměstnanec B* budou upravovat stejný soubor ve stejný okamžik viz obrázek 9. V tu chvíli si při každém uložení souborů přepisují data nahraná na FTP. Díky tomu musí neustále řešit, kdo právě dělá na které části projektu, aby nedošlo k přepsání změn. U větších projektů, na kterých pracuje více než 2 zaměstnanci, dochází k výše zmíněnému problému častěji. I přes kvalitní komunikaci v týmu dochází k neustálému pátrání, kde vznikl problém a jak ho následně odstranit. To většinou znamená znovu vytvořit již jednou hotovou funkčnost.



Obrázek 10: Vývoj na FTP

Produkční prostředí

Společnost se snaží vyvíjet projekty na testovacím prostředí a na produkci nahrávat pouze hotové řešení. Zákazník by na produkci neměl měnit žádné soubory patřící do projektu. Jediné změny se projevují v databázi a složkách určených k nahrávání souborů.

Postup nasazení nového projektu na produkční server:

1. *Zaměstnanec* stáhne aktuální odsouhlasenou verzi souborů z testovacího prostředí pomocí protokolu FTP.
2. *Zaměstnanec* stáhne aktuální verzi databáze z testovacího prostředí.
3. *Zaměstnanec* vytvoří novou databázi na produkčním serveru.
4. *Zaměstnanec* nahraje soubory stažené v bodě 1 na produkční server.
5. *Zaměstnanec* upraví přístupové údaje do databáze vytvořené v bodě 3.
6. *Zaměstnanec* upraví údaje o projektu, jako je URL adresa potřebná k některým funkcím.
7. *Zaměstnanec* zkontroluje funkčnost nasazeného projektu.
8. *Zaměstnanec* zkontroluje nastavení ukládání do mezipaměti.
9. *Zaměstnanec* zkontroluje změny a zkontroluje správné nastavení meta tagů.
10. Pokud vše v pořádku funguje a zákazník odsouhlasí nasazení, změní se DNS záznamy a projekt je viditelný pro veřejnost.

První nahrání nebývá problém, pokud se dodrží kontrola všech podstatných údajů, které mohou být změněny oproti testovacímu prostředí. Časová náročnost nahrání projektu na produkční prostředí se pohybuje kolem 2 hodin, kdy největší část tvoří nahrávání souborů.

V případě nasazování změn na produkční prostředí je průběh velmi podobný. Ale problematičtější z hlediska pravděpodobnosti udělat chybu. Časová náročnost je zbytečně velká, pokud se nejedná o malé změny typu úprava barvy nebo velikosti písma. V tom případě se nahrávají pouze změněné soubory. Naopak při změně více souborů, implementaci větších změn nebo přidávání nových funkcí to znamená nahrávání celého projektu znovu na FTP. Jak je již napsáno výše, nahrávání trvá i několik desítek minut. Nahrání změny na produkční prostředí může trvat od 10 minut do 2 hodin.

Postup je následující:

1. *Zaměstnanec* stáhne aktuální verzi souborů z testovacího prostředí pomocí protokolu FTP.
2. *Zaměstnanec* nahraje změny v databázové struktuře na produkční prostředí.
3. *Zaměstnanec* nahraje soubory stažené v bodě 1 na produkční server.
4. *Zaměstnanec* zkontroluje funkčnost nasazeného projektu.
5. *Zaměstnanec* zkontroluje nastavení ukládání do mezipaměti.
6. *Zaměstnanec* zkontroluje změny a zkontroluje správné nastavení meta tagů.

Hlavní problém je tedy v časové náročnosti a možnosti chyby ze strany lidského faktoru.

3.3.2 Komunikace

Jedním z problémů při správě projektů je komunikace. Aktuálně každý zaměstnanec řeší problematiku s projektovým managerem nebo přímo s klientem.

S první možností společnost nemá problém, existuje osoba, které má na starost komunikaci a o tu se stará. Získané informace následně prezentuje zaměstnancům ve formě úkolů. Některé úkoly však postrádají základní formu a neobsahují všechny potřebné náležitosti odpovídající například standardu SMART, na čemž je potřeba zapracovat. Nedodržením výše zmíněné metodiky dochází k předání nedostatečných informací potřebných k dokončení úkolu. Vzniká tak opakující se cyklus, kdy se úkol zadává v několika fázích a jeho časová náročnost díky tomu zbytečně roste.

Druhou variantou je komunikace zaměstnance přímo s klientem. Tato možnost má jednu velkou výhodu: redukci komunikačního šumu. Pokud zaměstnanec komunikuje se zadavatelem úkolu, je většinou schopný zjistit v krátkém čase potřebné informace. To se může zdát jako dobrá výhoda, avšak zaměstnanec v době komunikace se zákazníkem nemůže pracovat na přidělené práci. Tento fakt zhoršuje plánování úkolů a odhadovaný termín dokončení. Další překážkou může být decentralizace informací. Každý ze zaměstnanců ví pouze část informací o projektu a chybí styčná osoba, která ví o projektu vše potřebné.

3.3.3 Paušální platby za aplikace

Společnost využívá několik aplikací, které si buď vytvořila sama nebo je platí formou měsíčního paušálu. Za poslední rok společnost zvýšila počet zaměstnanců o 100 % a počítá s náborem dalších zaměstnanců. Většina služeb je placena podle počtu uživatelů, a to buď poplatkem za uživatele, nebo je cena určena podle mezí. Například 1–5 zaměstnanců bude mít jinou cenu než 5–10 zaměstnanců. Jsou aplikace sloužící přímo k výkonu pracovní náplně zaměstnanců, například grafické programy. Avšak společnost aktuálně platí i několik aplikací, které by v ideálním případě šlo nahradit jediným řešením ve formě systému pro správu úkolů. Díky tomu by společnost mohla ušetřit na měsíčních paušálech za služby se stejnou funkcí.

3.3.4 Vlastní aplikace

Vývojáři společnosti Litea si v minulosti vytvořili několik malých aplikací. Jedná se například o správu přístupů a správu klientů. I přes existenci a korektní fungování se aplikace nevyužívá naplno. Problém je správa jednotlivých rolí uživatelů, která není kompletně implementována a díky tomu je potřeba ke každému projektu schvalovat uživatele, kteří mají možnost vidět informace ke zvolenému projektu. Často se tedy stává, že zaměstnanec hledá v aplikaci konkrétní údaje a nemůže je najít, protože nemá dostatečná oprávnění. Dalším podstatnějším problémem je neaktuálnost údajů a jejich doplňování. Některá data v aplikaci nelze dohledat, protože v aplikaci nejsou zadána, případně nejsou vložena do systému Redmine. Zadávání údajů do interní aplikace může pouze osoba s dostatečným oprávněním, kterých není mnoho. Vzhledem k těmto faktům dochází k začarovanému kruhu, kdy zaměstnanci neustále shánějí přístupové údaje k jednotlivým projektům.

3.3.5 Správa časů a fakturace úkolů

V současné době každý ze zaměstnanců využíval vlastní aplikaci pro správu časů. Ve většině případů se jedná o některou z mobilních aplikací, umožňující přidávat k vytvořeným projektům čas. Princip těchto aplikací vychází ze stopek, které zapisují čas k projektu.

Na konci měsíce nebo projektu je vždy potřeba projít časy a ty předat vedoucímu pracovníkovi, který řeší fakturaci klientům. Tento úkon je časově velmi zdlouhavý, především pro vedoucího pracovníka. Ten musí pravidelně při vystavování každé faktury obejít všechny zaměstnance, kteří pracovali na daném projektu a přesvědčit se, zda od nich má všechna data.

Vedoucí pracovník pro fakturaci doposud využíval tabulek Excel, do nichž zapisoval, který zaměstnanec, v jaký den provedl danou činnost.

Při aktuálním stavu máme podezření, že si zaměstnanci nezapisují všechen strávený čas. Často dochází ke komunikaci s klientem, kdy je zaměstnanec na jednání nebo řeší emailovou či telefonickou komunikaci a tento čas si zaměstnanec nezapíše Odhadujeme, že zaměstnanec, který nevidí souhrn stráveného času za celý den nepřemýšlí nad svým výkonem, a proto mu mohou chybět zapsané minuty či celé hodiny.

Tento problém by společnost ráda odstranil a zaměřila se na přehledné zobrazení správy odpracovaných časů. Očekáváme od této změny pozitivní promítnutí do zapsaných odpracovaných hodin a zároveň i do přesnější správy aktuálního stavu projektů.

Při malém počtu zaměstnanců se tento způsob dal využívat i přes jeho časovou náročnost. S narůstajícím počtem zaměstnanců a využíváním externistů je ovšem tento způsob nereálné zachovat. Vedoucí zaměstnanec stráví každý měsíc jednotky hodin pouze sepisováním údajů, které jsou již jednou zaznamenané do některé z mobilních aplikací.

Nemožnost nahlédnout do aktuálně odpracovaných hodin je nepraktické i pro projektové řízení. Projektový management bez opakované komunikace s kolegy nemá možnost zjistit, kolik hodin již bylo odpracováno a kolik stále zbývá.

Pouze pro některé projekty společnost využívala výše zmíněnou aplikaci Redmine, která správu času řešila na bázi úkolů. Každému úkolu je v této aplikaci možné přidat strávený čas a popis. Pro měření času je však stále nutné využít stopky nebo jiný způsob. Systém Redmine také zapisuje strávený čas ve formátu *0,00 h*. Je tedy nutné převádět všechny časy naměřené pomocí stopek a následně je zadat do systému.

3.3.6 Správa klientů

Společnost aktuálně nevyužívá žádného sofistikovaného řešení pro správu kontaktů. V případě, kdy zaměstnanec potřebuje kontakt na styčnou osobu spojenou s projektem, musí o tuto informaci požádat někoho z vedení společnosti nebo projektového managera. Tento úkon se může opakovat několikrát, dokud zaměstnanec nenatrefí na osobu vlastnící hledaný kontakt.

Hlavním problémem aktuálního řešení správy klientů je jeho časová náročnost. Pro jednoduché předání telefonního čísla nebo emailu je často potřeba kontaktovat 2-3 osoby. V lepším případě bude mít kontakt některý z kolegů, sedící vedle v kanceláři. V horším případě musí zaměstnanec počkat na kolegu, který je právě na jednání a přístupy dostanete nejdříve za několik hodin. Toto čekání může vést i ke zhoršení vztahů s klienty, kteří očekávají provedení změn v co nejkratším termínu.

Získání kontaktních údajů probíhá většinou tímto způsobem:

1) *Zaměstnanec* vyhledá kontaktní údaje ve svém emailu.
   a) *Zaměstnanec* přístupy našel.
   b) *Zaměstnanec* přístupy nenašel.
2) *Zaměstnanec* musí kontaktovat některého *nadřízeného*.
   a) *Nadřízený* má hledané přístupy a předá je *zaměstnanci.*
   b) *Nadřízený* nemá hledané přístupy a *zaměstnanec* se poptá u jiného *nadřízeného.*

Zaměstnanci tráví hledáním kontaktů čas, který by mohli věnovat placené činnosti.

3.3.7 Správa přístupových údajů

Velmi podobný problém popsaný v kapitole 3.3.6 se opakuje také u správy přístupových údajů. Zaměstnanci společnosti nejčastěji potřebují přístup do webové administrace stránek, přístup k souborům prostřednictvím protokolu FTP a přístup do administrace databáze. Bez těchto přístupů není zaměstnanec schopen vykonat požadované úkoly.

Společnost kdysi vytvořila malou aplikaci, které měla tento problém eliminovat. V tomto programu však nejsou zapsány všechny přístupy, případně k nim nemají zaměstnanci přístup.

Získání přístupů probíhá většinou tímto způsobem:

1) *Zaměstnanec* se vyhledá přístupy v interní aplikaci.
   a) *Zaměstnanec* přístupy našel.
   b) *Zaměstnanec* přístupy nenašel.
      i) Přístupy v interní aplikaci nejsou.
      ii) Přístupy v interní aplikaci jsou, ale zaměstnanec nemá oprávnění k danému projektu.
   c) *Zaměstnanec* musí kontaktovat některého *nadřízeného*.
      i) *Nadřízený* má hledané přístupy a předá je *zaměstnanci.*
      ii) *Nadřízený* nemá hledané přístupy a *zaměstnanec* se poptá u jiného *nadřízeného.*

Jak vypadá schéma podobného problému je znázorněno na obrázku

Výše popsaný cyklus úkolů se může opakovat i několikrát denně a ač se to nezdá, každé hledání nových přístupů trvá několik minut, v horším případě i několik desítek minut v případě 2 a více zaměstnanců. Tento problém vede ke snížení efektivity zaměstnanců.

3.3.8 Správa projektů

Jak již je popsáno výše, společnost využívá aplikace Redmine [22], kterou však nevyužívá pro všechny projekty. Jedním z důvodů, proč tuto aplikaci nevyužívá pro každý projekt, je její nepřehlednost. Někteří ze zákazníků se ani po krátkém školení v aplikaci neorientují a zaměstnanci společnosti se tak stávají spíš technickou podporou. Nejčastěji musejí řešit situaci, kdy zákazník neví, jak zadat chybu nebo nový požadavek do systému.

Pro zbytek projektů se používají nesofistikované způsoby pomocí sdílených dokumentů prostřednictvím služby Google Drive [24]. Správa úkolů je řešena pomocí formátovaných excelových tabulek a informace o projektech bývají v textových dokumentech společně se zadávací dokumentací.

3.3.9 Fáze projektu

V této fázi je nutno podotknout, že každý projekt se může lišit vzhledem k obsahu projektu. Někteří zákazníci přicházejí s nedostatkem informací a dochází k situacím, kdy musíme analyzovat kompletní zadání a vymýšlet funkce systému, protože zákazník předem neví, co chce. V následujících podkapitolách se budu zabývat nejčastějším typem projektů, což jsou webové stránky na míru.

Zahájení/iniciace

Zahájení projektu zpravidla začíná přijetím poptávky od zákazníka. Většinou se jedná o úvodní fázi projektu, kdy nejsou známé všechny potřebné detaily potřebné pro budoucí vývoj. Zákazníkovi je podle dostupného zadání poslána oficiální nabídka obsahující předběžnou kalkulaci s rozpisem odhadovaného času pro jednotlivé části projektu a zajímavé reference projektů s podobnými funkcemi, které v minulosti společnost řešila.

Následuje stadium projektu, kdy se čeká na vyjádření zákazníka. Pokud je zákazník spokojen, může se začít s analýzou funkcí. V případě, že zákazníka nabídka nezaujala, dochází k ukončení projektu a jeho archivace.

Analýza a koncept

U málokterého projektu tvorby webových stránek je předem daná funkčnost. S klientem se uskuteční jednání, kdy probíhá diskuze o potřebných funkcích, účelu webových stránek a možných vylepšeních z naší strany. V případě nutnosti se tato jednání opakují, dokud nejsou obě strany spokojené se zadáním. Výsledkem této fáze projektu je dokument obsahující analýzu a zadání, podle kterého se následně bude finální webová prezentace vyvíjet. Součástí tohoto dokumentu jsou i wireframe nákresy znázorňující rozložení grafických prvků na stránkách, podle wireframe se následně bude tvořit grafický návrh. V této části projektu se také stanovuje finální cena, obě strany již mají dostatek podkladů a je tedy možné určit cenu pro jednotlivé části.

Této části projektu se účastní především projektový manager a vedoucí pracovník zabývající se především stránkou organizační, řešící termíny, časovou náročnost a nutné dokumenty

spojené s vývojem. Marketingový specialista spolu s grafickým designerem se zabývají návrhem funkcí a předběžným návrhem vzhledu – výše zmíněné wireframe. I v této části může dojít k ukončení projektu, například z důvodu nespokojenosti klienta s cenou nebo navrhovaným řešením.

Návrh

Po odsouhlasení a podepsání všech potřebných dokumentů spojených s vývojem nastává část návrhu vzhledu. Grafický designer vytváří vzhled podle wireframe návrhů vycházejících z analýzy. Tato část většinou probíhá v několika iteracích, kdy se ladí detaily ke spokojenosti zákazníka. Designer je však omezen předchozím zadáním, kterého se snaží držet, aby nedocházelo ke zvyšování časové náročnosti projektu.

Dle zadání z analýzy je na vytvoření grafického návrhu stanoven časový budget, do kterého by se měl návrh zvládnout. U některých klientů bývá problém s odsouhlasením grafických prací a je nutno tento budget v průběhu vytváření návrhu zvýšit. Projektový manager a designer tedy průběžně konzultují strávený čas s klientem.

Nutno podotknout, že u některých projektů obsahujících velké množství funkcionalit se návrh může prolínat se samotnou implementací. Programátoři již mohou připravovat administrační, databázovou část a některé z funkcí, zatímco designer ladí vzhled s klientem.

Realizace

Fáze realizace nebo také implementace je u většiny projektů nejdelší část. V této fázi se realizují funkce dohodnuté v analýze. Nejvíce práce tu bývá pro frontend developery, programátory a projektového managera. Po dokončení realizace probíhá několik iterací testování, kterého se účastní i zákazník.

Všechny změny si může zákazník prohlédnout na testovacím prostředí, kde je vyvíjeno.

Odevzdání/předání

Po odstranění všech nedostatků přichází fáze akceptace. V případě spokojenosti klienta dochází k nasazení vytvořené webové prezentace na produkční prostředí. Nastává kontrola, zda vše funguje, jsou-li nastaveny správné parametry i pro produkční prostředí. Po odsouhlasení dochází ke spuštění provozu.

Pokud nejsou nasmlouvané další funkce, dochází k ukončení projektu, v opačném případě se může pokračovat ve vývoji nových funkcí nebo marketingovým úkonům spojeným s vývojem.

Archivace

V případě ukončení prací i spolupráce dochází k archivaci projektu. To znamená skrytí projektu z interního systému Redmine [22] případně archivace souborů na cloudovém uložišti.

3.3.10 Postupy zadávání úkolů

Jedním z problémů projektového řízení je decentralizace úkolů. Každý z úkolů je zadáván jiným způsobem. Díky několika aplikacím, které společnost využívá, dochází k zadávání úkolu/ů do systému Redmine, ústně nebo pomocí emailu.

Nejběžnější zadání úkolu probíhá ústně. Pokud úkol splňuje všechny předpoklady pro dokončení, nejedná se o problém za předpokladu, že se jedná o jeden úkol. V případě více než jednoho jednoduchého úkolu dochází ke komunikačnímu šumu a osoba vykonávající požadované úkoly se může v informacích ztrácet. Vzhledem k faktu, že podrobnosti k těmto úkolům nejsou nikde sepsány, dochází k opakované komunikaci a ujišťování se, zda je zadání pochopeno správně. Nikde není uchován aktuální stav všech potřebných úkolů a spoléhá se na projektového managera nebo zadavatele úkolu, že si bude pamatovat, co je potřeba dodělat. Priority se dají určit, ale v případě jejich změn je komplikované přesunout se z jednoho úkolu na druhý. Další problém se objevuje v případě přeřazení úkolu na jinou osobu, v ten okamžik dochází k opakovanému vysvětlování celého úkolu, což zvyšuje časovou náročnost.

Zadání emailem je druhou nejčastější variantou. V případě detailního popisu úkolu se jedná o ideální případ, jak zadat úkol. Komunikace u nejasností lze řešit rovnou v emailovém klientovi. Bohužel tyto úkoly nejsou nikde archivovány. Dohledávání informací zpětně je komplikované a časově náročné. Samotné úkoly nejsou nikde uchovávány a díky tomu není vidět aktuální stav. Úkol lze přenést na jiného zaměstnance, ale nikde není zaznamenáno, kdo daný úkol aktuálně vykonává.

U některých klientů se využívá programu Redmine, do kterého mají přístup i klienti. Uživatel této aplikace může vytvářet nové úkoly a ty přidělovat jiným uživatelům. Ke každému z úkolů lze přidat přílohu a detailní popis. Zákazník nebo projektový manager tedy mohou zadat úkoly do tohoto sytému, ze kterého si informace přebere přiřazený zaměstnanec. V případě nutnosti lze využít implementované funkce komentářů. Bohužel ne všechny projekty tuto aplikaci využívají. Nastavení projektů je zdlouhavé a někteří klienti mají problém se v aplikaci orientovat.

Společnost hledá způsob, jak organizovat úkoly způsobem šetřícím čas a redukujícím nutnost opakovaného vysvětlování zadání.

4 Navrhované změny

V následující kapitole popíši jednotlivé změny, které nám po konzultaci se společností Litea Solution s.r.o. přišly důležité a bylo by dobré je začít využívat. Nejedná se pouze o funkce navrhovaného systému, ale také o změnu procesů související s každodenními úkony, které jsou časově náročné a čas na nich strávený bylo možné zkrátit nebo eliminovat.

V kapitole 7.2 následně určíme důležitost pro každou z funkcí a dle zvolených priorit budeme postupovat při následném návrhu a implementaci systému.

4.1 Změna procesů

Po prozkoumání výsledků analýzy aktuálního stavu jsme se se společností shodli na změně některých procesů týkajících se vývoje a správy projektů. V následujících podkapitolách seznámím čtenáře s odsouhlasenými návrhy změn procesů.

4.1.1 Komunikace

Zásadní změna v komunikaci bude nastavení styčné osoby pro každý projekt. Tato osoba bude u projektu, pokud to bude možné, od samého počátku. Díky tomu bude znát podrobnosti, které se v průběhu jednání dohodly. Stejná osoba, většinou se bude jednat o projektového managera, bude úkoly přerozdělovat mezi zaměstnance dle jejich kompetencí a zkušeností. Stejná osoba bude vždy komunikovat s klientem. Ostatní zaměstnanci, pokud nebude potřeba, s klientem komunikovat nebudou a budou se tak moct věnovat své práci.

Tato změna odstraní problém, kdy zaměstnanci místo řešení zadaných úkolů museli komunikovat s klientem. Zároveň bude existovat osoba, která zná všechny potřebné informace o projektu.

4.1.2 Zadávání všech úkolů do centrálního systému

Projektový management bude přidávat všechny úkoly do interního systému. Každý z úkolů bude podléhat pravidlům metodiky SMART [25]. Všechny úkoly musí být co možná nepřesněji specifikované, aby odpadla zbytečná duplicitní komunikace. Při zadávání úkolů bude brán zřetel na metodiku GTD [1], zaměstnanci proto nebudou muset nosit všechny informace o tom, co mají dělat v hlavě a místo toho budou využívat systému, kde tyto informace jsou uložené. Díky požadované možnosti přidat k úkolům a projektům přílohy dojde ke zjednodušení vysvětlování chyb pomocí obrázků a snímků obrazovky.

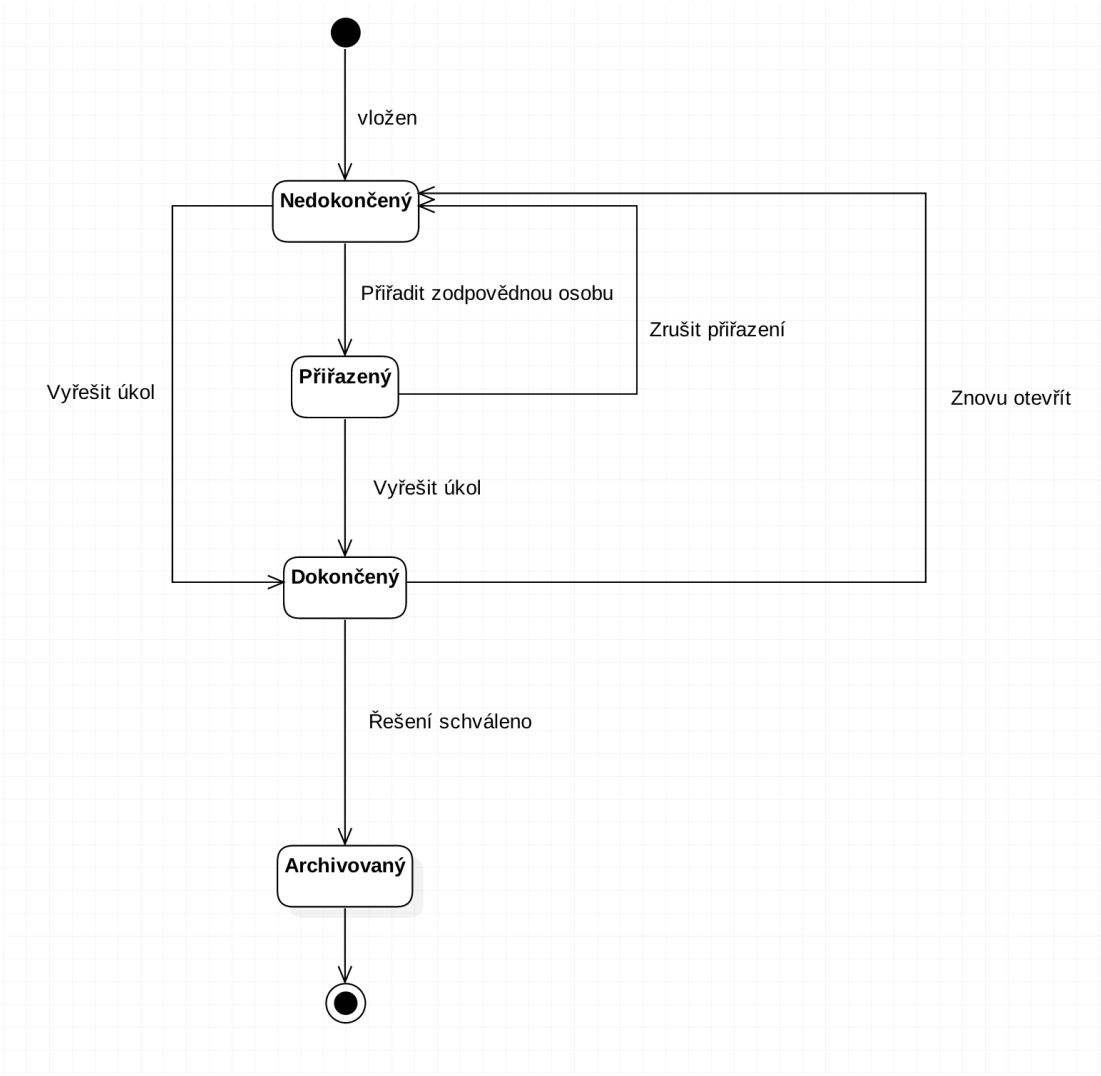
Proces zadávání úkolu:

Zadávání úkolů by mělo probíhat pomocí interního systému. Nejčastěji bude zadávat úkoly projektový manager, ale některé úkoly si mezi sebou budou zadávat i ostatní zaměstnanci. Proces vkládání úkolů by měl umožnit předávat informace získané od klienta do systému, kde je možné je přiřadit zodpovědné osobě. Dle možností systému by úkol měl obsahovat

všechny potřebné informace a zadavatel úkolu by verbálně pouze upřesňoval zadání. V případě malých úkonů je možné verbální komunikaci odstranit kompletně.

1. *Zaměstnanec A* získá informace od *klienta*.
2. *Zaměstnanec A* vytvoří nový úkol v interním systému.
3. *Zaměstnanec A* vyplní potřebné informace pro dokončení úkolu.
4. *Zaměstnanec A* nebo jiná zodpovědná osoba přiřadí úkol vykonávající osobě – v tomto případě *zaměstnanci B*.
5. *Zaměstnanec B* vykoná požadovaný úkol.
6. *Zaměstnanec B* přiřadí úkol ke kontrole *zaměstnanci A* nebo jiné zodpovědné osobě.
7. Kontrola úkolu *zaměstnancem B*.
    a. Úkol je v pořádku, je označen stavem dokončen.
    b. Úkol není v pořádku dokončen, doplní se informace a přeřadí se zpět na *zaměstnance B*.

V případě, že nastane situace v bodě 7.a, opakují se body 5–7, dokud úkol není dokončen v pořádku.



Obrázek 11: Stavový diagram úkolu

4.1.3 Zapisování časů a fakturace

V kapitole 3.3.5 jsme řešili hlavní problémy zapisování časů do systému nebo excelové tabulky v původním řešení. V novém systému by tyto problémy měly být zjednodušeny. Níže jsou popsané nevýhody původního řešení a výhody nového řešení.

Původní řešení:

- Zaměstnanci si museli zapisovat hodiny do své aplikace.
- Zaměstnanci museli přepsat hodnoty do excelové tabulky.
- Zaměstnanec musel kontrolovat, zda sedí hodiny v aplikaci oproti hodinám v tabulce.
- Projektový management musel průběžně kontrolovat zaměstnance, zda mají vše zapsané.
- Vedení společnosti muselo kontrolovat, zda jsou hodiny v rozsahu naceněného projektu. To bylo problematické ve chvíli, kdy všechny hodiny nejsou zapsané.

Zapisováním každého úkolu do vlastní aplikace a následným přepisováním do excelové tabulky strávil zaměstnanec řádově 5–7 minut.

Problémy spojené s fakturací:

- Vedoucí pracovník musí obejít všechny kolegy, zda je možné fakturovat (prodlužuje celý proces).
- Vedoucí pracovník musí obejít všechny kolegy, zda jsou všechny hodiny zapsány v systému (prodlužuje celý proces).
- Vedoucí pracovník musí projít zapsané hodiny. Zjistit, které z úkolů je možné fakturovat jako více práce a které v původním rozsahu projektu.
- Vedoucí pracovník musí vyfakturovat všechny hodiny.
- Vedoucí pracovník musí sepsat soupis práce – vybrat z Excelu.
- Vedoucí pracovník může vystavit fakturu.

Vedoucí pracovník odhaduje pracnost fakturace projektu mezi 40–60 minutami. U velkých projektů s pracností v řádů několika desítek až stovek hodin časová náročnost značně narůstá.

Nové řešení:

Zaměstnanci zapisují časy přímo k vytvořenému úkolu nebo projektu s popisem práce a možností vyplnění štítků (tagů), označujících o jaký typ práce se jednalo.

Pro projektový management má centrální uchovávání informací o strávených časech výhodu, že vždy vidí aktuálně odpracované hodiny.

Díky tomuto bodu budou všechny časy spojené s projektem zapsané v systému. V případě fakturace si vedoucí pracovník zobrazí časy k danému projektu. V případě, že některé práce není možné fakturovat – například oprava chyb, je možné zkopírovat celou tabulku do excelu a nepoužitelné hodnoty vymazat. Poté vedoucí pracovník může vystavit fakturu a odeslat ji klientovi.

Odhadovaná pracnost fakturace je okolo 10-15 minut. U velkých projektů může být o trochu větší. Záleží, zda zaměstnanci dodrželi správné vyplnění štítků označující typ práce.

V další verzi systému bychom rádi vytvořili modul pro samotnou fakturaci, kdy se z vybraných hodin vygeneruje faktura. Fakturační informace a hodinové sazby je možné brát ze systému. S fakturací samotnou je spojeno více komplikací, například napojení na účetní program Pohoda.

4.1.4 Vývoj pomocí verzovacího systému Git

V kapitole o aktuálním stavu 3.3.1 jsem popisoval vývoj aplikací na straně serveru za využití protokolu FTP. Nyní bych rád popsal výhodu využití některé ze služeb Version Control System [26]. Zaměřím se na systém Git [27], který je vhodný pro kontinuální vývoj aplikací a nabízí mnoho užitečných funkcí.

Verzování neboli ukládání aktuálních verzí souborů je velmi často využívanou službou.

Cílem je mít uložené všechny části aplikace s možností zobrazit si změny mezi každým uložením. Uložením u systému Git se rozumí commit souboru. Každý commit je zaznamenán a k této změně je možné se vrátit. Pokud například dojde ke změně v projektu a je potřeba tuto změnu vrátit, za normálních podmínek by bylo potřeba najít zálohu a tu

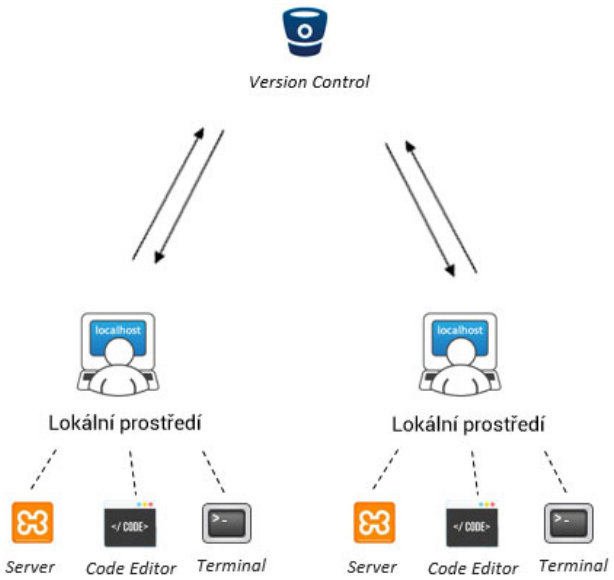
vrátit nebo naprogramovat celé řešení znovu. Díky možnostem verzovacího systému Git můžeme vrátit zpět celý commit.

Repositář je uložiště, do kterého se nahrávají soubory, se kterými se v systému Git pracuje. Většinou existují lokální kopie repositáře u každého z programátorů pracujících na projektu a hlavní repositář je uložen na vzdáleném serveru (*remote*). Uživatelé změny odesílají do vzdáleného repositáře, odkud si je mohou stáhnout ostatní kolegové.

Větve neboli *Branches* umožňují vývoj nových funkcí bez rizika zásahu do aktuální verze projektu. V případě implementování nové funkcionality si programátor vytvoří novou větev s vlastním názvem. Systém vytvoří kopii aktuálního stavu, na které je možné pracovat. Po dokončení a otestování může uživatel zažádat o spojení větví, čímž dojde ke spojení aktuální verze s vytvořenou větví.

Výhody:

- Jednoduchost
- Přidané funkce
- Možnost vrátit se zpět v historii kódu
- Možnost vytváření nových funkcí bez omezení aktuálního programu
- Úspora času



Obrázek 12: Schéma použití Version Control System [zdroj: autor]

4.1.4.1 Časová úspora

Vytvořili jsme testovací projekt postavený na redakčním systému Wordpress a otestovali, jak dlouho trvá stáhnout si aktuální projekt z repositáře a FTP serveru. Oba testy proběhly na stejném internetovém připojení za stejných podmínek. Velikost testovaného projektu byla 127 MB, obsahoval 5282 souborů.

Stažení projektu ze serveru FTP trvalo 7 minut 7 vteřin, tedy *427 vteřin*.

Stažení projektu z repositáře služby Bitbucket trvalo *29 vteřin*.

Využitím repositáře bylo více než 10x rychlejší než stahování celého projektu z FTP.

Průměrně zaměstnanec stahuje 2–3 projekty kvůli menším změnám. V původním řešení musel stahovat vždy celý projekt, aby si mohl být jistý, že má všechny potřebné soubory. Můžeme tedy napsat, že při každé úpravě strávil minimálně *7 minut*, strávených čekáním na stažení souborů.

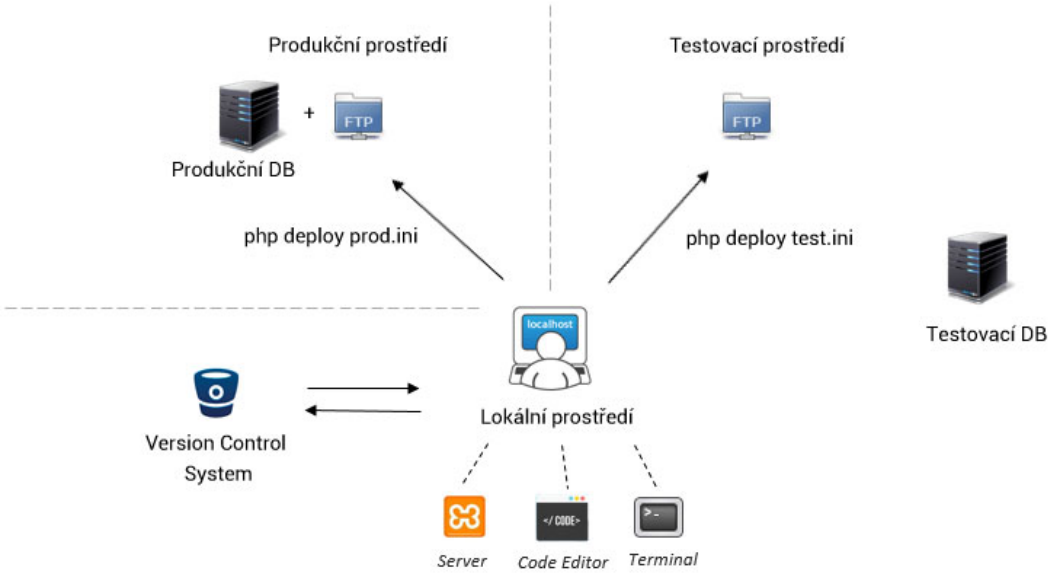
V novém řešení stažení změn trvá řádově několik vteřin, stahují se pouze změněné soubory.

U velkých projektů typu internetových obchodů je nutné stahovat mnohem větší počet souborů a velikost typického projektu se pohybuje mezi 500 MB až 2 GB. Stažení takového projektu přes FTP trvá 30 minut – 2 hodiny.

Nahrávání dat na FTP provádíme minimálně jednou denně. Budeme počítat, že alespoň jeden zaměstnanec bude stahovat projekt z repositáře. Počítáme-li časovou úsporu *7 minut*, tak za měsíc jeden uživatel mohl více než *2 hodiny* pracovat na jiných úkolech. V případě stahování pouze změn je rozdíl ještě větší, protože se stahují pouze změny v souborech.

4.1.5 Vývojové prostředí

Pro vývoj se budou používat stejné programy jako doposud. Změny budou především v procesech, jak se bude vyvíjet a kde. Hlavní rozdíl bude ve vývoji na lokálním prostředí a využití verzovacího systému Git.



Obrázek 13: Schéma vývojového prostředí [zdroj: autor]

Lokální prostředí

Lokálním prostředím označujeme vývoj na vlastním počítači, kde jsou umístěny všechny potřebné soubory. Není tedy potřeba se připojovat k žádnému serveru a při každé změně je znovu nahrávat, aby si zaměstnanec mohl prohlédnout změny.

Vývoj na lokálním prostředí funguje následujícím způsobem:

1) *Zaměstnanec* si z již vytvořeného repositáře Git stáhne aktuální verzi projektu.
2) *Zaměstnanec* na svůj počítač nahraje aktuální verzi databáze.
3) *Zaměstnanec* zapracuje požadovaný úkol.
4) *Zaměstnanec* změny nahraje na repositář, v případě konfliktů v souborech zajistí správné spojení souborů.
5) Po odsouhlasení *projekt managerem* nahraje *zaměstnanec* pomocí deploy scriptu celý projekt na testovací prostředí.
6) Pokud *zákazník* úkol schválí, je nahrána nejaktuálnější verze (shodná s verzí na lokálním prostředí) na produkční server.

Testovací prostředí

Testovací prostředí slouží pouze k prezentaci pro zákazníka. Veškeré změny jsou prováděny pouze v repositáři Git a na test jsou nahrávány až dokončené bloky práce. Pro nahrávání na testovací prostředí bude využíván deploy script [28], [29], který nahrává pouze upravené soubory. Zaměstnanci tím ušetří čas a odstraní se možnost zapomenutí nahrání některého z potřebných souborů na testovací server.

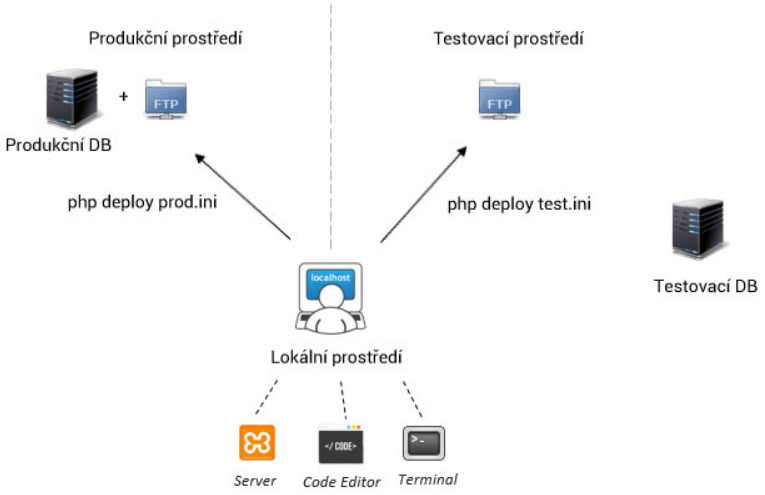
Produkční prostředí

Produkční server obsahuje vždy nejaktuálnější verzi schválenou zákazníkem. Slouží pouze pro prezentaci veřejnosti a klientům našeho zákazníka. Nahrávání na produkční prostředí funguje stejným způsobem jako na testovací.

FTP Deploy

V předchozí části jsem popsal výhody verzovacího systému Git [27]. V některých případech však není možné nahrávat soubory na hosting pomocí této služby a je možno využít pouze standardního protokolu FTP. Pro většinu projektů tedy budeme využívat PHP funkce od Davida Grudla [29] a. Tato funkce nahrává na server pouze změněné soubory, díky čemuž šetří čas i datovou náročnost. Zároveň tím snižuje chybovost lidského faktoru, tedy například zapomenutí nahrát některý ze souborů.

Další z výhod tohoto scriptu je možnost spustit ho v testovacím módu, kdy script vypíše informace o souborech, které se budou nahrávat a mazat. Díky tomu lze předejít nechtěným změnám na testovacím nebo produkčním prostředí.



Obrázek 14: Schéma nahrávání změn na FTP

4.1.5.1 Časová úspora

Použili jsme stejný testovací projekt, jako v kapitole 4.1.4.1 a změřili jsme, jak dlouho potrvá nahrát soubory na server. Jedná se o úkon, který musejí zaměstnanci provádět pravidelně, pokud potřebují ukázat změny zákazníkovi. Stejný postup jsme využili s použitím výše zmíněného PHP Deploy scriptu.

První nahrávání souborů na server:

- Nahrání souborů pomocí programu Filezilla trvalo *6 minut a 23 sekund*.
- Nahrání pomocí PHP Deploy scriptu trvalo *6 minut 55 sekund*.

Při prvním nahrávání je lehce rychlejší klasická metoda nahrávání souborů přes FTP oproti využití scriptu.

Nahrávání změn:

Pro další pokus, jsme změnili obsah 30 souborů a přidali nových souborů. Tyto soubory je potřeba nahrát znovu na server.

Při použití klasické metody FTP by bylo nejjednodušší nahrát pouze soubory, ve kterých jsme dělali změny nebo byly nově přidány. Samotné nahrání těchto souborů by trvalo 17 sekund. Vybrání pouze souborů, které se budou nahrávat na server trvalo *necelé 2 minut*. Většinou se však úpravy dělají v mnoha souborech a uživatel by mohl na některé z nich zapomenout, proto se nahrává celý projekt. Tento úkon by trval více než *6 minut*.

Zjednodušeně by v některých případech šla nahrát pouze šablona, která by byla nahrána cca za *2 minuty*. Ani jeden z těchto postupů nezaručuje, že někdo neprovedl změny v souborech na serveru a nahráním nové verze budou přemazány.

Využitím PHP Deploye nahrávají pouze změněné soubory, nahrávání trvalo *19 sekund*. Díky verzování nehrozí ztráta již vytvořené práce.

Můžeme tedy konstatovat, že nahrávání pomocí tohoto scriptu je rychlejší, bezpečnější a zamezuje možnosti ztráty dat.

Podobně jako u využití služby Bitbucket místo stahování dat z FTP, odhadujeme úsporu *více než 2 hodiny* měsíčně na každého vývojáře.

4.1.6 Vytvoření nové šablony webového projektu

S přechodem na VSC bylo potřeba vytvořit nový skeleton pro projekty. Do šablony bylo přidáno mnoho nových funkcí umožňujících vývoj na lokálním prostředí s možností nahrávání souborů na testovací nebo produkční prostředí. Webová šablona po provedených úpravách obsahuje příznaky, zda se jedná o testovací, lokální nebo produkční prostředí. Tyto příznaky rozlišují různá nastavení ovlivňující funkčnost šablony. Většina z úprav znamenala zvýšení efektivity a snížení pracnosti při každodenních činnostech. Šablona například automaticky rozeznává, jaké mají být výše zmíněné příznaky meta robots [30], čímž odpadá riziko neviditelnosti webů pro uživatele. Dále jsou přidány funkce automatické úpravy velikosti obrázků pomocí bezeztrátové komprese, minifikace CSS a JS souborů. Tyto úkony musel zaměstnanec dělat při každé změně ručně, a i malá úprava na projektu se prodlužovala o jednotky až desítky minut. Nyní se tyto úkony provádějí automaticky.

Vyhledávače, jako je Google, nezohledňují pouze obsahovou formu webů, ale také jejich kvalitu. Pomocí stránky Google PageSpeed Insight [31] je možné prověřit, jak společnost Google hodnotí webovou prezentaci po funkční stránce. Přidáním výše popsaných funkcí získávají webové prezentace skóre, bez potřeby ručních úprav.

Minifikace souborů

Mezi hodnocenými kritérii služby Google PageSpeed Insights [31] je také velikost souborů obsahující grafické styly a javascriptové funkce potřebné k výpočtům nebo animacím. Tyto soubory je možné tzv. minifikovat, což je proces odstraňující nepotřebné znaky a mezery z těchto souborů. Výsledný soubor obsahuje stejné funkce, jen je menší. Důvodem pro minifikaci je především nižší datová náročnost webů. Na stolních počítačích tato změna bude téměř neznatelná, u mobilních zařízení již může být poznat. Minimálně dochází k ušetření cenných mobilních dat.

Google vyžaduje také spojení souborů stejného typu do jediného souboru, což snižuje počet HTTP požadavků odesílaných na server. Tato funkce je v nové šabloně také implementována.

Bezeztrátová optimalizace obrázků

Ze stejných důvodů jako u minifikace souborů se provádí i optimalizace obrázků. Existuje mnoho služeb a algoritmů, které dokážou upravit obrázky tak, že lidské oko nepozná rozdíl, avšak změna ve velikosti souboru je obrovská, u některých souborů formátu *.jpg* až 64 %.

Dříve bylo nutné každý obrázek nahrávat do speciální služby, která ho upravila a následně stáhla. Tento proces byl časově velmi náročný především v případě velkých prezentací obsahujících velké množství grafických podkladů.

Do nové šablony byl implementován script, který automaticky upravuje všechny obrázky přidané do specifikované složky a ty upraví a uloží do cílové složky, kterou využívají ostatní pracovníci. Tento proces uchovává i originál obrazových materiálů, pro případ potřeby využití neupravených podkladů. V tabulce 1 vidíte rychlost úpravy obrázků ručně a pomocí implementovaného scriptu *imagemin*.

počet obrázků	Upravení obrázku ručně v s	Automaticky pomocí Gulp imagemin
1	23s	253ms
2	50s	430ms
10	4min 53s	12s
15	8min	16s
100	odhadem 33min	149s

Tabulka 1: Porovnání rychlosti úpravy obrázků

Nastavení meta robots

Webové stránky obsahují v hlavičce informaci říkající webovým vyhledávačům, zda mají být stránky zobrazovány ve výsledcích. Pro tyto účely je nastavován meta tag *robots*. U tohoto příznaku lze nastavit několik možností, určujících, jak se vyhledávač bude chovat ke stránkám.

Do šablony byla implementována funkce, která rozpoznává, zda se jedná o testovací, produkční nebo lokální prostředí. Podle vývojového prostředí se nastaví příslušný meta tag na hodnotu odpovídající požadovanému stavu.

Hlavní důvod implementace této funkce byla velká míra chybovosti. Špatným nastavením výše popsaného meta tagu dochází k nepřístupnosti webu pomocí vyhledávačů. Dá se říct, že stránky s nesprávným nastavením nejsou vidět ve výsledcích vyhledávání a klienti mohou přicházet o zákazníky.

Problém nejčastěji nastával při migraci nových funkcí z testovacího prostředí na produkci, kam se přeneslo i nastavení.

```
<META NAME="ROBOTS" CONTENT="NOINDEX, FOLLOW">
<META NAME="ROBOTS" CONTENT="INDEX, NOFOLLOW">
<META NAME="ROBOTS" CONTENT="NOINDEX, NOFOLLOW">
```

Tabulka 2: Nastavení meta robots [30]

4.2 Výběr vhodných funkcí pro interní systém

Pro každý systém je důležité nadefinovat jeho základní funkce. V této kapitole se zaměřím na funkce, které společnost od interní aplikace očekává a jsou pro ni zásadní. Jednotlivé funkce byly vybrány pomocí afinitních diagramů [14], kdy každý ze zaměstnanců přidával do na tabuli všechny své nápady na funkce a požadavky budoucího systému. Po shromáždění všech informací jsme roztřídili požadavky na systém a proběhlo několik kol revizí, kdy jsme odebírali funkce, které pro nás neměli dostatečnou váhu. Níže jsou popsány funkce, na kterých jsem se ve společnosti shodli, že je systém musí umět. Společnost však počítá s následným rozšiřováním systému.

4.2.1 Registrace do systému

Systém bude obsahovat interní data, kontakty a přístupy do administračních částí webových stránek klientů. Přístup do interního systému je nutno zabezpečit. Idea je, aby do systému měli přístup pouze registrovaní uživatelé.

4.2.2 Registrace uživatelů

Registrace uživatelů bude probíhat na straně společnosti Litea Solution s.r.o.

Společnost nepředpokládá velké množství nově registrovaných uživatelů – odhadované jsou jednotky účtů měsíčně. Každého uživatele proto bude vytvářet administrátor. Uživatelský účet bude spjat s emailovou adresou klienta a vygenerovaným bezpečným heslem. Administrátor zároveň vytvoří role uživatele a přiřadí ho k projektům, do kterých má právo nahlížet, případně je upravovat.

Tyto údaje budou následně zaslány uživateli.

Jiné řešení například pomocí účtů Google nebo vlastního řešení se společnost nebrání, vše záleží na implementaci existujícího řešení.

4.2.3 Přihlášení do systému

Pokud chce uživatel využívat systém, je potřeba se přihlásit. Přihlášení se provádí na speciální webové stránce, na které je pouze přihlašovací formulář a možnost obnovit zapomenuté heslo.

Pro úspěšné přihlášení je uživatel povinen zadat přihlašovací jméno ve tvaru emailové adresy a heslo.

Po přihlášení je uživatel přesměrován na úvodní stránku systému.

4.2.4 Odhlášení ze systému

Přihlášený uživatel má možnost se odhlásit ze systému pomocí tlačítka „odhlásit se“.

Odhlášený uživatel nemá přístup k žádným datům aplikace.

Z bezpečnostních důvodů je v systému nastaveno automatické odhlášení.

4.2.5 Správa rolí

Systém bude obsahovat několik rolí. Každá role bude mít jiná oprávnění. Role bude možné slučovat, kombinovat, a tedy rozšiřovat pravomoci jednotlivých uživatelů podle potřeb pracovníka.

Host

Systém má implementovanou základní roli nazvanou *host*. Tato role má nastavena pouze základní oprávnění pro pohyb v aplikaci, avšak nevidí žádná data.

Externista

Externista, nebo také externí zaměstnanec, bude mít nastavená práva pouze do určitých projektů a neuvidí všechny informace, například kontakty. Účelem této role je možnost přidávat úkoly externím zaměstnancům do interního systému. Zajistit jim přístup k dostatečným informacím o projektech, tedy souborům, přístupovým údajům a správě časů. Důležité je neposkytnout jim příliš volný pohyb po aplikaci.

Externista bude moct přidávat a upravovat úkoly, avšak nebude mít přístup k vytváření nových složek projektů. V záložce s projektem uvidí informace o projektu, soubory k němu přiřazené, ale neuvidí časovou náročnost a kontakty. Nebude také moct prohlížet časy strávené na projektu.

Klient

Role *klienta* je velmi podobná jako role *externisty*. Jedná se o uživatele se zpřístupněnými projekty, do kterých může nahlížet. Nevidí však interní informace o projektu.

Jeho přístupy budou omezeny do správy úkolů, kde může vytvářet nové úkoly a upravovat již existující úkoly.

Pracovník

Většina interních zaměstnanců bude mít přiřazenou roli *pracovník*, což je role s vyváženými pravomocemi potřebnými ke každodennímu plnění úkolů. Na rozdíl od role *externista* má *pracovník* přístup ke všem projektům, může vytvářet nové stránky a přidávat nové přístupy, nemůže je však mazat. Bude také mít možnost vytvářet připomínky, které lze přiřazovat ostatním uživatelům. Tato role také vidí stav projektů, kolik času je již na projektu stráveno, do kdy má být projekt hotov atd.

Pracovník bude moct zobrazit všechny své odpracované časy s možností filtrování podle projektu nebo stráveného času.

Projektový manager

Projektový manager bude druhá nejvyšší role v systému. Umožňuje například přidávat externí zaměstnance k projektům nebo k nim přidávat vícepráce.

Ze stránky projektu se dostane i na výpis zobrazující všechny práce provedené na tomto projektu s popisem, štítky a s časovou náročností.

Uživatel s touto rolí také vidí všechny odpracované hodiny ostatních uživatelů, což mu umožňuje kontrolovat efektivitu práce zaměstnanců. Jedná se o stejný výpis, který si může zobrazit role *pracovník*, jen zobrazuje všechny zaměstnance.

Administrátor

Administrátor je role s nejvyšším uživatelským oprávněním. *Administrátor* jako jediný může měnit oprávnění ostatních uživatelů. Tato role je přiřazena pouze vedení společnosti. Aplikace v sobě bude mít funkce spojené s tokem peněz, které uvidí pouze tato role. Jednoduše řečeno *administrátor* je role, která může v systému využívat všech implementovaných funkcí.

4.2.6 Inteligentní stopky

Jednou z hlavních funkcionalit systému by mělo být měření času stráveného na jednotlivých úkolech. Díky měření časů odpracovaných na projektech získá vedení společnosti přesný přehled o oddělaných hodinách na dílčích úkonech. Informace o časech získané tímto způsobem budou sloužit především k fakturaci klientům a interním statistikám, budou poskytovat informace o aktuálně stráveném čase na projektech a o výkonnosti zaměstnanců a jako podklady k fakturaci externistů.

Stopky budou fungovat následovně:

1. *Zaměstnanec A* vybere projekt, na kterém bude pracovat.
2. Do pole popis vypíše, co v daném úkolu udělal, případně doplní poznámku o vzniklém problému.
3. Vybere, jakého typu byl plněný úkol. K tomuto účelu bude implementován výběr ze *štítků* tzv. *„tagů“*. Pracovník bude mít možnost vybrat z již předdefinovaných *štítků* (grafické práce, design, programování, komunikace atd.). Systém bude umožňovat i přidání nového *štítku* vystihujícího řešenou problematiku.
4. Pro začátek práce stiskne tlačítko *„začít práce“*.
5. Po dokončení úkolu nebo jeho části lze stopky ukončit tlačítkem *„konec práce“*.
6. Všechna vyplněná pole lze v průběhu práce i po jejím skončení měnit.
7. Stopky budou mít i možnost začít práce v jiný čas, než je aktuální. Tato funkce se může zdát riskantní vůči zákazníkům. Nicméně zaměstnanci budou měřit i čas

strávený na komunikaci s klientem. Může tedy nastat situace, kdy zaměstnanec pracuje na *projektu A* a volá mu klient ohledně *projektu B*, kde potřebuje technickou podporu. Zaměstnanec si následně může upravit konec práce na *projektu A* a dopsat čas k *projektu B*.

4.2.7 Správa projektů

Systém bude umožňovat vytvoření nového projektu s danými atributy.

Systém podporuje přidávání webových stránek ke každému projektu – tyto stránky budou rozlišeny, zda se jedná o produkční prostředí nebo testovací.

U projektů je možné nastavit externí zaměstnance. Tuto funkci může využít každý s rolí *projektový manager* a vyšší. Přidáním externího zaměstnance získá uživatel s rolí *externista* přístup k informacím o daném projektu. Zároveň se uživateli *externista* zpřístupní možnost připisovat si časy k danému projektu. Každému projektu je možno přiřadit libovolný počet externích zaměstnanců.

Podporované atributy:

Společnost

Každému projektu je nutno přiřadit společnost, pro kterou se projekt realizuje.

Klient

Styčná osoba zodpovídající za projekt na straně zákazníka.

Název projektu

Pokud společnost dělá více projektů pro jednoho zákazníka, může být problém jednotlivé projekty rozlišit. Každému projektu je tedy nutno přiřadit krátký název.

Popis

Popis projektu bude obsahovat základní informace o specifikaci. Důležité milníky, doporučené technologie nebo odkazy na soubory spojené s realizací.

Odhadovaná pracnost

Číselná hodnota zobrazující klientem schválenou časovou dotaci.

Hodinová sazba

Číselná hodnota určující, kolik je hodinová sazba pro daný projekt. V budoucnu bude rozšířeno na hodinovou sazbu jednotlivých rolí pracovníků. Hodinová sazba je viditelná pouze pro uživatelské role.

Celková částka

Některé projekty nejsou odvozovány od odpracovaných hodin, jedná se o nasmlouvanou jednorázovou cenu. Systém bude mít možnost přidat i tuto informaci. Stejně jako hodinová sazba bude viditelná pouze zaměstnancům s dostatečným oprávněním.

Provize

Částka určující, kolik je odhadovaný výdělek po odečtení nákladů.

Zdroj projektu

Informace, odkud společnost daný projekt získala. Jedná se o informaci, která bude sloužit k interním statistikám.

Bude se jednat o výběr z předem definovaných hodnot:

- Stávající klient
- Z ulice
- Doporučen zákazníkem
- Partneři

Typ produktu

Informace, o jaký typ produktu se jedná, taktéž informace sloužící pro interní statistiky. Bude se jednat o výběr z předem definovaných hodnot:

- Aplikace
- Webová prezentace
- Eshop
- Grafické práce
- Marketing

Status projektu

Stavová informace o projektu.

- Potenciál
- Schůzka
- Připravit nabídku

- Nabídka podána
- Pracuje se
- Fakturováno
- Zrušeno
- Hotovo

Pravděpodobnost úspěchu

Interní hodnocení rizika projektu.

Datum odevzdání

Plánované datum odevzdání, možné vybrat datum, pomocí funkce datepicker.

Datum splatnosti

Při fázi fakturace bude možno zadat datum splatnosti. Tato informace bude udržovat informace na jednom místě v interním systému a vidět je by bylo umožněno pouze pověřeným osobám.

Přidání víceprací

Projekt již má informaci ohledně odhadované pracnosti. Občas se však stává, že zákazník chce přidat další funkce. Pokud na tyto funkce nestačí rozpočet, považují se za placené vícepráce. Vícepráce by bylo možné přidávat jako samostatné projekty, ale to by zvýšilo počet projektů a znesnadnilo orientaci v systému. V projektu bude možnost přidat vícepráce. Bude se jednat o krátký popis práce a hodinový rozpočet.

4.2.8 Správa klientů

Společnost si vytvoří databázi klientů, se kterými spolupracuje. Cílem tohoto modulu je uchovávat kontaktní informace o klientech na jednom místě. Tento modul bude propojen s ostatními, například správou projektu, kde se bude vybírat odpovědná nebo kontaktní osoba. Díky tomuto bloku si může každý zaměstnanec v interním systému najít kontakt na styčnou osobu, se kterou se potřebuje domluvit na detailech ohledně daného úkolu a nemusí vyrušovat ostatní kolegy, aby mu kontakt předali.

Tento blok bude přístupný pouze uživatelům s dostatečným oprávněním. Nenastane tedy situace, kdy externí zaměstnanec bude mít přístup ke všem kontaktům, které má společnost uložené.

Správa klientů bude obsahovat následující atributy:

- Jméno a příjmení
- Email

- Telefon
- Společnost (půjde o výběr z databáze vytvořených společností s možností filtrace)
- Poznámka

4.2.9 Správa hostingových služeb

S rozrůstajícím se počtem klientů se společnost musí potýkat s rostoucím počtem webových hostingů obsahujících klientské stránky. Oproti původní aplikaci, kterou společnost využívala, se rozhodla pro využití jedné aplikace jako centrálního systému pro správu hostingových služeb. Bude se jednat o administrační část s možností přidávat nové hostingové služby a upravovat stávající.

Pro správu hostingových služeb se počítá s následujícími atributy:

Název hostingu

Krátké označení specifikující, o jaký hosting se jedná – například společnost, u které je provozován.

URL adresa pro přihlášení do administrace

V tomto modulu bude možnost uložit přístupy potřebné pro přihlášení do administrace hostingové služby. Bude se jednat o URL adresu, vedoucí k přihlašovacímu formuláři dané hostingové společnosti.

Přihlašovací údaje do hostingu

Bude se jednat o vstupy přihlašovací jméno a heslo, sloužící k přihlášení do administrace hostingové služby.

Cena za rok

Pokud klient využívá hosting společnosti Litea, je důležité uchovat si informaci, kolik klient bude platit za využití této služby. Tuto informaci by měl vidět pouze uživatel s dostatečně vysokým oprávněním.

Odkaz na přihlášení do databáze

Většina hostingových služeb využívá vlastního rozhraní pro připojení k databázi daného webu. Toto pole bude obsahovat URL adresu, pro přístup k této stránce. Přihlašovací údaje jsou následně spjaty s jednotlivými aplikacemi/stránkami běžícími na daném hostingu.

Logo hostingu

Pro přehlednost bude implementována i možnost vložit logo hostingové společnosti. Pokud bude ve výpisu více společností, podle loga se uživatelé orientují často rychleji než podle názvu.

4.2.10 Správa webových stránek

Podobně jako u hostingových služeb si společnost hodlá udržovat informace o jednotlivých stránkách, které spravuje nebo vytvořila. Uživatelé s dostatečným oprávněním budou mít přístup k informacím ohledně přístupových údajů na FTP a do administrační části webu. Všechny tyto informace budou pro přehlednost na jednom místě.

Výběr hostingu

Ke každé stránce bude možné zvolit hostingový server, kde je produkt uložen. Bude se jednat o výběr z již vytvořených hostingů a půjde v něm filtrovat.

Výběr společnosti

Bude se jednat se o výběr společnosti, pro kterou jsou stránky vyvíjeny. Půjde o selectbox obsahující všechny do databáze zařazené společnosti.

URL adresa stránky

Adresa, na které jsou stránky umístěny.

Typ vývojového prostředí

Informace, zda se jedná o testovací nebo produkční prostředí. Tato informace se bude využívat pro přehlednost výpisů stránek. Plánovaná je také možnost filtrovat pouze stránky s odpovídajícím vývojovým prostředím. Hodnota tohoto pole bude využívána i pro níže popsaný script *checker*, který kontroluje nastavení právě podle této hodnoty.

Datum spuštění webu

Interní informace, kdy byl projekt spuštěn.

Datum vypršení webu

Některé weby jsou plánované pouze na krátké období. Pro tyto příležitosti bude implementována možnost přidat datum ukončení stránek. Plánovaná je i funkce upozornění pomocí emailu, která by byla opakovaně odesílána určeným uživatelským rolím v dostatečném předstihu.

Cena za měsíc

Cena za poskytování hostingových služeb, pokud se platí společnosti.

Poznámka

Textový blok, do kterého bude možné přidávat informace o projektu, které se nehodí do jiných předdefinovaných polí. Očekávány jsou specifikace a odkazy na externí soubory.

Checker

Společnost využívá vlastních scriptů kontrolujících určité funkčnosti každého webu. Tento script společnost nazývá Checker a bude implementován do modulu „správa webových stránek“.

Všechny z funkcí tohoto scriptu se spouštějí automaticky minimálně jednou denně a budou kontrolovat veškeré stránky přidané do modulu „správa webových stránek“.

Mezi jeho hlavní funkce patří:

Response

Zjišťuje, zda stránky fungují.

Meta Robots

Script na stránce vyhledá informace obsažené v meta tagu robots. Jedná se o HTML meta tag, určující, zda vyhledávače indexují konkrétní stránku, nebo ji ignorují. Tato informace je důležitá jak pro vývoj aplikace, kdy společnost nechce, aby unikla informace o vývoji daných stránek na testovacím prostředí, tak i pro produkční prostředí, kde je naopak vyžadováno, aby vyhledávače stránky indexovaly a následně se zobrazovaly ve výsledcích předních vyhledávačů jako je Google.cz, Seznam.cz, atd.

Rychlost odpovědi webu

Tento script zjišťuje i rychlost odpovědi webové stránky. Tato informace udává, zda stránky fungují v pořádku, případně zda nenastal problém na serveru.

4.3 Správa úkolů

Nejpoužívanější částí systému by měla být správa úkolů. Cílem je udržet všechny úkoly na jednom místě a využívat pouze jednu aplikaci pro jejich správu.

4.3.1 Složky

Systém bude umožňovat vytvářet složky pro každý projekt. Složkou se rozumí virtuální místo, do kterého lze přiřazovat úkoly.

Složky bude moct vytvářet kterýkoli zaměstnanec.

Každý zaměstnanec bude mít minimálně jednu složku, základní složka bude nazvána *inbox* a nebude se vztahovat k žádnému projektu. Tato složka nepůjde smazat ani přesunout.

Složkám půjdou nastavit tyto atributy:

Název

Jednomu projektu je možné vytvořit více složek, proto je potřeba tyto složky rozlišovat. Může nastat například situace, kdy dojde k rozdělení složek pro zaměstnance dle jejich rolí. *Projekt X* může být rozdělen na *Projekt X – Grafika*, *Projekt X – Programování*, protože úkoly rozdílné role nezajímají.

Projekt

Ke každé složce bude nutno definovat, k jakému projektu náleží. Úkoly vložené do složky budou brát tento projekt jako výchozí při přidávání časů do systému. Projekty jsou vytvořeny předem a u složek se pouze vybírá ze seznamu existujících projektů.

Ikona

Pro přehlednost bude možné složkám přiřadit ikonu, nejčastěji půjde o logo společnosti nebo projektu.

4.3.2 Úkol

Systém bude umožňovat vytvoření nových úkolů k danému projektu. Úkolem se rozumí jedna entita problému určená k vyřešení. Úkol bude mít základní stav *nedokončený*. Po vytvoření se zobrazí v seznamu úkolů v dané složce. Pro přidání více podrobností bude nutné znovu vybrat vytvořený úkol a díky tomu se zobrazí podrobné vlastnosti.

Pro dokončení úkolu bude u každého úkolu grafický prvek checkbox, sloužící k odškrtnutí úkolu. Dokončením úkolu se rozumí převedení do stavu *dokončený*. V tomto stavu se úkol nebude zobrazovat v základním zobrazení seznamu úkolů. *Dokončené* úkoly bude možné zobrazit pomocí tlačítka na stejné obrazovce, které pod nedokončenými úkoly zobrazí graficky oddělené *dokončené* úkoly. Tato funkčnost bude vycházet z aplikace popsané v kapitole 5.1.4.

Každému z úkolů bude možné nastavit tyto atributy:

Název

Název úkolu bude text zobrazený na seznamu úkolů. Měl by krátce vystihovat, o jakou problematiku jde. Dobře zadaný název úkolu může být například „Grafické práce: vytvoření podstránky kontakt“. První část úkolu rozlišuje, o jaký typ práce se jedná a druhá krátce popisuje úkol samotný.

Popis

Aby v názvu úkolu nebylo moc informací, což by bylo nepřehledné, každý úkol bude mít atribut popis. Bude se jednat o textové pole s neomezenou délkou. Zadavatel úkolu zde může rozepsat úkol podrobněji, přidávat URL odkazy na stránku, kde se problém vyskytuje nebo vkládat nahrané externí soubory.

Přiřazená osoba

V základním stavu úkolu *nedokončený* není přiřazena žádná osoba, zodpovědná za splnění úkolu. Kterýkoli uživatel s dostatečným oprávněním může přiřadit zodpovědnou osobu.

Priorita

Nepovinný atribut označující, jak důležitý tento úkol je. Tato informace bude sloužit především pro zaměstnance, kteří mají přiřazeno několik úkolů ve stejný okamžik. Díky prioritě budou zaměstnanci vědět, na jakém úkolu začít pracovat a které úkoly zpracovávat následně. V ideálním případě budou úkoly označeny grafickým elementem podle priority, například každá priorita bude mít jinou barvu, nebo úkoly s vyšší prioritou budou vždy v horní části seznamu úkolů.

Datum dokončení

Systém bude umožňovat přidat k úkolu datum jeho potřebného dokončení. Tato informace se bude zadávat pomocí javascriptového pole vyvolávajícího výběr data. Po kliknutí na toto pole se zobrazí možnost vložit čas a datum dokončení.

Čeká se na klienta

U každého úkolu bude pole indikující, zda úkol čeká na vyřízení na straně společnosti nebo klienta. Toto pole je pro společnost důležité kvůli smluvním reakčním dobám. Někteří z klientů mají ve smlouvě dohodnutou reakční dobu od 2 do 48 hodin. Jedná se o dobu, do které musí společnost oznámit, jakým způsobem se bude daný úkol řešit, zda má všechny potřebné podklady atd. Nejedná se o čas, do kdy má být úkol vyřešen.

Cílem tohoto atributu je uchovávat informace o reakčních časech u každého úkolu a díky tomu mít v případě nespokojenosti klienta důkaz, po jakých časových intervalech společnost reaguje. V následujícím příkladu se pokusím znázornit očekávanou funkčnost.

Příklad:

Klient má smluvně dohodnutou reakční dobu 4 hodin v předem stanovených pracovních hodinách.

1. *Klient* zadá úkol do systému v pondělí 14:00.
2. *Zaměstnanec* se tomuto úkolu začne věnovat v 17:30 a dává informaci pomocí komentáře o chybějících podkladech potřebných pro dokončení úkolu. Zároveň přehodí přepínač na stranu *klienta*. Systém si uloží čas reakce 17:30, tedy 3 hodiny a 30 minut.
3. *Klient* dodá potřebné informace v úterý dopoledne v 11:00 a přepne úkol zpět na *zaměstnance*. Systém si uloží čas reakce klienta 11:00, tedy 17 hodin 30 minut.
4. *Zaměstnanec* v 11:30 již má vše potřebné k jeho dokončení a může úkol realizovat. Systém si uloží čas reakce 30 minut.

V případě potřeby je systém schopný zobrazit časy reakce a ty se mohou prezentovat zákazníkovi.

Štítky

Systém bude obsahovat již zmíněné štítky. Ke každému úkolu bude možné přiřadit štítky nesoucí informaci, o jaký typ úkolu se jedná. Štítky nebo také tagy budou sloužit pro přehlednost a následnou kontrolu, kolik času bylo stráveno na kterých aktivitách pro daný projekt. Díky tomu se vedení společnosti může lépe rozhodovat při určování časové náročnosti nových projektů.

V systému budou přednastavené štítky, ale bude zde i možnost přidávat nové dle potřeby.

Příklady štítků:

- Grafické práce
- Projektové řízení
- Programování
- Kódování
- Oprava
- Vícepráce

4.3.3 VSC – Verzovací systém

Se změnou způsobu vývoje aplikací pomocí VCS služby Bitbucket by bylo dobré mít v interním systému informace o změnách na jednotlivých projektech. Jednalo by se o speciální záložku v interním systému spjatou s daným projektem. Na této záložce by se zobrazovaly změny provedené pomocí verzovací aplikace GIT. Tato funkčnost nemá vysokou prioritu, jednalo by se pouze funkci informačního charakteru. Výhodou implementace tohoto malého rozšíření by byla centralizace všech informací o daném projektu na jedno místo. Díky tomu by nemuseli zaměstnanci hledat na službě třetí strany, kdy a jaké byly naposledy provedeny změny. Možnost implementace této služby je možná využitím API, kterou služba Bitbucket poskytuje.

4.3.4 Přehled časů

Funkce přehled časů má za účel přehledně zobrazovat aktuální stav projektů. V tomto výpisu se bude zobrazovat termín odhadovaného dokončení zvýrazněný v případě překročení data odevzdání. Dále zde bude zobrazen rozpočet ukazující domluvenou časovou náročnost, hodiny zobrazující již odpracovaný čas a status bar graficky znázorňující hodinový stav. Pokud bude status bar zelený, na projektu ještě zbývá čas. Při červeném zabarvení byla překročena odhadovaná pracnost a je potřeba se zaměřit na rychlé dokončení úkonů. Barevnost status baru má za cíl upozornit management na blížící se konec projektu. To má pomáhat při komunikaci se zákazníkem v případě potřebných víceprací. Zadáním dalších víceprací do systému ve správě projektu se navýší celkový čas.

4.3.5 Budoucí plán

Budoucí plán je část programu určená především pro vedení společnosti a projektový management. Prozatím se by se mělo jednat o jednoduchý grafický výpis ukazující, kolik pracovní kapacity je odhadováno za využitou v následujících měsících. Výpis bude zobrazovat průměrný počet hodin, které lze odpracovat v budoucích obdobích.

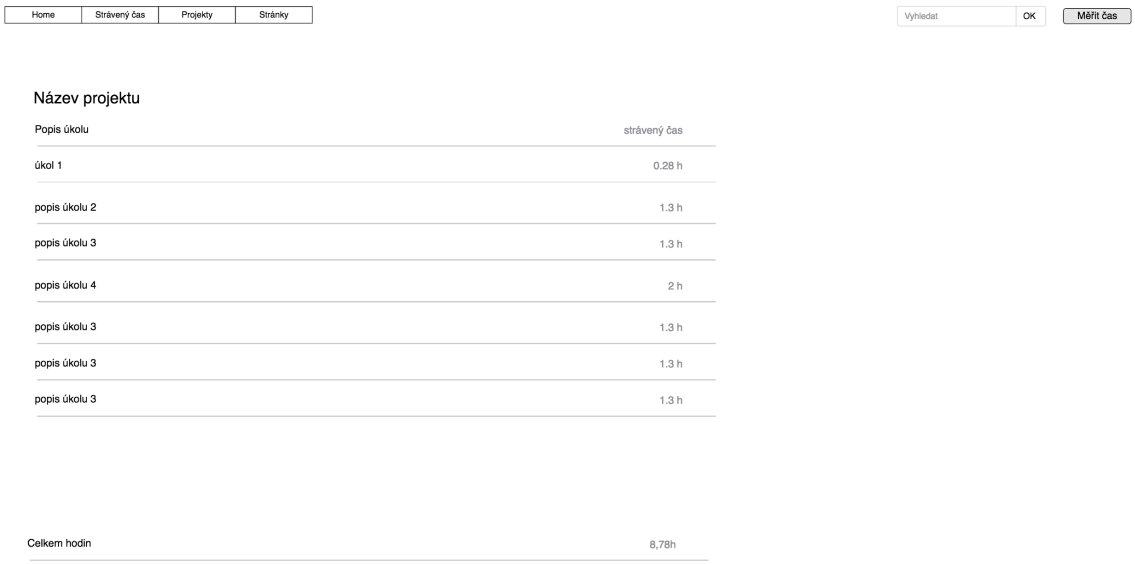
V této části aplikace je velký prostor pro budoucí rozšíření, například zobrazování dílčích projektů pomocí ganttových diagramů, možnost plánovat každý projekt po částech – programování, grafické práce, projektové řízení, nebo odhadovanou pracnost přiřazovat zaměstnaneckým rolím. Díky výše zmíněným funkcím by se dosáhlo přesnějšího zobrazení obsazených kapacit. Obáváme se však, že pracnost těchto rozšíření bude přesahovat rozsah této diplomové práce.

4.3.6 Výpisy časů

V aplikaci bude několik typů výpisů časů. Půjde především o informace o projektech, ale také zobrazení odpracovaných hodin zaměstnanců.

Výpis časů strávených na projektu

Jeden z výpisů se bude týkat projektu. Každý projekt bude mít svůj výpis všech odpracovaných hodin. Mezi informacemi bude popis úkolu, kdo úkol vypracoval a strávený čas. Pod výpisem by měl být zobrazen celkový strávený čas a informace, kolik času zbývá v rozpočtu. Ve výpisu bude možné filtrovat dle uživatelů a štítků.



Obrázek 15: Návrh výpisu prací projektu

Měsíční přehled všech uživatelů

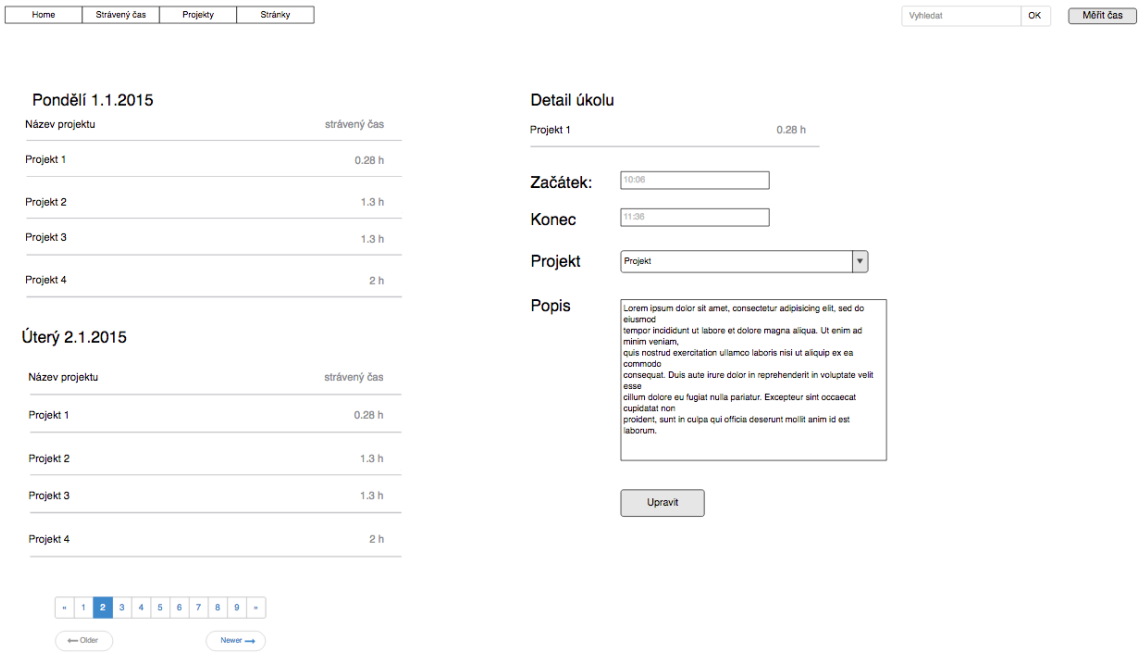
Pomocí položky menu bude mít uživatel přístup ke statistikám odpracovaných hodin za jednotlivé měsíce. Ve výpisu bude vždy vypsán název projektu, na kterém zaměstnanec pracoval a celkový počet hodin odpracovaných za vybraný měsíc.

Vypsané projekty budou sloužit jako odkaz na stránku s výpisem všech odpracovaných úkolů u zvoleného projektu.

Podle uživatelských rolí bude tento výpis zobrazovat různé informace navíc. Uživatelé s rolí *projektový manager* a vyšší uvidí v tomto výpisu ne jednoho uživatele, ale všechny. Tento výpis půjde řadit nejen dle projektů, ale také podle zaměstnanců.

Denní přehled odpracovaných hodin zaměstnance

Aplikace bude obsahovat přehled odpracovaných hodin aktuálně přihlášeného uživatele za poslední den. Tento výpis půjde rozšířit na zobrazení celého týdne. Mezi jednotlivými týdny bude možnost se pohybovat a prohlížet veškerou historii odpracovaných hodin. Pokud bude mít uživatel dostatečné oprávnění, může i upravovat zadané informace.



Obrázek 16: Návrh zobrazení denního výpisu odpracovaných hodin

5 Vlastní řešení nebo využití již existujícího

Na internetu je možno najít nespočet aplikací plnících podobné nebo dokonce stejné funkce, jako námi hledaný systém. V této kapitole se zaměřím, na již vytvořené systémy, které je možné využít. Otestované budou jak aplikace volně šiřitelné pod licencí open source, tak placená řešení. Z nasbíraných informací odvodím, zda se společnosti vyplatí použít již hotové řešení, ať již by se jednalo o koupi softwarové licence, úpravu volně šiřitelného řešení, nebo placení měsíčních poplatků některé z existujících aplikací.

Průzkum trhu zároveň rozšíří obzory o možné funkce, které by bylo dobré implementovat do případné vlastní aplikace, pokud by se společnost rozhodla pro tvorbu vlastní aplikace. U jednotlivých aplikací je hodnocena jak celková funkčnost, tak i možnost integrace služeb třetích stran a její rozšiřitelnost pomocí vlastních sil.

Otázkou pak je, zda se vyplatí systém implementovat vlastní nebo využít již existujících řešení. Nejdůležitějším faktorem bude, do jaké míry již existující řešení splňují specifické požadavky zadavatele. Následně je důležité určit možnosti upravitelnosti stávajících řešení do podoby, kterou si zadavatel představuje tak, aby nebyla výsledná cena mnohonásobně vyšší než vytvoření celé aplikace na zelené louce.

Při využití již existujícího řešení se nevyhneme úpravám, protože požadavky zadavatele jsou velmi specifické. Většina existujících systémů je stavěná právě na jeden účel, a to buď správa úkolů nebo času. Naše řešení má být spojením obou funkcí a zároveň obsahovat mnoho dalších funkcionalit, jako je správu klientů, webových stránek a hostingů. V ideálním případě tedy hledáme aplikaci obsahující co možná největší množinu požadovaných funkcí pokud možno s minimálními úpravami.

Dalším problémem jsou úpravy na již vytvořeném řešení. Programátor, který by se zabýval úpravami, musí nastudovat cizí kód aplikace. Pokud je daný kód kvalitně zdokumentován, nemusela by být úprava složitá, avšak v opačném případě se nemusí jednat o nejlevnější řešení. Musíme také počítat s preferovanou volbou využití vlastního programátora, který by mohl na aplikaci pracovat, pokud nemá zadaný jiný projekt. Firemní programátor neovládá všechny typy jazyků, tedy volba úpravy řešení bude záležet i na typu programovacího jazyka, ve kterém je systém implementován.

Dlouhodobě provozované systémy mají na druhou stranu výhodu v odladění možných chyb. Přece jen po několika letech vývoje a testování na reálných uživatelích je velká šance, že drobné problémy budou odstraněny. U vlastní aplikace je potřeba počítat nejen s vývojem, ale také s opravou odhalených chyb v průběhu užívání. Využitím softwaru, který je dále vyvíjen ostatními vývojáři, nezískáme pouze již vytvořený základ, ale také podporu od ostatních programátorů, možné opravy chyb z jejich strany, přidávání nových funkcí a zpětné vazby od ostatních uživatelů.

Rozhodne-li se zadavatel pro tvorbu zcela vlastního řešení, přinese to společnosti kompletní přehled nad tvorbou aplikace. Všechny funkce budou navrženy přesně na míru dle zadání. Systém nebude obsahovat funkce, které nejsou potřebné. Při dodržení programovacích

konvencí by nemělo být následné rozšiřování aplikace problém. Budeme moct určit, jaké technologie budou na vývoj použité a tvorba dokumentace bude zcela na nás. Co se týče oprav chyb, nebudeme muset čekat na cizí programátory, než chybu opraví. Jednou z dalších výhod by bylo, že společnost bude vlastníkem aplikace. Může s ní tedy nakládat dle svých představ. To neplatí při úpravě již existujících řešení. Většina projektů, i těch volně šiřitelných pod licencí open source, požaduje uvolnění upravené verze dále na internet pro ostatní uživatele.

Při volbě vytvoření vlastního řešení se nejedná pouze o implementaci. Bylo by potřeba udělat analýzu, následný návrh a následnou implementaci. Tento proces se může časově protáhnout a být mnohem dražší než pouhá úprava existujícího řešení.

Každý z přístupů přináší svá pro a proti, jejich výčet obsahuje tabulka na této stránce.

Vlastní řešení	**Existující řešení**
Výhody	
Aplikace přesně na míru dle požadavků společnosti	Podpora
Využití interních zaměstnanců, jejich učení	Více funkcí, než je potřeba
Cena	Následné updaty
Využití vlastního severu	Spolupráce s více systémy
Bude se jednat o referenci	
Možnost rozšiřování	
Neměnná cena s počtem uživatelů	
Nevýhody	
Časová náročnost návrhu a implementace	Cena
Možnost chyb	Neupravitelné nebo těžce upravitelné řešení
Nutnost aktualizovat pomocí vlastních sil	Nutnost doupravit aplikaci
	Chybějící funkce
	Drahé doprogramování funkcí, pokud je možné

Tabulka 3: Zhodnocení vlastního řešení nebo využití existujícího

5.1 Analýza existujících aplikací

V následujících podkapitolách se zaměřím na analýzu existujících řešení. Výběr byl zvolen společností, dle priorit a zkušeností s danými aplikacemi. Některé z nich slouží pro inspiraci pro případnou implementaci systému. Pro stanovení vhodnosti aplikace budeme vycházet především z výsledků kapitoly 4.2.

5.1.1 Trac

Trac je softwarové řešení zabývající se především managementem softwarových projektů. Jeho hlavní část se zabývá issue tracking systémem s implementovanou Wiki, ta se může vždy hodit a bereme ji jako výhodu. Vzhledem k tomu, že se jedná o systém zaměřený přímo na softwarové projekty, obsahuje velké množství užitečných funkcí. Ne všechny jsou však pro potřeby společnosti důležité, pro lepší přehlednost v aplikaci by bylo dobré je vypnout nebo odstranit.

Přidávání nových úkolů a jejich editace je relativně uživatelsky příjemné, avšak je znát, že se jedná o aplikaci pro správu chyb, a ne obyčejných úkolů. Vzhled aplikace je strohý a na dnešní poměry vypadá dost staře. To však z hlediska funkčního nemá velký vliv. Zobrazení

úkolů probíhá pomocí tabulky, která umožňuje různé typy filtrování. Nenalezl jsem možnost přidávat k jednotlivým úkolům úkoly podřazené, což je jedna z požadovaných funkčností. Systém v základu nemá měření času jako takového, bylo by nutné přidat nová pole pro vkládání těchto informací nebo lépe funkci stopek. Každý z úkolů lze přiřadit jednomu řešiteli, což odpovídá našim požadavkům. Úkolům lze také přidávat různé typy kategorií, v administraci jsem možnost kategorie upravovat nenašel, nemělo by se jednat o velkou úpravu. V základu jsou kategorie defekt, rozšíření, úkol. V aplikaci je možné nastavovat uživatelská práva každému uživateli, s trochou práce by se dala upravit přímo podle našich představ. Výhodou, která se nám líbí, je integrace verzovacích systémů. I když nemá vysokou prioritu, je dobré vědět, že je v systému vytvořena. Trac umožňuje tvořit milníky a jednotlivé verze software. Neumožňuje však odhad pracnosti v hodinách. Přidání těchto statistik by znamenalo velkou úpravu systému. Pro správu kontaktů a stránek by šla využít Wiki, ale hesla by nebyla zabezpečena, což není rozumné řešení. Celý systém je možno nainstalovat na vlastní server s možností instalace různých balíčků a jazykových mutací.

Jednou z výhod aplikace Trac je dobrá dokumentace a velké množství uživatelů. Aplikace se vyvíjí od roku 2003 a nové aktualizace jsou pravidelné. Na webu je možné si projít jednotlivé změny. Trac je napsán ve skriptovacím jazyce Python. Je to jeden z jazyků, se kterým nikdo z našich programátorů nemá zkušenosti a vývoj by se díky neznalosti jazyka Python prodražil. Celkově systém obsahuje zajímavé funkce, ale velké množství požadovaných funkcionalit by bylo potřeba dodělat.

5.1.2 Bugzilla

Bugzilla [32] je webová aplikace pro sledování chyb, tzv. Bug tracking. Jedná se o software vytvořený organizací Mozilla, stojící za známým internetovým prohlížečem Mozilla Firefox. Celý systém je napsán ve skriptovacím jazyce Perl. Základní jádro systému bylo navrhnuto profesionály ze společnosti Mozilla, následně projekt rozšiřují dobrovolníci z celého světa.

Pro běh aplikace je potřeba nainstalovat webový server Apache a databázi běžící na databázovém serveru MySQL. I když se jedná o aplikaci pro řešení chyb, samotný systém jich obsahuje nemalé množství. Opravy a nové funkce přibývají pravidelně a je tedy doporučeno sledovat její vývoj. [33]

Vzhled systému připomíná systém z roku 1998, kdy začal vývoj. Nejedná se tedy o aplikaci, kterou chcete prezentovat klientům. Ovládání je však logické a po krátkém seznámení s rozložením prvků nemá uživatel problém doklikat se kam potřebuje.

Nenašel jsem možnost spravovat více projektů zároveň.

Systém Bugzilla obsahuje velké množství zajímavých funkcí odpovídajících našemu zadání. Je znát, že aplikace je určena pro *„issue tracking“* a ne samotnou správu úkolů a projektů. Z našeho testování máme podobný pocit, jako z aplikace Redmine. Jedná se o velmi dobrou aplikaci, která by potřebovala spoustu úprav ve správě stránek, hostingů, klientů a implementaci stopek. Vzhledem k tomu, že celá aplikace je napsána ve skriptovacím jazyce

Perl, se kterým nemáme moc zkušeností, jednalo by se pro společnost o časově náročné úpravy.

5.1.3 Redmine

Redmine [22] je zajímavá volba mezi správci projektů. Nabízí spoustu užitečných funkcí jako je Wiki, samostatná správa dokumentů, kalendář, a dokonce plánování pomocí Ganttových diagramů. [10]

Správa úkolů je velmi dobře zpracována, úkolům je možno přiřadit strávený a odhadovaný čas, jedná se o požadavky, které by nebylo třeba doprogramovat, pouze lehce upravit. Systém však nezvládá čas samostatně měřit, což je škoda, a bylo by potřeba vytvořit funkci zaznamenávání času. U jednotlivých úkolů je možné nastavit stavy, které lze v administraci upravovat. Je možné nastavit cyklus z jakého stavu se může úkol přesunout do jiného. Výhodou je možnost spravovat několik projektů současně.

Správa uživatelských rolí je propracovaná a lze tvořit uživatelské skupiny. Přidávání oprávnění uživatelům není problém. Pro zobrazování úkolů existuje velký počet filtrů, podle kterých lze zúžit počet úkolů přesně dle potřeby. Na úvodní stránku – „moje stránka“ je možné přidat výpis úkolů podle daného filtru. Každý uživatel si může nastavit zobrazení podle vlastních preferencí. Úkoly lze přiřazovat do kategorií, které je možné v administraci upravovat, což je vítaná vlastnost. Pracovníci i klienti mohou k úkolům přidávat přílohy a nechybí ani možnost konverzace. Při každé změně je možné změnit stav úkolu a zaznamenat čas strávený na úkolu.

Redmine je velmi povedené řešení, které neohromí svou grafickou stránkou, ale svými funkcemi. Celý systém je implementován v jazyce Ruby, přesněji na frameworku Ruby on Rails. Podpora databází je dostatečná od MySQL, PostgreSQL a SQLite. Pro stažení a upravování je nutno dodržet licenční podmínky GNU GPL 2. [34]

Ze všech zkoumaných řešení vychází Redmine jako nejzajímavější řešení pro správu úkolů. Jeho správa více projektů, dokumentů a Wiki z něj dělají výborného kandidáta. Nesplňuje však některé z požadavků, takže by bylo nutné doprogramovat velké množství funkcí. Bylo by potřeba přidat správu webových stránek a hostingů, správu klientů a měření časů. Problémem může být také jazyk Ruby on Rails, se kterým nemá nikdo z firemních programátorů zkušenost. Pochopení základů tohoto jazyka a nastudování velmi dobré dokumentace systému Redmine by mohlo trvat déle než vývoj nového systému na míru.

Nutno podotknout, že společnost systém Redmine využívala po dobu téměř dvou let pro některé z projektů a hlavním problémem bylo nepochopení jeho funkcí ze strany klienta. Někteří klienti odmítli tento nástroj používat, protože jim každý úkon trval mnohonásobně déle, než zavolat nebo poslat email přímo.

5.1.4 Wunderlist

Wunderlist [35] je jednoduchý správce úkolů, který nesplňuje téměř žádné z námi požadovaných kritérií, nejedná se totiž o pokročilý systém obsahující velké množství funkcí. Wunderlist je aplikace, kterou je možné spustit na většině zařízení, podpora je pro všechny hlavní počítačové operační systémy Windows a macOS. Na Linux zatím nemá

plnohodnotnou aplikaci, což ale uživatelům tohoto systému nebrání v jejím používání. Wunderlist má webovou aplikaci, kterou je možné spustit v internetovém prohlížeči. Výhodou multiplatformního přístupu jsou také aplikace pro mobilní telefony všech operačních systémů.

Tato aplikace se zaměřuje na správu úkolů. Neumí nic víc a není možné si ji jakkoli upravit, kromě barvy pozadí plochy zobrazující seznamy úkolů. V čem je však tento úkolovník šikovný, je jakým způsobem lze s úkoly pracovat. Každý z úkolů je možné přiřadit do složky, které lze přiřadit uživatele pozváním pomocí emailu. Úkolům je možné nastavit datum, do kdy má být daný problém splněn. Je zde šikovné pole pro přidávání popisku úkolů a možnost vkládat přílohy. Při každé změně je dole možné vyčíst, kdy k ní došlo. Bohužel se nelze podívat na seznam všech změn u úkolů – historie změn. Každý úkol má vlastní část věnovanou konverzaci, lze se například domlouvat s kolegy nebo klientem, zda je již úkol splněn dle požadavků. V konverzaci je možno označit uživatele pomocí jednoduchého symbolu *@jméno*. Toto je důležité, protože tato aplikace podporuje funkci notifikací, tedy kdykoli nastane událost, která se týká aktuálně přihlášeného uživatele, obdrží upozornění o dané změně. Wunderlist nemá dobře řešené vnořené úkoly. Možnosti filtrování úkolů by šlo také rozšířit. Líbila by se nám větší propojenost s kalendářem a možnost exportovat listy, ale to jsou funkce, které nemají vysokou prioritu.

Z grafického hlediska se nám líbí jednoduchost a možnost přetahovat úkoly pomocí „drag & drop“ a podpora notifikací, které jsou velmi užitečné. Wunderlist sice dokáže pracovat s přizvanými uživateli, avšak nastavení práv tu není. Aplikace sice nesplňuje většinu požadavků, ale pokud se rozhodneme jít cestou vlastní aplikace, jedná se o výborný zdroj inspirace.

Wunderlist je možné využívat zdarma, v tom případě ale neobsahuje některé z pokročilejších funkcí. Po dobu testování jsem neměl potřebu využívat aplikaci v PRO verzi, vše, co jsem potřeboval, uměla i verze základní.

5.1.5 Todoist

Todoist [36] bych zařadil do podobné kategorie, jako výše popsaný Wunderlist. Jedná se o samostatnou aplikaci sloužící pro jednoduchou správu úkolů, bez žádných pokročilejších možností nebo administrace. Po dobu testování se nám však zalíbilo několik funkcí, které by bylo vhodné zmínit.

Todoist na rozdíl od Wunderlistu [35] funguje na bázi předplatného za každého uživatele. Cena za jednoho uživatele měsíčně vychází na ekvivalent 2,5 € při využití předplatného na celý rok. Ve verzi zdarma Todoist nenabízí nic víc než Wunderlist, dokonce i méně. Chybí například upozornění, komentáře, vkládání příloh. Jeho silnou stránkou je řešení podprojektů a vnořených úkolů. Přesouvání a orientace v seznamu úkolů je mnohem přehlednější. Výborná je možnost vytvářet si šablony úkolů, které lze jednoduše naimportovat. Tato funkce se může hodit například pro různé checklisty, které jsou vždy stejné pro každý projekt. [37]

Stejně jako Wunderlist je tato aplikace nerozšiřitelná a bereme ji pouze jako inspiraci. S tímto programem se nám pracovalo příjemně, ale měsíční paušál je nevýhodný, obzvláště

pro nové zaměstnance a externisty, kterých nevyužíváme pravidelně, ale potřebují přístup do správy úkolů.

5.1.6 Trello

Tato aplikace funguje na jiném principu než všechny dosud testované. Původ aplikace vychází z japonského konceptu Kanban, který v překladu znamená „cedule“ nebo „billboard“. Tento koncept využívala japonská automobilka Toyota při využívání metodiky JIT (Just In Time). [38]

Osobně bych ji přirovnal k whiteboardu rozdělenému na několik sloupců, které si lze definovat. Těchto ploch si může uživatel vytvořit kolik chce, dá se tedy pro každý projekt vytvořit vlastní tabule s různými sloupci dle potřeby. Sloupce je možné přejmenovat. Základní tři sloupce jsou *ToDo*, obsahující položky, které je potřeba dokončit. *Doing* je sloupec zabývající se rozdělanými úkoly a *Done* jsou hotové úkoly čekající například na schválení. Úkoly lze pomocí přetažení z jednoho sloupce přesunout do jiného, jedná se o jednoduché a intuitivní řešení.

Každému úkolu je možné přiřadit název, popisek, přílohy a termíny, do kdy má být úkol hotov. Systém také umožňuje vkládat seznamy dílčích úkolů, bohužel se jedná opravdu jen o seznam checkboxů a nelze nastavit, kdo má dílčí úkol řešit. Těchto seznamů lze k jednomu *lístku* přiřadit několik. Výborně je řešena funkce štítků, díky kterým lze v aplikaci jednoduše filtrovat úkoly. U úkolů je řešená i historie změn, která se však mísí s komentáři, což dělá tuto funkci lehce nepřehlednou.

Mezi zajímavé funkce patří zobrazení dlouho stojících úkolů, na kterých se nic nezměnilo. Tyto úkoly postupem času získávají texturu starého papíru. Jedná se o elegantní a zároveň přehledný způsob, jak upozornit na neaktivní úkoly.

Přílohy jsou řešeny pomocí nahrávacího tlačítka, zároveň je tu i možnost napojení na Google Dokumenty. [24] Díky tomu není potřeba při každé změně dokumentu nahrávat novou verzi. Systém místo dokumentu vloží pouze odkaz na daný dokument a pro jeho zobrazení se otevře nová záložka webového prohlížeče.

Oprávnění lze nastavit minimálně, v aplikaci nejsou podrobná práva. Lze pouze omezit, kdo může komentovat a lze přizvat další členy k dané tabuli. U každého ze sloupců je možnost posílání notifikací a emailů při provedené změně, což přináší přehled o dění. Tato funkce může být důležitá například pro projektového managera.

Velkou výhodou Trella jsou jeho nativní desktopové a mobilní aplikace.

Věřím, že plochy by bylo možné upravit dle potřeby, avšak způsob, jakým Trello třídí úkoly, je jiný než u všech ostatních systémů. Nejsem si jist, zda by vyhovovalo našim potřebám. Vzhledem k tomu, že Trello neobsahuje mnoho funkcí, které jsou požadovány, jedná se především o inspiraci, jak lze řešit správu projektů. Aplikaci je možné využívat zdarma, obsahuje však i premium verzi, která ale nepřináší žádné důležité funkce.

5.1.7 JIRA

JIRA [39] je aplikace od australské společnosti Atlassian, což je stejná společnost, která vytvořila systém Bitbucket využívaný ve společnosti pro verzování souborů.

Aplikace má možnost nastavit si takzvaný Dashboard dle vlastních představ. Jedná se o úvodní obrazovku zobrazující se po přihlášení do systému. Na této ploše si může uživatel pomocí gadgetů nastavit, jaké informace ho zajímají a jsou pro něj podstatné. Informační plochy lze nastavovat a přesouvat dle libosti. Uživatel také není omezen pouze na jeden Dashboard, ale má možnost si jich vytvořit více s rozdílnými gadgety. Tuto funkci shledáváme velmi užitečnou, ne každý zaměstnanec potřebuje vidět stejné informace.

U úkolů lze nastavit snad vše požadované, jen měření času není součástí systému, tato informace se ke každému úkolu dá doplnit několika způsoby: začátek a konec práce nebo vložením pracnosti. Všechny události se ukládají k úkolu, jejich historii je kdykoli možné si prohlédnout. V systému jsme nenašli možnost tvořit vnořené úkoly, avšak textový editor umožňuje vkládat odkazy na jiné úkoly přímo. Textový editor je celkově povedený, obsahuje pokročilé možnosti formátování a integraci s dalšími službami společnosti Atlassian.

Překvapivě některé úkony trvají dlouho a na obrazovce se zobrazuje indikátor, že systém pracuje. Při delší práci s aplikací na tento fakt uživatel zapomene, ale točící se kolečko nebo nekonečný progres bar se zobrazuje téměř při každé aktivitě.

Aplikace i při nejmenších změnách odesílá upozornění na email, což je rušivé a v době testování jsme nenašli možnost nastavit si, které informace si uživatel přeje sledovat.

JIRA poskytuje několik možností, jak přistupovat k projektovému řízení, od agilních metodik vývoje přes Kanban podobný aplikaci Trello, až po řízení úkolů jako takových.

Za velkou výhodu považujeme obchod s moduly a povedené měření časů úkolů. Líbí se nám možnosti, jak se dá zapisovat strávený čas. Aplikace nabízí možnost měření, nebo zadání času stráveného nad úkolem ve formátu například 12h 10m, kdy dokáže rozlišit hodiny a minuty.

Základní verze má příjemnou cenu 10 $ za 10 uživatelů. Bohužel s přibývajícím počtem uživatelů cena roste. V Tabulka 4 je vidět nárůst cen. Společnost počítá s náborem nových a externích zaměstnanců a je na zvážení, zda platit vysoké měsíční paušály.

Počet uživatelů	Cena za rok [$]
1–10	100
11–15	750
16–25	1500
26–50	3000
51–100	4500

Počet uživatelů	Cena za rok [$]
101–500	7500
501–2000	15000

Tabulka 4: Tabulka cen aplikace JIRA

5.1.8 Zhodnocení

Z testovaných systémů jsme usoudili, že žádné z existujících řešení nesplňuje požadavky společnosti bez větších zásahů do struktury aplikace. Ani jedna z aplikací neodstraňuje potřebu využití aplikací třetích stran, protože neumožňuje čas měřit, pouze zapisovat. Většina existujících systémů se zaměřuje na správu projektů a úkolů, vlastnosti systému, který si společnost představuje, jsou však centrální uložiště informací. Systémy jako Redmine nebo Bugzilla mají dobře vyřešenou problematiku úkolů, ale úprava systému v jazycích, ve kterých jsou napsány, je pro naši společnost nereálná. Jednoduché správce úkolů Wunderlist a Todoist sloužily jako výborná inspirace. Velmi se nám líbila aplikace JIRA od společnosti Atlassian, která je velmi povedená a splňuje mnoho našich požadavků, její cena je však pro rostoucí společnost překážkou. Společnosti v aktuálním stavu přijde výhodnější postupně si vytvořit vlastní aplikaci, která bude řešit specifické funkce i přes možnost ušlého zisku po dobu vývoje.

6 Analýza a výběr použitelných technologií

V této kapitole se zaměřím na analýzu problému a zjistíme, jaká rizika vývoj vlastní aplikace přináší. Následně zvolíme programovací jazyk, který nejvíce hodí pro implementaci systému a je vhodný po naše programátory.

6.1 SWOT analýza

V následující tabulce je zobrazena krátká SWOT analýza, ze které lze vyčíst hlavní výhody a nevýhody aplikace.

Jako hlavní hrozbu vidíme časovou náročnost a tím i cenu. Aplikaci plánujeme vytvářet převážně ve volném čase, případně v období, kdy společnost čeká na vyjádření klientů k zakázkám. Pokud budeme na této aplikaci pracovat v průběhu pracovní doby, je nutné s počítat strávený čas jako s ušlý zisk.

Hlavní silnou stránkou aplikace je vlastní řešení a možnost postupného dodělávání vlastních funkcí dle potřeby.

Za slabé stránky považujeme podporu a dokumentaci, kterou si musíme dělat na vlastní náklady a čas.

Mezi příležitosti jsme přidali možnost naučit zaměstnance novým technologiím a postupům. Společnost si vždy přála, aby zaměstnanci pracovali na interních projektech, na kterých se mohou rozvíjet, provádět code review a zdokonalovat své dovednosti, na které v průběhu klasického pracovního dne není čas. Očekáváme, že díky novým zkušenostem spojených s vývojem vlastního systému a změnám interních procesů se zaměstnancům bude lépe a produktivněji pracovat.

Strengths	**Weaknesses**
Vlastní aplikace	Podpora
Možnost rozšíření	Dokumentace
Funkce dle požadavků	
Chybějící software na trhu	
Opportunities	**Threats**
Využití zaměstnanců	Časová náročnost
Code review	Cena
Učení nových zaměstnanců	
Zlepšení procesů	
Reference	

Obrázek 17: SWOT analýza

6.2 Analýza rizik

Následující seznam rizik zobrazuje možné problémy, se kterými se během vývoje můžeme potkat. Každé z rizik je označeno očekávanou pravděpodobností a dopadem v případě jeho uskutečnění. Dopad je hodnocen 1 až 4 (1 znamená kritický, 4 minimální dopad).

Skryté požadavky

Návrh a vývoj aplikace je tvořen přímo v prostředí společnosti. Cíle jsou předem specifikované dle požadavků jednotlivých zaměstnanců. Věříme, že další požadavky vzniknou po nasazení aplikace, kdy bude docházet k dalšímu vývoji dle potřeb společnosti. Dopad na fungování společnosti bude minimální, pozdější úpravy nejsou problém. Naopak jsou vedením vítány.

Pravděpodobnost: 25 %

Dopad: 4

Nezkušenost

Návrhem a implementací takového systému se zabývám poprvé, je tu přirozeně vyšší riziko. Mám však na své straně zkušené kolegy, se kterými mohu vzniklé problémy řešit. Časté konzultace se zadavatelem, vedoucím práce a intenzivní studium projektů s podobnou problematikou by mělo snížit toto riziko.

Pravděpodobnost: 80 %

Dopad: 2

Špatný odhad pracnosti

Dle sesbíraných požadavků víme, že se nebude jednat o malý projekt. Funkčnost je navržena tak, aby se dala stihnout v plánovaném termínu. V případě vyšší časové náročnosti je společnost smířena s dokončením zbylých i následných požadavků po odevzdání této práce. Priority pro dokončení prototypu jsou uvedeny v tabulce funkčních požadavků v části. Stanovení harmonogramu prací a zvolení dostatečné časové rezervy nám pomůže minimalizovat možný časový skluz.

Pravděpodobnost: 50 %

Dopad: 3

Defektní produkt

Vzhledem k velikosti aplikace se počítá s možnými chybami. Těm se dá předejít pomocí pravidelného testování již od začátku prací. Počítáme s možností nasazení každého většího bloku na testovací prostředí, kde bude přístupný všem zaměstnancům a následně bude zařazen do ostrého provozu. Díky testování několika zaměstnanci se možné chyby odhalí mnohem rychleji.

Pravděpodobnost: 60 %

Dopad: 2

Přetížení aplikace

Aplikace bude využívána především interními zaměstnanci společnosti a klienty. V nejhorších případech společnost očekává několik desítek aktivních uživatelů ve stejnou chvíli. Pro předejití tohoto rizika můžeme provést zátěžové testy, které nám odhalí, jaké jsou hranice aplikace. Více než přetížení aplikace může být problém s výpadkem hostingové služby, na které bude aplikace umístěna. To lze řešit správným výběrem hostingu a smluvní garancí dostupnosti serveru.

Pravděpodobnost: 30 %

Dopad: 2

Bezpečnostní rizika

Všechna hesla a údaje ukládané do databáze musejí být uložena v bezpečném formátu, které nemůže útočník zneužít. Ve většině programovacích jazyků existují frameworky, které tyto problém řeší. Aplikace by však stejně měla být otestována proti běžným typům útoků.

Pravděpodobnost: 30 %

Dopad: 1

Nedodržení zadání

Všechny požadavky jsou popsány v kapitole zabývající se definováním funkčních požadavků. Následná závěrečná kontrola, zda systém obsahuje vše dle zadání, by měla eliminovat toto riziko.

Pravděpodobnost: 30 %

Dopad: 2

6.3 Programovací jazyky

Společnost chce, aby systém běžel na všech zařízeních bez omezení operačního systému. Tento fakt přímo nahrává možnosti vytvořit centralizovanou webovou aplikaci. Pro tento typ aplikací existuje celá řada technologií a záleží pouze na preferencích, kterou si vybrat. Mezi nejčastěji používané webové technologie dnes patří ASP.NET, Java EE, PHP, Javascript, Ruby a další scriptovací jazyky, za předpokladu využití některého z frameworků. [40] V této kapitole se zaměřím na výběr nejvhodnějšího z jazyků pro implementaci interního systému.

Budeme se přiklánět k jazykům, se kterými má některý z firemních programátorů zkušenosti, protože následné úpravy a dodělávání funkcí jsou jedním z hlavních požadavků. Proto se budeme zaměřovat na jazyky s dostatečnou rozšířeností. [41] Častěji využívané jazyky mívají lépe řešenou dokumentaci a je možné najít více rad na internetu v případě nejasností.

Jedním z důvodů je cena. Společnost nechce hledat pro tento jediný projekt placené technologie, když lze využít alternativních řešení. Jednou z podmínek také je, aby si programovací jazyk rozuměl s open source typem databází. Aplikace by následně měla běžet na firemním serveru, znovu je důvodem minimalizace nákladů.

I přes fakt, že aplikace bude cílit na desítky, maximálně stovky uživatelů v živém provozu, je potřeba, aby programovací jazyk byl dostatečně výkonný. Přístup k aplikaci bude převážně nárazový, mezi typické úkonu bude patřit přidání úkolu nebo jeho dokončení, případně přidaní komentáře nebo vyhledání přístupových údajů. Nejedná se tedy náročné operace, vyžadující vysokou výkonnost. Nejedná se tedy o nejdůležitější parametr a společnost se přiklání k volbě jazyků, se kterými již má zkušenosti.

6.3.1 ASP.NET

Programovací jazyky běžící na platformě .NET jsou vyvíjeny společností Microsoft. Jsou určeny především pro systémy Windows a webový server IIS. Existují i open source alternativy pro použití této platformy. Jednou z nich je Mono. Využitím open source řešení se vystavujeme určitým nekompatibilitám, které mohou způsobit prodloužení a prodražení vývoje.

Pro vývoj aplikací v ASP.NET poskytuje Microsoft propracované IDE nazvané Visual Studio. V době výběru vhodného programovacího jazyka bylo dostupné pouze pro OS Windows. Dnes již je možnost využít aplikace i pro Linux a OSX, ale aplikace Visual Studio Code na těchto systémech nedosahuje kvalit Windows verze. Napojení na jiný webový server, než IIS je možné, ale jedná se o složitější řešení, ke kterému se na oficiálních stránkách vývojářů nic nedočteme. Výběr serverů poskytujících IIS je však dostačující a lze si vybrat ze známých poskytovatelů s podobnou cenou jako server pro PHP.

Výhodou tvorby aplikací na platformě ASP.NET je možnost používat různé jazyky, nemusí se jednat pouze o C# vytvořený společností Microsoft. Lze využít i skriptovacích nebo

kompilovaných jazyků. Na internetu existuje spousta již vytvořených komponent, které se mohou hodit při vývoji nových aplikací. Platforma jako taková je považována za velmi bezpečnou, ale dle zvoleného jazyka náročnější na paměť. Při velkých počtech návštěvníků může být lehce pomalejší než konkurence. Tento problém se námi navrhované aplikace netýká.

Platforma .NET je platforma s velkou komunitou, a tedy i dobrou podporou pro nové vývojáře (například na portálu MSDN) zaměřující se především na zákazníky vlastních produktů z řady společnosti Microsoft. Nutnost využívat systému Windows může komplikovat vývoj aplikace pro vývojáře využívající jiné operační systémy.

6.3.2 Java EE

Java EE neboli Java Enterprise Edition je velmi populární programovací jazyk. Je využíván především u velkých aplikací a bankovních systémů. Jeho výhodou je návaznost na standardní jazyk Java, díky tomu jsou aplikace multiplatformní a lze je spustit na téměř kterémkoli operačním systému. Na rozdíl od klasické Javy, přináší komponenty a služby určené pro webové aplikace, možnosti asynchronního zpracování dat.

Velkou výhodou pro velké projekty může být, že Java EE přirozeně vede k rozdělení vývoje do rolí. Lidé s odlišnými dovednostmi uplatňují své znalosti. Toto je způsobeno především rozsáhlostí projektů vytvářených v tomto jazyce.

Mezi hlavní výhody Java EE patří její vysoká spolehlivost a dostupnost. Java nutí programátory přemýšlet a programovat objektově, což může být také bráno jako výhoda. Nemůžeme zapomenou ani na bezpečnost a možnosti testování. Díky těmto vlastnostem je ideální pro aplikace většího rázu s potřebou kvalitního zabezpečení. Stejně jako u většiny jazyků je program bezpečný, tak jak je bezpečný jeho kód.

Mezi nevýhody může patřit škálovatelnost a rozšiřitelnost. Pro vývoj aplikací je potřeba vytvořit výborný model, podle kterého se bude vyvíjet. Pokud dojde ke změně modelu, může to znamenat velkou změnu do celého programu.

Ani jeden z našich programátorů nemá velké zkušenosti s tímto programovacím jazykem a myslíme si, že se jedná o zbytečně komplikované a časově náročné řešení. Vzhledem k nedostatečným znalostem by vývoj trval příliš dlouho a využití externích programátorů by se mohlo prodražit.

6.3.3 Ruby a Python

Ruby a Python jsou scriptovací jazyky, které využívají frameworků pro tvorbu webových aplikací. Jejich výhodou bývá jednoduchost a díky tomu rychlost učení pro nováčky. Jejich popularita je však minimální. Dle statistik [42] se jedná o jednotky procent, v České republice i méně. Díky tomuto faktu pro nás ztrácí smysl uvažovat nad vývojem v těchto jazycích.

6.3.4 PHP

Jazyk PHP je ze všech webových programovacích jazyků nejrozšířenější, dle statistik [43] je více než 75 % všech webových stránek postaveno na tomto jazyce.

Jeho hlavní výhodou je relativní jednoduchost a srozumitelnost. Pro tento jazyk existuje nespočet frameworků usnadňujících práci, zlepšujících kvalitu kódu, a především zvyšujících bezpečnost. Jazyk sám o sobě nenutí uživatele psát objektový kód, ale s výběrem správného frameworku uživateli nic jiného nezbývá a člení kód do částí pro Model View a Controller, tj. MVC [44]. Hledání chyb pomocí debuggeru je však mnohem horší než v jiných jazycích.

Výběr hostingu by neměl být problém, téměř každý poskytovatel ho nabízí, jen je potřeba vybírat hosting kompatibilní s poslední verzí 7. PHP si dobře rozumí s většinou databázových serverů, jako je MySQL, MSSQL nebo PostgreSQL. Pro webový server se dá použít Apache, který je nezávislý na platformě. [45]

Na rozdíl od ASP.NET nebo Java EE se nejedná o kompilovaný jazyk a je tedy zpravidla pomalejší. Tento fakt může být i výhodou při rychlé potřebě upravit chybu bez potřeby kompilování celého projektu.

Ze zkušeností našich programátorů víme, že potřebné IDE není problém najít. Je možné použít Netbeans, PhpStorm, Sublime Text nebo Atom. Všechna IDE by měla být pro naše účely dostačující.

6.3.5 Javascript

V posledních letech se objevuje čím dál více aplikaci napsaných pomocí javascriptových frameworků. Javascript již není pouze jazykem pro jednoduché výpočty a hýbání HTML elementy, ale dají se v něm psát celé aplikace. Mezi nejrozšířenější frameworky patří Angular a React.

React

Tento relativně nový framework je vyvíjen společností Facebook, která ho využívá i na své sociální síti. Tento framework přímo vyzývá k využití pro SPA (Single Page Application). Jeho hlavní výhodou je využití standardního javascritpu rozšířeného o knihovnu React. Ta umožňuje vývoj jednotlivých částí programu jakožto komponent. Každá komponenta si uchovává své stavy a v případě potřeby se překresluje. Nedochází však k obnovení celé stránky, ale pouze dané komponenty. Aby šla obnovovat pouze část webové aplikace je využit tzv. virtuální HTML DOM, který React vytváří a obsluhuje. Tento přístup velmi zrychluje používání aplikace, protože není potřeba překreslovat celou stránku, místo toho si aplikace pouze zažádá o informace a ty zobrazí.

Díky rozšíření React Native lze vytvářet nejen webové aplikace, ale i programy fungující na mobilních zařízeních se systémy iOS a Android. Možnost vytvořit jednu aplikaci, fungující nejen ve webovém prohlížeči, ale také na mobilních systémech, je zajímavá. Pro chod navrhované aplikace není nutná.

Jako výhodu vidíme modularitu systému a možnost využití jedné aplikace pro chod na všech zařízeních.

Mezi nevýhody aktuálně řadíme nedořešené vývojářské prostředí a fakt, že se jedná o relativně novu technologii. Obáváme se především možnosti zaseknutí se na banálních problémech.

Pokud bychom pracovali s frameworkem React, nevyužívali bychom přímý přístup k databázi, ale zvolili bychom možností webové Rest API.

Pozn.: Pokud bychom navrhovali aplikaci na konci psaní této práce, velmi rádi bychom zvolili React, jako hlavní programovací jazyk. Za dobu, psaní této práce rozšířila komunita tohoto frameworku jeho možnosti a vývojářské prostředí o velké množství výhod a věříme, že by se jednalo o správnou volbu.

AngularJS

Tento jazyk je relativně, první verze vznikla v roce 2010 a je vyvíjen společností Google. Již tento fakt nám říká, že tento framework dostává pravidelná vylepšení v podobě aktualizací. [46]

Obdobně, jako v Reactu lze tvoři komponenty, ale způsob zápisu je rozdílná, více podobná klasickému HTML a javascriptu. AngularJS lze velmi dobře kombinovat s jinými javascriptovými frameworky typu jQuery nebo Google Closure.

Stejně jako React se tento framework snaží o objektový přístup, čehož si především firemní programátoři cení. Technologie AngularJS je založena na tzv. *Two Way Data-Binding.* Jedná se o velmi užitečnou vlastnost řešící synchronizaci stavů mezi modelem a view. Jde především o přenášení dat z formulářů do modelu a následně z modelu do ostatních view. Tato funkce je implementována v mnoha frameworcích, avšak v AngularJS je automatizována a šetří mnoho kódu.

Dokumentace je velmi kvalitní a přehledná. Komunita uživatelů je dostatečná a je možná najít mnoho volně dostupných materiálů a dokumentů, pomáhajících k rychlému učení nového jazyka.

S tímto frameworkem nemáme žádné zkušenosti, ale zdá se, že se jedná o ideální doplněk pro určité funkce. Nejsme si jisti, zda je vhodný pro naše řešení komplexní aplikace.

6.3.6 Závěr

Z výše popsaných jazyků a vývojových podmínek, rozšířenosti a osobních zkušeností jsme zúžili výběr na ASP.NET, PHP a Javascript. S Java EE nemá žádný z programátorů zkušenost a naučit se v tomto jazyce programovat by bylo zbytečně náročně oproti ostatním volbám. ASP.NET je sympatická platforma s velkým množstvím uživatelů, bohužel je omezena na jeden operační systém, a i přes její bezpečnost a kvalitu výsledných aplikací se jedná o omezení, které společnosti nevyhovuje.

Společnost lákalo zkusit vytvořit aplikaci v jazyce, který získává čím dál větší přízeň programátorů. Chybějící zkušenost s tvorbou velkých aplikací v tomto jazyce nás však odradila. Vzhledem k modularitě frameworků React a AngularJS uvažujeme o použití javascriptu pro některé části aplikace, ale ne pro systém jako takový.

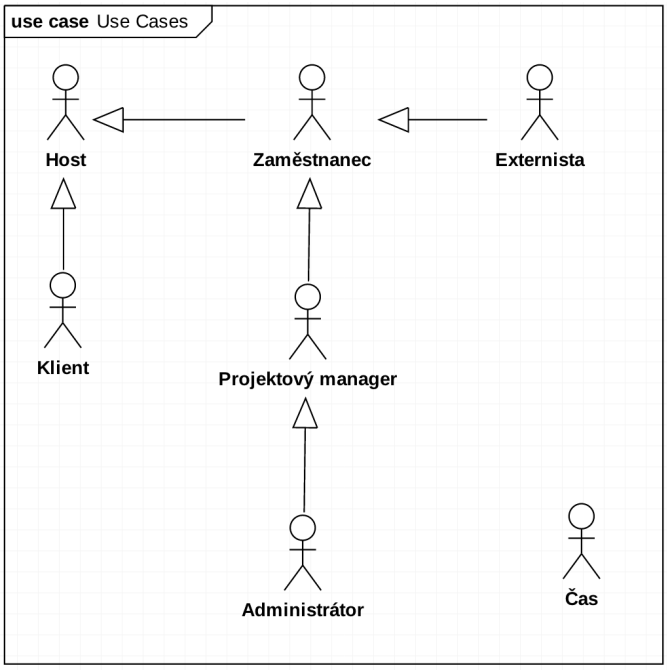
Nejzajímavější variantou je PHP, se kterým má společnost dlouholeté zkušenosti a vlastní všechny potřebné nástroje pro vývoj v tomto jazyce. Díky zkušenostem by byl vývoj jednodušší a následné rozšiřování aplikace by se dalo přiřadit kterémukoli zaměstnanci.

7 Návrh aplikace

V této kapitole se zaměříme na samotný návrh aplikace. Nadefinujeme aktéry, určíme důležité funkční požadavky a nastavíme jejich prioritu dle potřeb společnosti. Dle sesbíraných funkčních požadavků vytvoříme případy užití a nastíníme některé z grafických prvků aplikace.

7.1 Aktéři

V kapitole **Chyba! Nenašel se zdroj odkazu.** o uživatelských rolích jsme definovali typy uživatelů, kteří budou využívat vyvíjený informační systém. Nadefinovali jsme si 6 uživatelských rolí. Z podstaty aplikace se jedná o přiměřené množství. Každá ze skupin je specifická svým využíváním aplikace a potřebnými funkcemi. Máme tedy tyto aktéry: *host*, *externista*, *zaměstnanec*, *projektový manager*, *administrátor*, *klient*. Vztahy mezi nimi znázorňuje Obrázek 18. Nejedná se o přímé dědění vlastností, spíše o vymezení oprávnění. Role je možné kombinovat, například role *administrátor* v základu nemá možnost měřit čas, přiřazením role *zaměstnanec* tuto funkcí získá.



Obrázek 18: Diagram aktérů

Základní aktér je *host* umožňující vstup do systému pomocí přihlašovacího formuláře. Ostatní aktéři od *hosta* tuto vlastnost dědí.

Společnost bude většinou využívat několik skupin uživatelských rolí. V tabulce Skupiny uživatelských rolí jsou zobrazeny nejčastější kombinace rolí. Role *zaměstnanec* umožňuje uživatelům měřit čas, role *projektový manager* spravovat více informací o projektech, klientech a časových plánech a *administrátor* tyto funkce rozšiřuje o správu uživatelů.

Uživatelská role	Systémové role
Klient	Klient
Externista	Externista, zaměstnanec
Zaměstnanec	Zaměstnanec
Projektový manager	Zaměstnanec, Projektový manager
Administrátor	Zaměstnanec, Projektový manager, Administrátor

Tabulka 5: Skupiny uživatelských rolí

V systému bude také figurovat čas jako aktér. Jeho prací bude vykonávat určité úkoly v daný čas, například automatické odesílání emailů nebo uzavírání otevřených časů na konci dne. [47]

7.2 Funkční požadavky

	Funkční požadavek	Priorita
	Přihlášení	
FP 01	Uživatel se může přihlásit do systému pomocí uživatelského jména a hesla	1
FP 02	Uživatel se může odhlásit ze systému	1
FP 03	Uživatel může změnit své heslo	1
	Správa uživatelských rolí	
FP 04	Administrátor může spravovat (přidávat, odebírat) uživatelské role, registrovaných uživatelů	1
FP 05	Administrátor může vytvářet nové uživatelské účty	1
FP 06	Administrátor může aktivovat a deaktivovat uživatelské účty	1
FP 07	Uživatel může změnit informace o svém účtu, jméno a email	1
	Správa úkolů	
FP 08	Uživatel může upravovat detail úkolu, ke kterému má přístup	1
FP 09	Uživatel může dokončit úkol, ke kterému má přístup	1
FP 10	Uživatel může přidávat úkoly k projektu, ke kterému má přístup	1
FP 11	Uživatel může přidávat komentář k úkolu, ke kterému má přístup	3
FP 12	Uživatel může přiřadit úkol jinému uživateli, pokud má přístup k danému úkolu	1
	Připomínky	
FP 13	Uživatel může vytvářet nové připomínky	1
FP 14	Uživatel může editovat stávající připomínky	1
FP 15	Uživatel může smazat nebo dokončit existující připomínky	1
FP 16	Uživatel může přiřadit připomínku jinému uživateli	1

	Funkční požadavek	Priorita
	Odpracovaný čas	
FP 17	Uživatel si může zobrazit své aktuálně odpracované časy	1
FP 18	Uživatel může editovat již vložené časy	2
FP 19	Uživatel může přidávat nové časové záznamy ručně	2
FP 20	Projektový manager a administrátor si může zobrazit přehled časů strávený na všech projektech	1
FP 21	Uživatel může ve výpisu projektů a zúžit hledané informace pomocí filtrace	2
	Projekty:	
FP 22	Uživatel si může zobrazit seznam všech projektů, ke kterým má přístup	1
FP 23	Uživatel si může zobrazit detail projektu, ke kterému má přístup	1
FP 24	Uživatel může měnit detail projektu, pokud má dostatečné oprávnění	1
FP 25	Projektový manager a administrátor může vytvářet nové projekty	1
FP 26	Projektový manager a administrátor může přidávat externí zaměstnance ke zvolenému projektu	2
FP 27	Projektový manager a administrátor si může zobrazit podrobné informace o projektu obsahující informace, které uživatel bez oprávnění nevidí (ceny, časovou náročnost, kontakty)	1
FP 28	Uživatel může k projektu nahrávat soubory	2
FP 29	Projektový manager a administrátor si může zobrazit přehled časů strávený na všech projektech	1
FP 30	Uživatel může ve výpisu projektů a zúžit hledané informace pomocí filtrace	2
FP 31	Systém automaticky ohlásí uživatele po nastavené době neaktivity	1
FP 32	Systém automaticky ukončí práce na projektech z předchozího dne a přidá k nim poznámku o jejich automatickém uzavření	2
FP 33	Systém automaticky ukončí pravidelné měsíční práce k poslednímu dni měsíce	3
FP 34	Systém automaticky vytvoří nový projekt pro pravidelné měsíčné práce k prvnímu dni v měsíci	3
FP 35	Systém bude umožňovat přihlášení do aplikace pomocí emailu a hesla na základě předešlé registrace, na straně společnosti	1
FP 36	Systém bude vyžadovat přihlášení uživatele pro zobrazení dat uložených v aplikaci	1
FP 37	Uživatel může zobrazovat detail projektu	1
FP 38	Uživatel může k projektu nahrávat soubory	3
FP 39	Systém bude umožňovat odebírání externích zaměstnanců z projektů	2
FP 40	Uživatel může upravovat detail úkolu, ke kterému má přístup	1
FP 41	Uživatel může dokončit úkol, ke kterému má přístup	1
FP 42	Uživatel může přidávat úkoly k projektu, ke kterému má přístup	1
FP 43	Uživatel může přidávat komentář k úkolu, ke kterému má přístup	3

		Funkční požadavek	Priorita
FP	44	Uživatel může přiřadit úkol jinému uživateli, pokud má přístup k danému úkolu	1
FP	45	Projektový manager nebo administrátor může vytvořené složce projekt	1
FP	46	Systém bude umožňovat přesouvání úkolů mezi projekty pomocí "drag and drop" funkcionality	2
FP	47	Systém bude umožňovat vytvářet nové úkoly ve vybrané složce	1
FP	48	Systém automaticky vytvoří základní složku pro každého uživatele nazvanou inbox	1
FP	49	Systém bude umožňovat uživatelům měnit informace a stavy úkolu	1
FP	50	Systém bude umožňovat komunikaci na úrovni úkolu pomocí jednoduchého komunikačního okna	3
FP	51	Systém bude informovat uživatele o nových úkolech a změnách v již vytvořených úkolech, ke kterým je daný uživatel přiřazen, pomocí notifikací.	3
FP	52	Systém bude u úkolů automaticky přidávat základní stav "vytvořen". Tento stav získá každý nový úkol.	1
FP	53	Systém automaticky přidá dokončenému úkolu stav "dokončený".	1
FP	54	Systém bude umožňovat zobrazit nedokončené úkoly.	1
FP	55	Systém bude umožňovat zobrazení již dokončených úkolů.	2
FP	56	Systém dokončené úkoly nemaže, pouze jim nastaví stav "dokončený"	1
FP	57	Systém bude umožňovat filtraci v úkolech podle (přiřazeného uživatele, termínu dokončení, data vytvoření, názvu)	3
FP	58	Systém bude umožňovat editaci nastavení složky.	2
FP	59	Systém bude umožňovat přidat ikonku ke složce pro větší přehlednost	2
FP	60	Systém umožní přiřadit úkol pouze k jedné složce.	3
FP	61	Systém umožní každému úkolu přiřadit a upravit název úkolu	1
FP	62	Systém umožní každému úkolu přiřadit a upravit popis úkolu	1
FP	63	Systém umožní každému úkolu přiřazení zodpovědné osoby	1
FP	64	Systém umožní každému úkolu nastavení priority úkolu	2
FP	65	Systém umožní každému úkolu nahrání souborů	2
FP	66	Systém umožní každému úkolu přidání komentáře	3
FP	67	Systém umožní každému úkolu přiřadit informaci, zda se čeká na vyjádření od klienta	3
		Přístupy	
FP	68	Systém bude umožňovat správu přístupů	1
FP	69	Uživatel s dostatečným oprávněním si bude moci zobrazit přístupové údaje k daným stránkám	1
FP	70	Systém bude umožňovat správu nových a úpravu již existující přístupů	1
FP	71	Uživatel bude mít možnost zobrazit si všechny stránky ke kterým má dostatečné oprávnění	1
FP	72	Systém bude mít přednastavená pole po přihlašovací údaje (FTP, databáze, přístup do administrace)	1

		Funkční požadavek	Priorita
FP	73	Systém bude mít možnost přidat neomezené množství přístupů k jedné webové stránce bez přednastavených polí	2
FP	74	Systém bude umožňovat nastavit, zda se jedná o testovací webové prostředí nebo produkční	1
FP	75	Systém bude automaticky kontrolovat stav webových stránek. (Jejich dostupnost, nastavení meta tagů, rychlost odezvy serveru)	3
FP	76	Uživatel bude mít možnost vyvolat manuálně kontrolu stavu webových stránek. (Jejich dostupnost, nastavení meta tagů, rychlost odezvy serveru).	3
FP	77	Uživatel bude mít možnost nahrát soubory patřící k webové stránce	2
FP	78	Systém bude umožňovat vyplnit údaje patřící k webové stránce (hostingu, klient, cena za hosting v případě hostování u naší společnosti, datum konce smlouvy hostingu)	1
FP	79	Systém bude umožňovat nastavit, kteří uživatelé mají oprávnění pro zobrazení informací o webových stránkách	2
FP	80	Systém bude umožňovat správu webových stránek	1
FP	81	Systém bude umožňovat správu hostingů	1
FP	82	Systém bude umožňovat správu nových a úpravu již existující hostingů	1
FP	83	Uživatel bude mít možnost zobrazit si všechny hostingy ke kterým má dostatečné oprávnění	1
FP	84	Systém bude napojen na verzovací systém Bitbucket a pro každý projektu půjde nastavit zobrazení aktuálních změn v repositáři	3
FP	85	Systém bude mít možnost přidávat štítky (tagy) k úkolům ve správě úkolů	3
FP	86	Systém bude obsahovat následující uživatelské role (Host, Externista, Pracovník, Projektový manager, Administrátor)	1
FP	87	Systém bude obsahovat roli klient	2

Tabulka 6: Tabulka funkčních požadavků

7.3 Nefunkční požadavky

		Nefunkční požadavek	Priorita
NP	1	Systém bude dostupný 24/7	1
NP	2	Systém bude jednoduchý a přehledný pro uživatele	1
NP	3	Systém bude zabezpečen proti běžným útokům	1
NP	4	Systém bude možné spustit v posledních verzích webových prohlížečů	1
NP	5	Systém bude implementován dle standardů HTML5 a CSS	2
NP	6	Systém bude rozšiřitelný	1

Tabulka 7: Tabulka nefunkčních požadavků systému

7.4 Případy užití

V kapitole 7.1 jsme definovali aktéry vyskytující se v aplikaci, samotné případy užití je možné nalézt v příloze A.

7.4.1 Základní pojmy

Projekt

Projektem se v interním systému rozumí jedna zakázka a správa všech informací k ní patřících. Všechny časy se budou zapisovat k danému projektu.

Složka

Složka bude sloužit k organizaci úkolů do skupin. Každá složka bude mít přiřazený projekt.

Úkol

Úkol je jeden řešený problém. Úkol bude mít atributy stanovené v kapitole 4.3.2. Každý úkol je přiřazen do složky projektu nebo složky inbox.

Připomínka

Připomínka je způsob, kterým si mohou zaměstnanci předávat informace. Nemají přímou návaznost na projekt. Slouží především k internímu předávání informací, například o zajímavých technologiích souvisejících s oborem.

Hosting

Hosting je služba spravující server, na kterém jsou uloženy soubory k webovým prezentacím a aplikacím.

Uživatel

Aktér vykonávající popisovanou činnost.

8 Ekonomicko-manažerské zhodnocení

8.1 Náklady na implementaci systému

Hlavní část nákladů spojených s tvorbou výsledné aplikace tvoří ohodnocení vývojářů. Na pokrytí nákladů spojených s vývojem informačního systému počítáme s průměrnou hodinovou sazbou dle Tabulka 8. Hodinové sazby jsou brány dle hrubého odhadu a mohou se lišit.

Pozice	Cena/h
Grafické práce	1000
Programátorské práce	1000
Projektové řízení	1000
Testování	800

Tabulka 8: Průměrná hodinová sazba vývojářů

Odhadovaná pracnost je zobrazena v Tabulka 9, všechny časy jsou odhadovány dle zkušenosti z dřívějších projektů, reálné časy se nakonec mohou lišit.

Název	Odhadovaná pracnost v hodinách	Cena v Kč
Návrh a analýza	40	40 000
Návrh aplikace	60	60 000
Implementace	160	160 000
Testování	30	24 000
Projektové řízení	50	50 000
Celkem	320	334 000

Tabulka 9: Odhad pracnosti vývoje aplikace

Celkové očekávané náklady spojené s implementací vlastního systému dle specifikace v této práci jsou 334 000 Kč.

Společnost měla za poslední rok obrat okolo 4 000 000 Kč, budeme počítat, že 12,5 % z obrazu je čistý zisk po zdanění, tedy 500 000 Kč. Očekáváme, že díky změně procesů zefektivníme práci a zvýšíme tím čistý zisk alespoň o 15 %.

Částka 334 000 Kč je odhadem, kolik by stálo vytvoření aplikace u externí firmy. Pokud bychom počítali s 15% navýšení zisku každý rok, což je 75 000 Kč, při diskontní sazbě 7 %, vyšla by čistá současná hodnota po 5 letech záporně, tedy by se nám projekt nevyplatil.

Reálné náklady na samotný vývoj budou mnohem nižší. Vývoj aplikace probíhal z velké většiny mimo pracovní dobu, jako součást této diplomové práce nebo jako volnočasová aktivita, kdy práce na tomto projektu nebyla finančně ohodnocena.

Zbytek prací byl prováděn v čase mezi projekty, kdy některý z vývojářů neměl přidělenu práci. Vzhledem k faktu, že společnost musí platit zaměstnance i případě, že zrovna nepracují na projektu, přišlo společnosti ideální řešení využít volné kapacity k užitečné činnosti, která jim umožní se rozvíjet. Můžeme však počítat, že společnost platila odhadem 30 % odpracovaných hodin. Nejednalo se však o náklady v plné hodnotě dle tabulky Tabulka 8, ale část odpovídající hodinové sazbě zaměstnance. Pokud budeme počítat 100 hodin po 300 Kč, dostaneme cenu odpovídající 30 000 Kč. To za předpokladu, že nebudeme počítat ušlý zisk, protože zaměstnanec nepracoval na placeném projektu.

Pokud budeme počítat čistou současnou, při zvýšení zisku o 15 %, tak se investice vrátí již první rok. Projekt je tedy pro nás společnost výhodný.

Aktuálně má společnost 12 stálých zaměstnanců a plánuje nábor dalších. Do konce roku 2017 se očekává stav okolo 20 zaměstnanců, v roce 2018 okolo 30-35. Pro další roky nemáme stanovený odhad, budeme tedy počítat se stejným tempem růstu počtu zaměstnanců. Tyto čísla jsou pouze orientační, avšak jedná se plán, který by společnost chtěla dodržet.

8.1.1 Porovnání s existujícím řešením

V případě, že by společnost nevyvíjela vlastní systém, nejspíše by využila aplikace JIRA od společnosti Atlassian, která se zdá, že splňuje nejvíce požadavků na informační systém. Kdy při aktuálním počtu zaměstnanců by společnost platila 75 dolarů měsíčně za maximálně 15 uživatelů. Můžeme tedy říci, že pokud by společnost rozšiřovala své řady dle výše popsaného systému, bude ročně platit paušální poplatek za licenci software JIRA popsaný v tabulce Tabulka 10.

Rok	Počet zaměstnanců	Cena v USD
1.	12	750
2.	20	1500
3.	30	3000
4.	40	3000
5.	50	3000

Tabulka 10: Odhadované roční ceny software JIRA

Za pět let by společnost zaplatila 11 250 dolarů, což je v přepočtu na aktuální měnu 286 850 Kč, což je částka podstatně nižší, než je odhadovaná cena vytvoření vlastní aplikace pomocí externího zdroje a zároveň přinášející hotové řešení, kdy by se společnost nemusela zabývat vývojem dalších funkcí, které následující vývoj učiní ještě dražší. Na druhou stranu, JIRA nesplňuje všechny funkční požadavky, které si společnost stanovila.

8.2 Náklady na provoz a hardware

Ve výsledných nákladech nejsou zahrnuty ceny z hardware, společnost pro tento účel bude využívat vlastní server, který aktuálně využívá pro vývojové prostředí. Tento server má teoreticky neomezenou kapacitu místa a výkonu, výsledná aplikace tento server nebude výrazně zatěžovat. Cena pronájmu serveru se také nezmění, protože společnost platí paušální poplatek za celý server i bez používání vlastní aplikace.

8.3 Přínosy vytvořeného řešení

Vytvořená aplikace není cílena k prodeji, ani pronájmu licence. Výsledné řešení je určeno pouze pro interní využití, prozatím jako prototyp.

8.3.1 Testování nových technologií

Projekt interního systému přinesl vývojářům zajímavou výzvu, na které mohou testovat nové technologie, které by jinak museli zkoušet přímo na zákaznících. Zaměstnanci si tak mohou rozšiřovat své know-how, které následně mohou využít na projektech a tím přidat zakázce přidanou hodnotu.

8.3.2 Úspora a přehled časů

Díky informačnímu systému máme všechna data centralizovaná a všichni zaměstnanci k nim mají přístup. Společnost získala podrobný přehled o stavu projektů a úkolů. Vybrané funkce umožňují lépe organizovat kapacity a zrychlují, v některých případech úplně odstraňují komunikaci.

Největší časová úspora je u fakturace, delegování úkolů a předávání přístupových údajů. Zajímavé je také zvýšení počtu zapsaných hodin. Oproti původnímu stavu se jedná o navýšení v průměru o téměř 30 minut denně u každého zaměstnance, ne však všechen čas je možné fakturovat. Při aktuálním počtu 12 zaměstnanců znamená téměř 120 hodin měsíčně.

Odpozorovali nejen na ostatních zaměstnancích, ale také na sobě, že pokud na konci dne zaměstnanec nemá odpracováno alespoň 6 fakturovatelných hodin, snaží se zjistit, kde jim chybí zapsané hodiny. Dle jejich přístupových práv je následně možné si hodiny zapsat nebo se dohodnout na zapsání s některým z nadřízených.

Odhadujeme, že měsíčně ušetří vedoucí pracovník 2-3 pracovní dny, které dříve strávil nad fakturací. Nový proces je mnohem rychlejší a přehlednější, zároveň nevyžaduje spolupráci ostatních kolegů.

Z ekonomického hlediska je pro společnost nejcennější ušetřený a zároveň získaný čas. Díky změnám procesů, centralizaci úkolů a informacích o projektech zaměstnanci neztrácejí čas hledáním informací.

8.3.3 Finanční úspora

Zadavatel vytvořením vlastního řešení může ušetřit poplatky za služby třetích stran. V kapitole 8.1 jsou popsány ceny za vývoj aplikace při hodinových cenách, které simulují práci externího dodavatele, využití již existujícího řešení JIRA a implementace ve volném čase. Vzhledem ke způsobu, jakým je tato aplikace vyvíjena se společnosti s jistotou vyplatí.

8.3.4 Reference

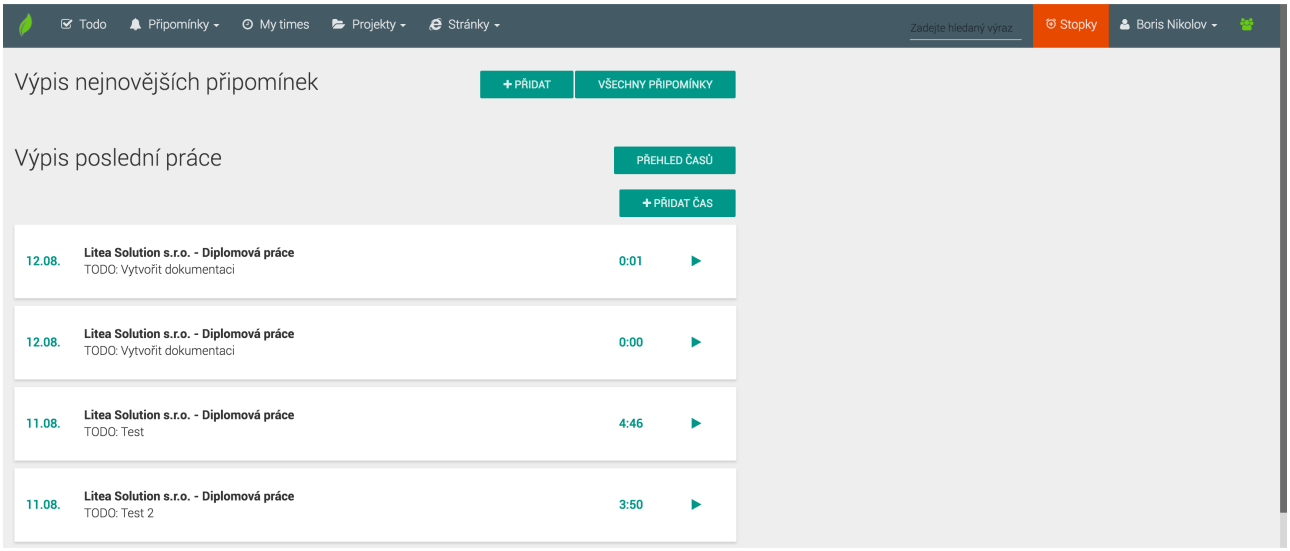
Společnost Litea Solution s.r.o. tvoří především webové prezentace, ale návrh a programování webových aplikací jí není cizí. Implementovaný systém bude zobrazován mezi díly, které společnost vytvořila a díky tomu společnosti přibude cenná reference.

Investice do vývoje vlastní aplikace se může zdát na první pohled vysoká, vlastní řešení však v sobě skrývá nemálo výhod, které se nedají jednoduše vyčíslit.

9 Náhled vytvořené aplikace

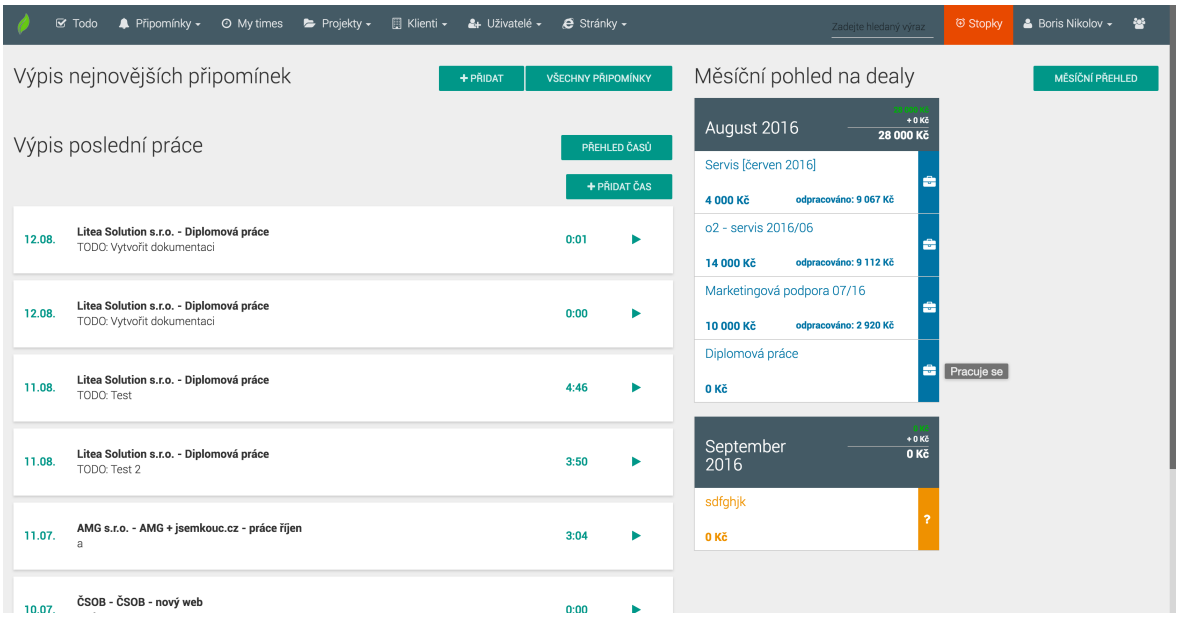
V následující kapitole jsou vybrány zajímavé části aplikace, které by mohli čtenáře zajímat. Zbytek snímků aplikace se nachází v příloze C.

Na Obrázek 19 je vidět úvodní obrazovka, na kterou se dostane zaměstnanec po přihlášení do systému. Na levé straně je zobrazen seznam posledních odpracovaných časů s popisem úkolu a názvem projektu. Pohyb mezi stránkami aplikace probíhá pomocí horního menu.



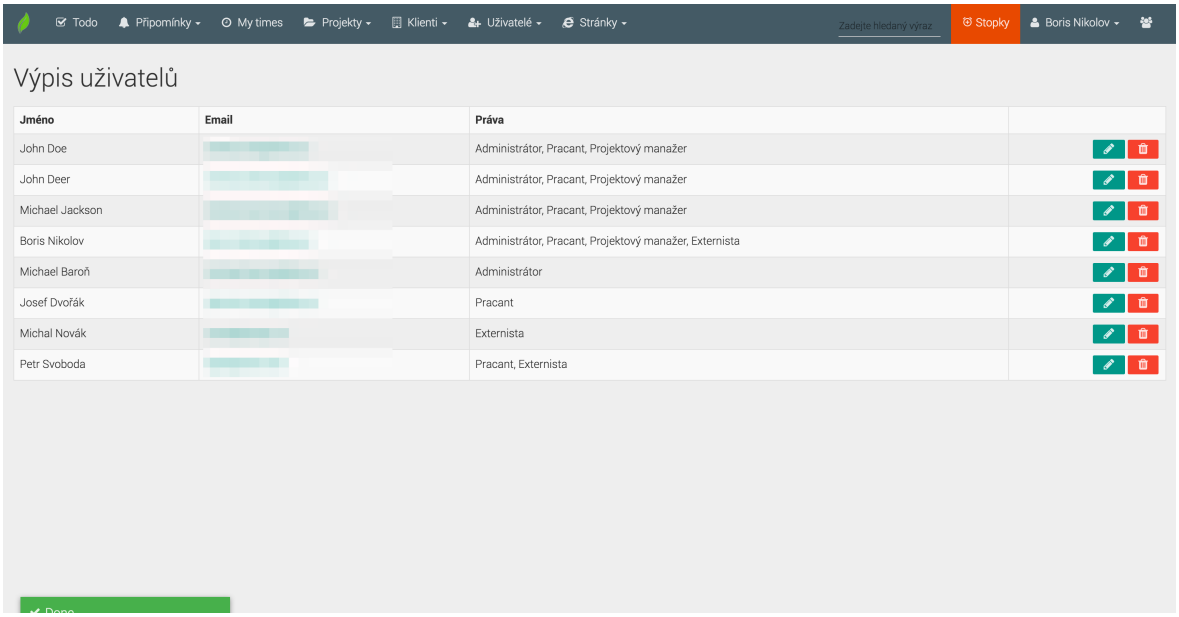
Obrázek 19: Úvodní obrazovka – role pracovník

Na Obrázek 20 je zobrazena úvodní obrazovka pro roli administrátor, které má na pravé straně navíc rozšíření zobrazující informace o aktuálním měsíci. V této části může administrátor vidět, které projekty se budou tento měsíc dokončovat a jaké je odhadovaná cena.



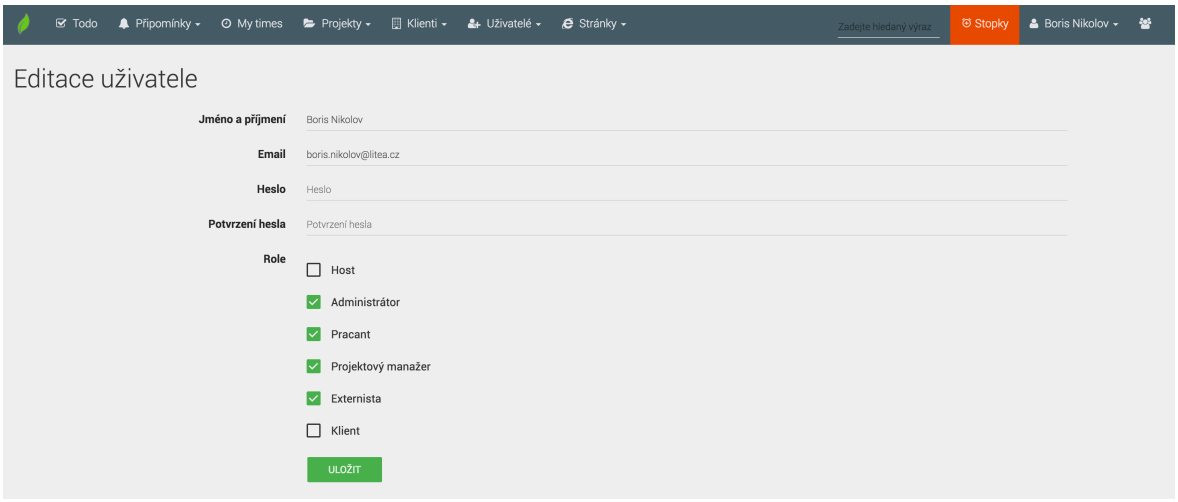
Obrázek 20: Úvodní obrazovka – role administrátor

Ve výpis uživatelů jsou vidět všechny důležité informace jako email a uživatelská práva. Po kliknutí na ikonku tužky je možné upravit detail uživatele, více zobrazeno na Obrázek 22. Kliknutím na ikonku koše bude uživatel smazán.



Obrázek 21: Výpis registrovaných uživatelů

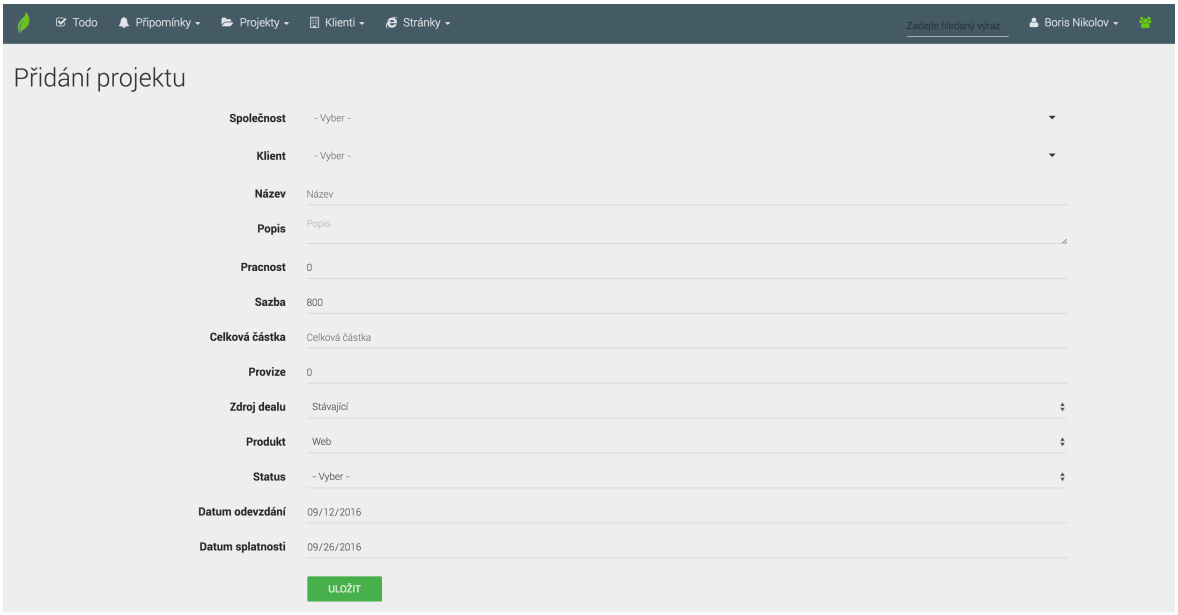
Při editaci uživatelů je možné změnit jejich role, které je možné kombinovat.



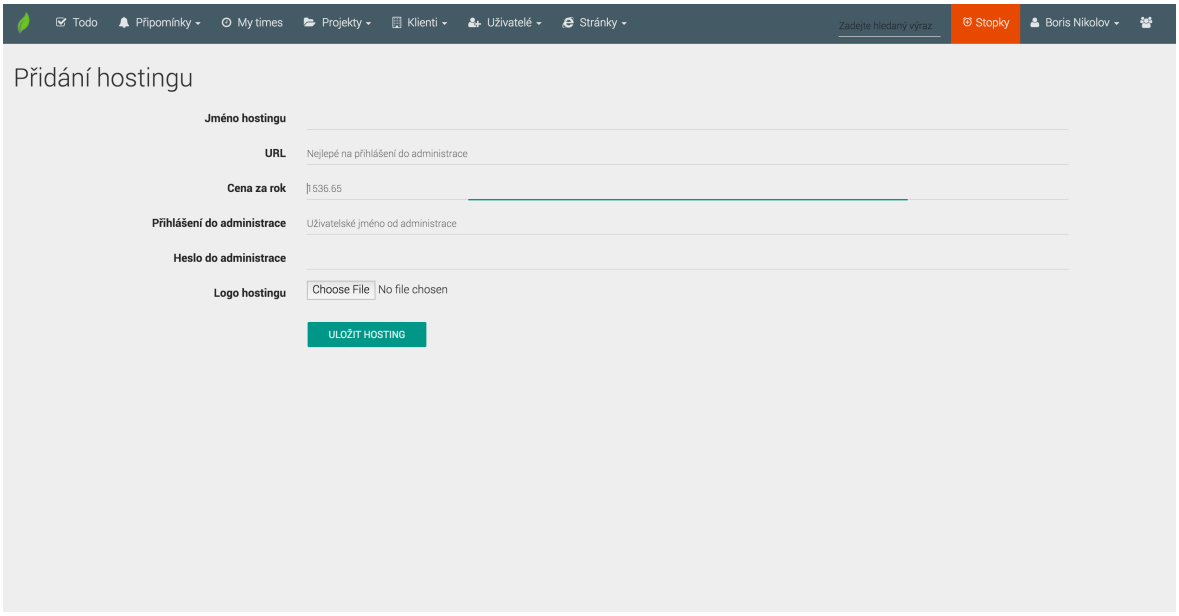
Obrázek 22: Obrazovka editace uživatelských informací

Obrázek 23: Obrazovka přidání nového klienta

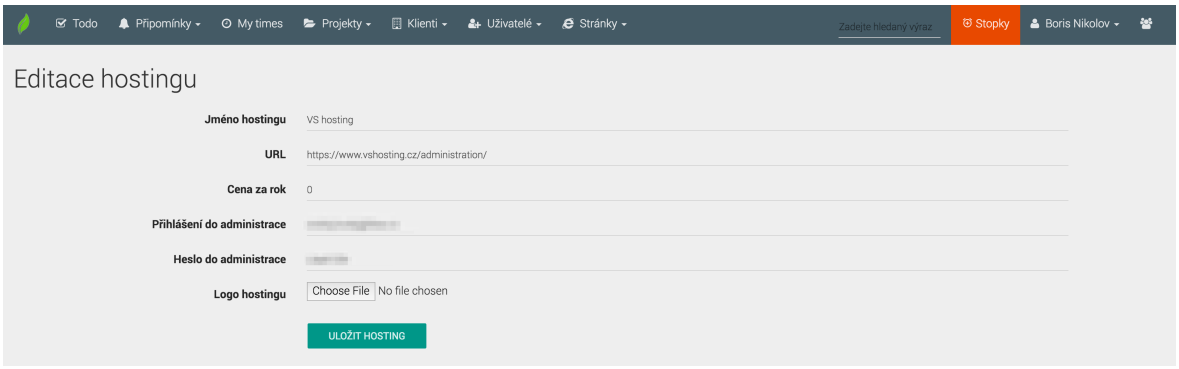
Přidání nového projektu probíhá pomocí formuláře na Obrázek 25, řádky, které u sebe mají ikonku šipky jsou rozbalovací menu, kde je možné vybrat z přednastavených možností.



Obrázek 24: Obrazovka přidání nového projektu

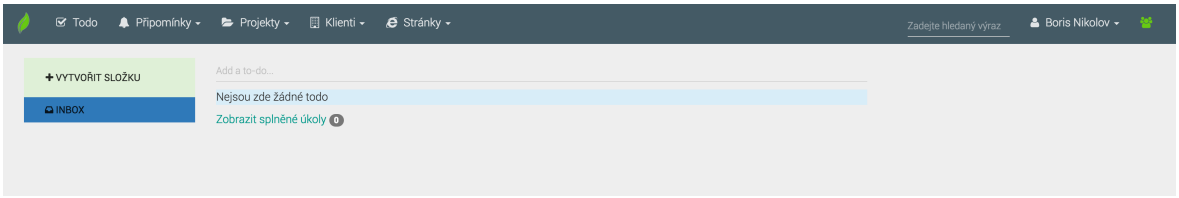


Obrázek 25: Obrazovka přidání nové hostingové společnosti

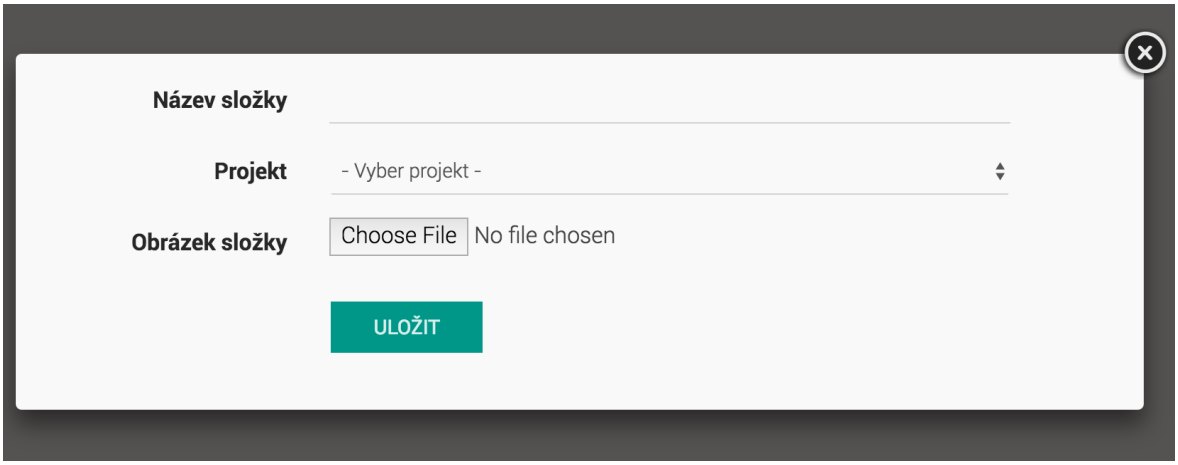


Obrázek 26: Obrazovka editace existující hostingové společnosti

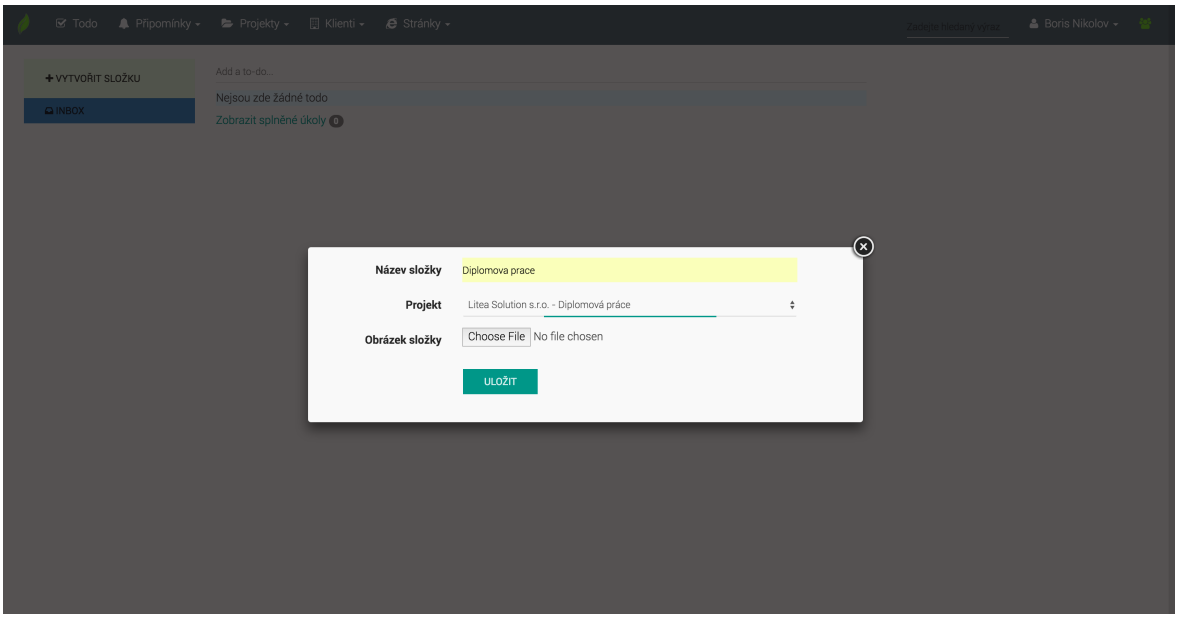
Na obrázku Obrázek 27 je vidět prázdná složka inbox, zobrazující informační hlášku. Na této obrazovce je možné vytvořit novou složku projektu pomocí tlačítka „vytvořit složku“, po stisknutí se objeví nabídka zobrazená níže na Obrázek 28. Na dalších obrázcích až do Obrázek 34 jsou zobrazeny možnosti úkolovníku. Úkoly je možné editovat kliknutím na jejich název. V tu chvíli se na pravé straně zobrazí nabídka, kde je možno upravit detail úkolu, přidat popisek nebo přiřadit čas dokončení a zodpovědnou osobu.



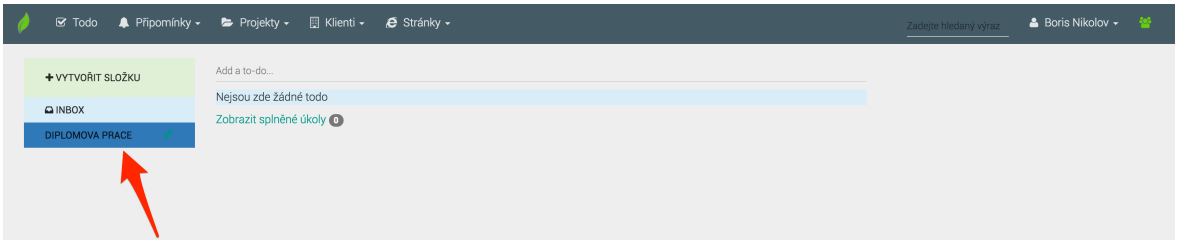
Obrázek 27: Obrazovka správy úkolů – složka inbox



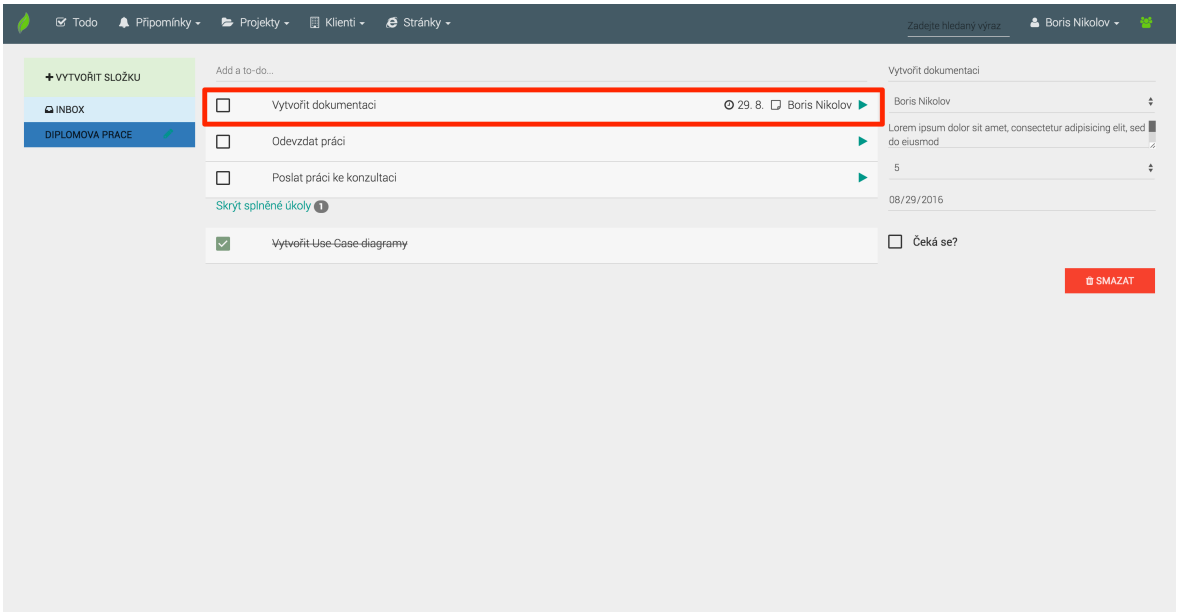
Obrázek 28: Výřez obrazovky zobrazující vytvoření nové složky projektu



Obrázek 29: Obrazovka vytvoření nové složky projektu



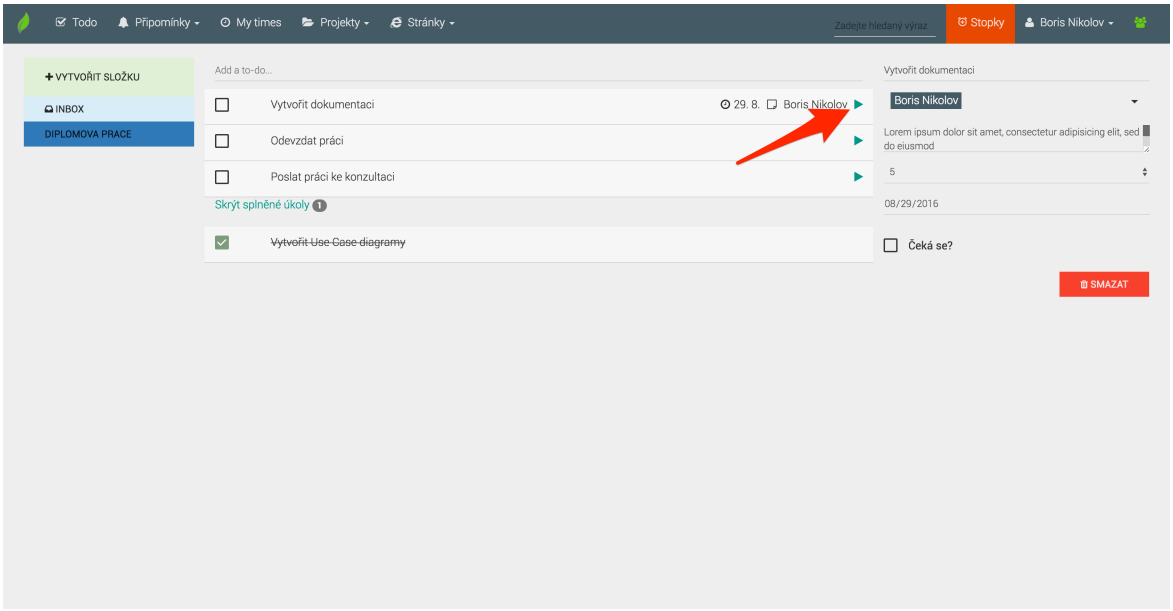
Obrázek 30: Obrazovka zobrazující vytvořené složky



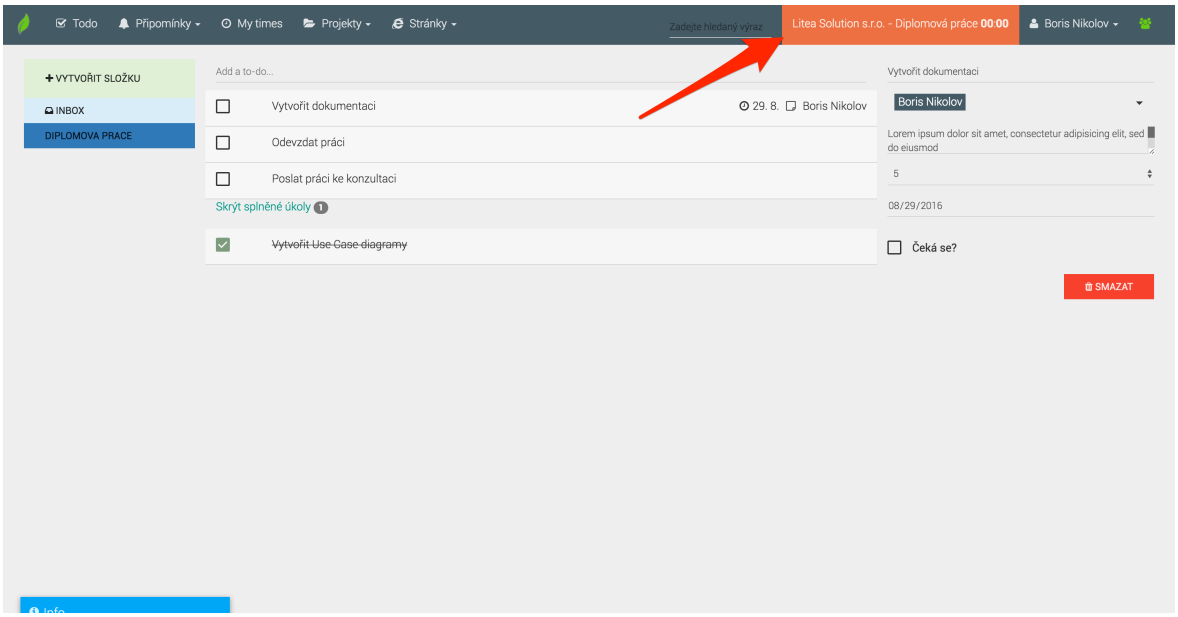
Obrázek 31: Zobrazení detailu úkolu

Obrázek 32: Editace vytvořeného úkolu

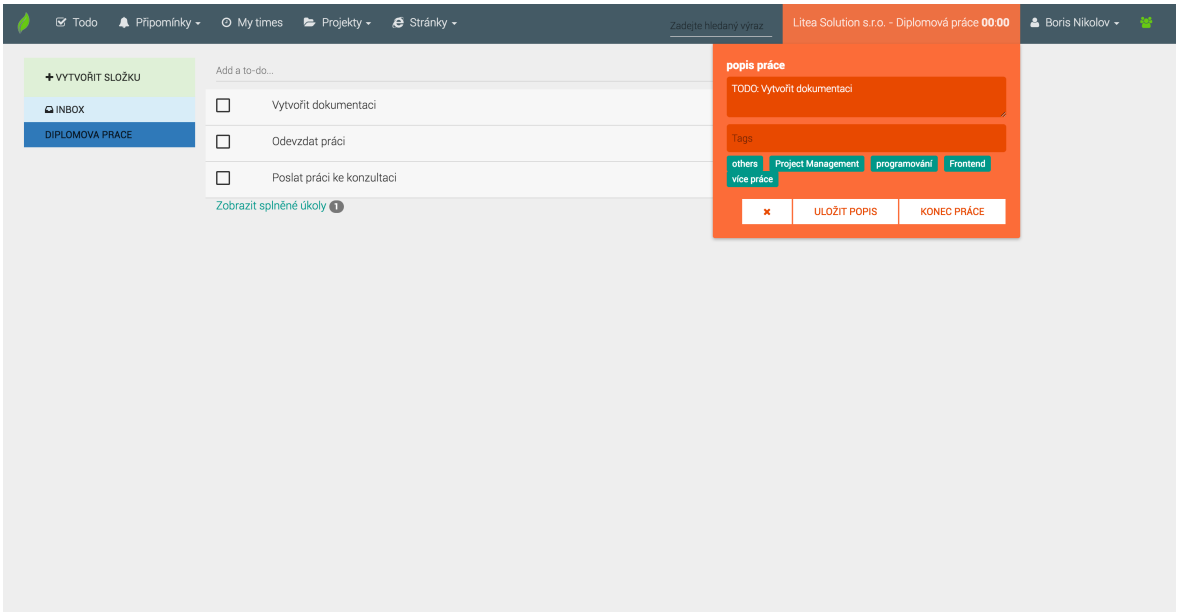
Pokud je úkol přiřazen do složky, je možné vyvolat funkci stopek stisknutím zeleného trojúhelníku na pravé straně od názvu úkolu. Tato možnost je zobrazena na Obrázek 33 až Obrázek 35. Systém automaticky doplní informace do popisku vykonávané práce dle popisku úkolu. Automaticky je také vybrán projekt a časy jsou k němu připisovány.



Obrázek 33: Měření času úkolu

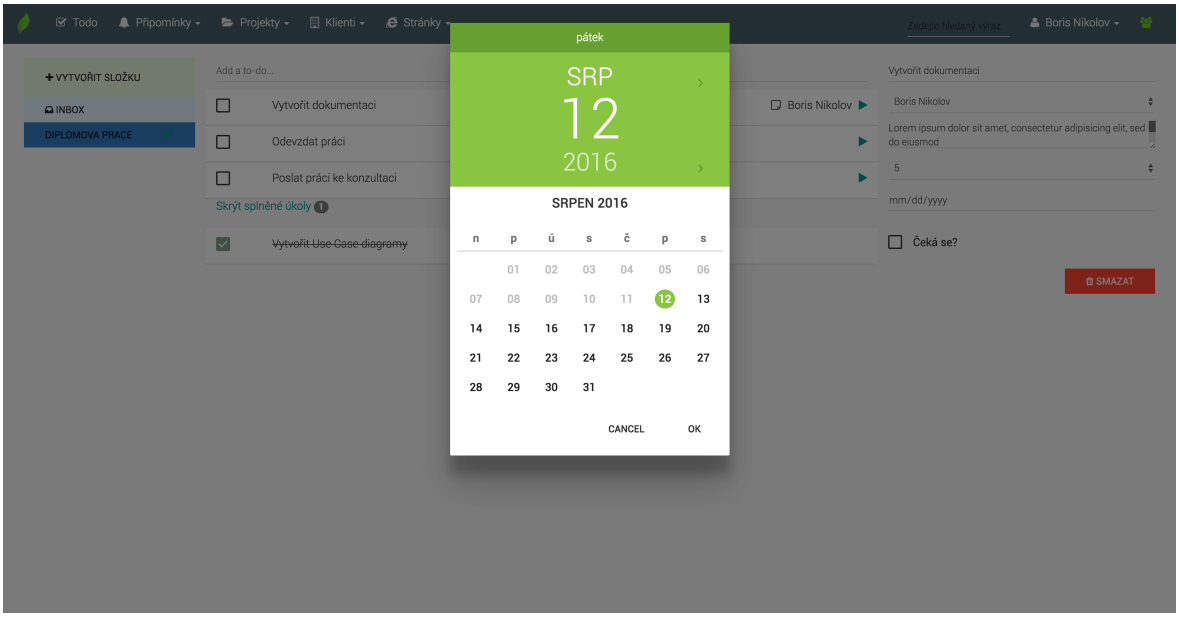


Obrázek 34: Měření vytvořeného úkolu



Obrázek 35: Ukončení či editace aktuálně měřeného úkolu

Pomocí editace úkolu je možné nastavit termín dokončení. Tento čas se vybírá pomocí zobrazeného kalendáře. Pro výběr dne dokončení stačí kliknout na jeho datum a na tlačítko ok ve spodní části kalendáře.

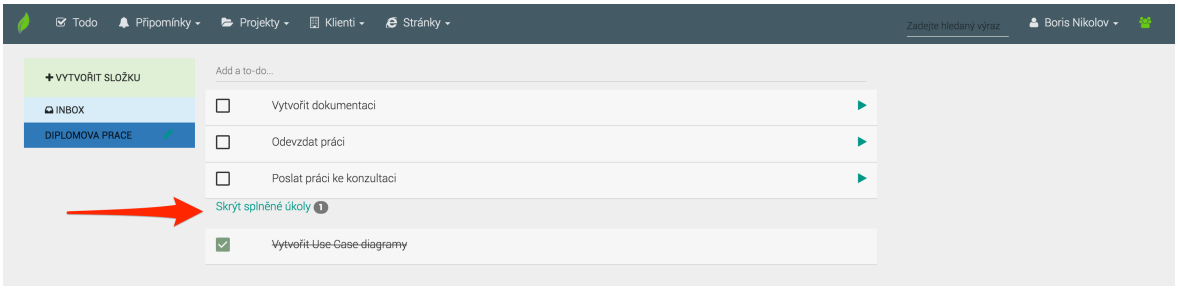


Obrázek 36: Výběr termínu dokončení úkolu

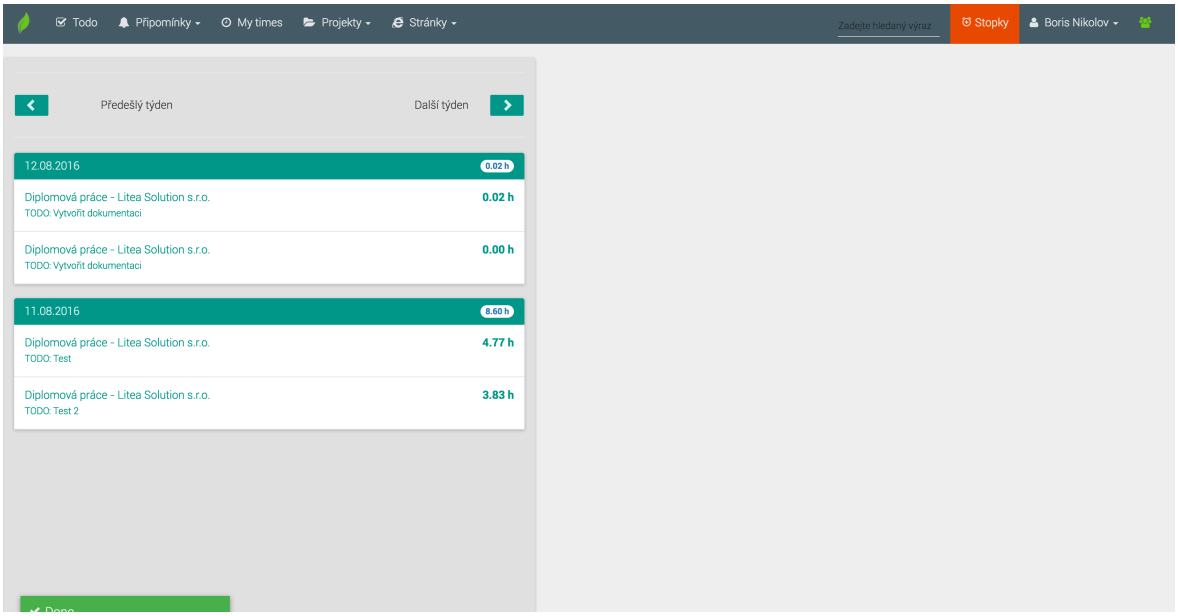
Úkol, který je již vyřešen je možné skrýt kliknutím na čtverec v jeho levém rohu, čímž se úkol označí jako hotový. Hotové úkoly se přesunou do skryté sekce, kterou je možno

zobrazit kliknutím na odkaz pod seznamem úkolů. Hotové úkoly je možné označit jako nedodělané, kliknutím na ikonku fajfky, čímž se úkol vrátí do horní části obrazovky.

Obrázek 37: Dokončení úkolu

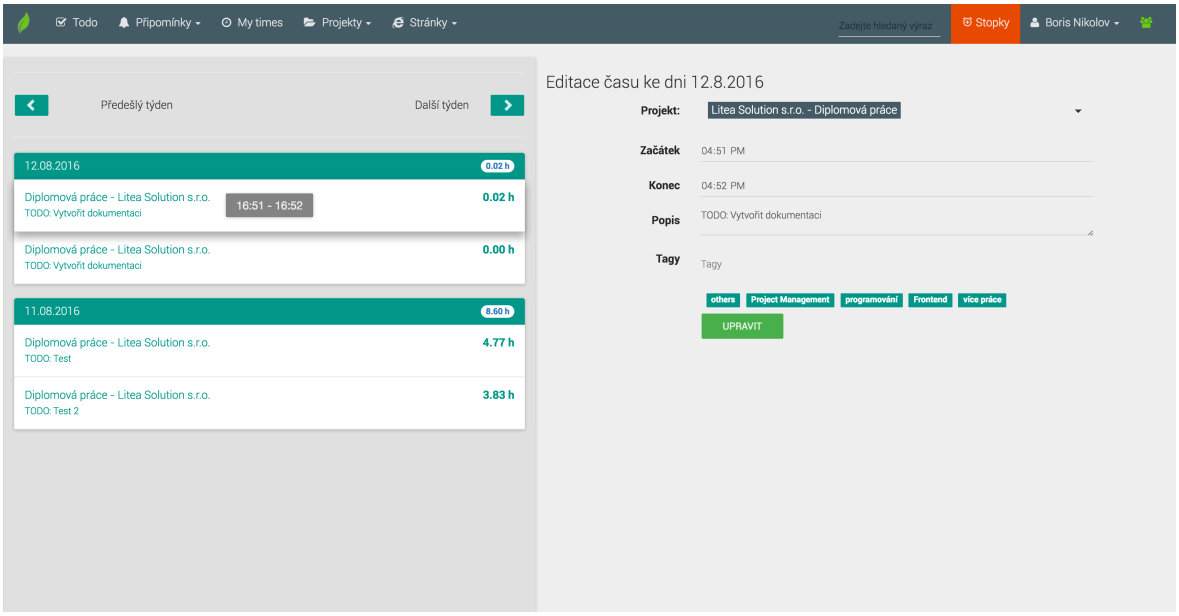


Obrázek 38: Zobrazení dokončených úkolů

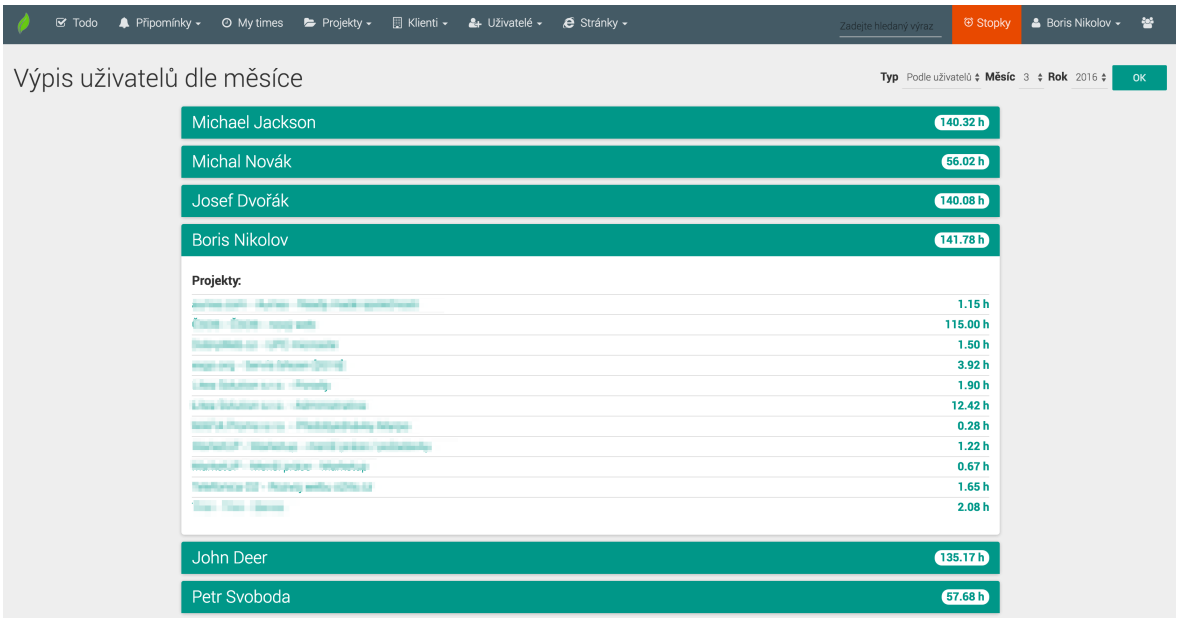


Obrázek 39: Obrazovka odpracovaných hodin za aktuální týden

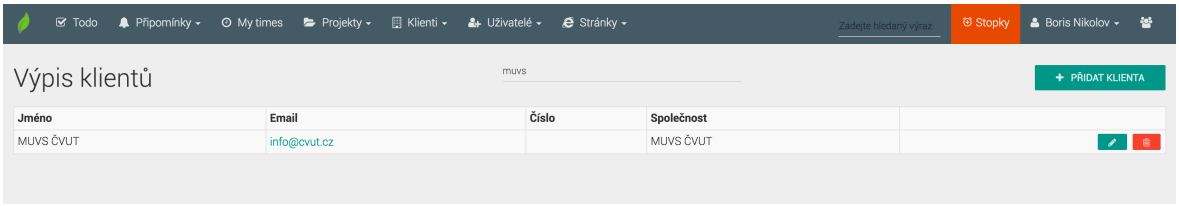
Na dalším obrázku je vidět editace odpracovaného času, kdy zaměstnanec může upravit začátek a konec času, doplnit štítky nebo upravit popis, co dělal.



Obrázek 40: Obrazovka odpracovaných hodin za aktuální týden – editace úkolu

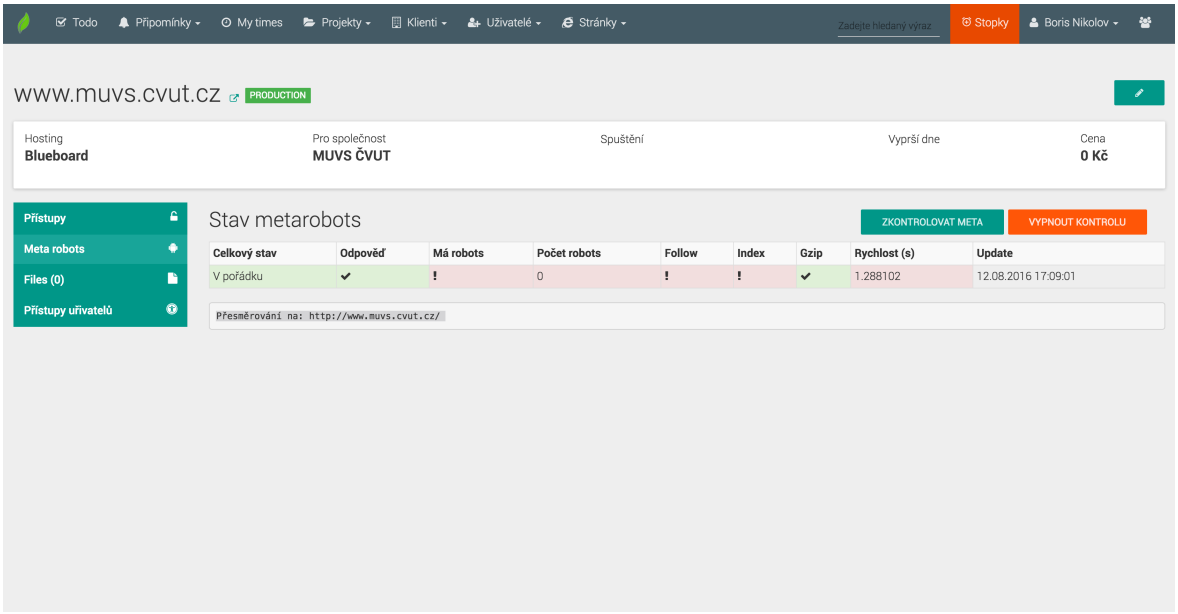


Obrázek 41:Výpis odpracovaných hodin uživatelů s vybraným uživatelem, zobrazeno dle měsíce

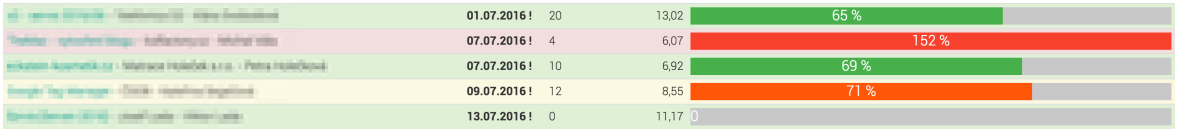


Obrázek 42: Výpis správy klientů

Zobrazení detailu webové stránky obsahuje informace získané pomocí scriptu Checker, který kontroluje její stav. Kontrolují se meta tagy, odezva serveru a nastavení kompresí webu.



Obrázek 43: Zobrazení detailu webové stránky



Obrázek 44: Zobrazení aktuálně strávených časů na projektu

10 Další vývoj

V průběhu vývoje a následného používání aplikace jsme dospěli k závěru, že některé části nejsou dostatečně uživatelsky přívětivé. Tyto nedostatky chceme v dalších verzích odstranit.

Za nedostatky považujeme uživatelsky nepřívětivý postup zakládání nových projektů pro společnosti nebo klienty, kteří ještě nemají vytvořené informace. Často se nám stává, že projektový manager chce založit nový projekt a ve chvíli kdy ho systém požádá o vyplnění společnosti zjistí, že společnost ještě neexistuje. V takovém případě je nutné vyplnit informace o společnosti a následně znovu vyplnit informace o novém projektu.

S vývojem nových technologií také přemýšlíme o přepsání některých částí aplikace do jazyka Javascript, přesněji využitím frameworku React. Za dobu psaní této práce se komunita okolo této technologie velmi posunula a změnila náš pohled. Myslíme si, že využití této technologie by některé části aplikace činilo rychlejší a možnost nativní aplikace pro správu úkolů nám přijde velmi zajímavá.

Dále se aplikaci chystáme rozšířit o další moduly, které nám usnadní práci. Mělo by se jednat o modul obsluhující faktury. Ten by měl být napojený na účetní program Pohoda, což by vedoucímu pracovníkovi usnadnilo proces fakturování. U každého klienta nebo společnosti by bylo možné vyplnit fakturační údaje. Z vybraných, strávených časů by se následně vypočítávala fakturovaná částka a bylo by možné přímo ze systému odesílat faktury klientům na email.

Chystáme se implementovat checklisty pro nasazování projektů na produkční prostředí. Stále se nám stává, že některý ze zaměstnanců nedodrží všechny body, které by měl kontrolovat při nahrávání webové aplikace nebo prezentace na produkční prostředí. Tomuto problému bychom rádi zamezili vyplňováním webového formuláře, který by část úkolů automaticky kontroloval. Zbytek by však byl stále na zaměstnanci, který by musel potvrdit, kontrolu všech úkolů. Tímto postupem bychom získali alespoň informaci o tom, kdo nasazoval daný projekt a zpětně se na něj mohli obrátit. Povinné vyplnění formuláře by mělo zaměstnance donutit body opravdu vykonat a zároveň mu sloužit jako osnova všech požadovaných úkolů.

Systém se chystáme více propojit se službami společnosti Google a Bitbucket. Od tohoto propojení očekáváme více automatických kontrol, které zamezí chybovosti. Mělo by se jednat o kontrolu kvality stránek pomocí služby PageSpeed Insight, kterou poskytuje společnost Google a je možné využít její API. U projektů chceme implementovat informaci o repositáři Git, kdy přímo v našem systému budou zobrazeny adresy pro stažení projektu.

11 Závěr

Cílem této práce bylo analyzovat, navrhnout a implementovat funkční prototyp aplikace pro správu projektů, úkolů a vnitropodnikových informací. Důraz byl kladen především na různé typy ukládaných informací, přístupových údajů pro webové stránky, hostingové služby a kontaktní informace o klientech. Zaměřili jsme se i na samotnou správu projektů, úkolů a odpracovaných časů, která nám u všech existujících řešení chyběla.

V práci bylo navrženo několik změn procesů. Jednalo se především o změny ve vývoji samotných aplikací, díky nimž zaměstnanci šetří čas a vyvíjejí bezpečněji. Odpadá též faktor lidských chyb při nasazování aplikace na produkční server. Díky navrženým změnám a vytvořené aplikaci, šetří management čas potřebný pro sběr dat nutný k fakturaci a získává dohled nad aktuálním stavem projektů.

11.1 Zhodnocení

Domníváme se, že hlavní cíl práce se nám povedlo splnit. Námi navržená aplikace splňuje požadavky sepsané v této práci. Z pohledu společnosti se prototyp aplikace funkcemi vyrovnává konkurenčním řešením a v některých ohledech je i překonává. Navíc máme kontrolu nad celým projektem a jeho rozšiřováním. Jsme si však vědomi, že velké projekty jsou za léta svého provozu mnohem vyladěnější a máme stále co dohánět.

Kompletní vývoj vlastního řešení přinesl spoustu poznatků. Využití PHP frameworku Nette se ukázalo jako dobré řešení, které urychlilo vývoj aplikace, především díky výbornému generování formulářů, kterých je v aplikaci opravdu mnoho.

Funkční testování nám však odhalilo, že některé kroky nejsou zcela logické a je potřeba posloupnosti některých formulářů poupravit. Jedná se například o vytváření nových projektů, kdy je potřeba mít již existujícího klienta a společnost.

Pro mě osobně byla tato práce výbornou zkušeností. Měl jsem příležitost vyzkoušet si analýzu, návrh i implementaci poměrně obsáhlé aplikace od úplného počátku, až po funkční prototyp. Během analýzy i návrhu jsem se seznámil s novými systémy i technologiemi, díky pomoci kolegů ze společnosti jsem se dostal i k samotnému programování samotné aplikace.

Společnost díky změnám procesů šetří čas, který museli zaměstnanci vykonávat na každodenních úkonech. Z ekonomického hlediska si myslíme, že vývoj aplikace vyšel vcelku levně, v porovnání s využíváním jiných systémů, které neobsahují námi požadované funkce. Společnost je především spokojená se zvýšenou produktivitou a průměrně vyšším odpracovaným, a především fakturovatelným hodinám.

Rád bych pokračoval ve vývoji tohoto softwaru, jelikož si myslím, že se jedná o unikátní projekt s velkým potenciálem.

Během vývoje práce jsme stihli implementovat všechny funkce označené nejvyšší a střední prioritou. Většina požadavků s nízkou prioritou je dokončena a čeká na otestování nebo byla nahrazena jiným řešením. Ve společnost plánujeme další rozšíření aktuální aplikace a již máme vymyšlené funkce, které by nám usnadnily každodenní procesy.

Seznam použité literatury

1. ALLEN, D. *Mít vše hotovo*. 2016. ISBN 978-80-7555-000-2.
2. Managementmania.com. *Projekt* [online]. 9. 9. 2015 [cit. 2016-06-20]. Dostupné z: https://managementmania.com/cs/projekt
3. managementmania.com. *Standardy a normy v managementu* [online]. 13. 3. 2016 [cit. 2016-08-03]. Dostupné z: https://managementmania.com/cs/standardy-a-normy-v-managementu
4. http://www.ipma.cz/wp-content/uploads/2014/10/narodni-standard-kompentenci-projektoveho-rizeni.pdf. *www.ipma.cz* [online]. 1. 10. 2014 [cit. 2016-08-13].
5. *http://ebrana.cz/* [online]. 8. 8. 2011 [cit. 2016-07-10]. Dostupné z: http://ebrana.cz/magazin/optimalni-zivotni-cyklus-webu-s-ohledem-na-koncoveho-uzivatele
6. DOLEŽAL JAN, P. M. A. B. L. *Projektový management podle IPMA. 2. aktualiz. a dopl. vyd.* Expert. Praha: Grada, 2012. ISBN 978-80-247-4275-5.
7. KOLEKTIV, K. B. A. Filozofická fakulta Univerzity Palackého v Olomouci. [Základy projektového řízení] In: *Základy projektového řízení* [online]. 1. 1. 2012 [cit. 2016-07-16]. Dostupné z: http://www.ff.upol.cz/fileadmin/user_upload/FF-katedry/psychologie/publikace/Bendova/Bendova_K_a_kol_zaklady_projektoveho_rizeni.pdf
8. ROSENAU, M. D. *Řízení projektů.* [Praxe manažera]. 2. Brno: Computer Press. ISBN 80-7226-218-1.
9. DRŽKA, M. eBrána.cz. *Optimální životní cyklus webu s ohledem na koncového uživatele* [online]. 8. 8. 2011 [cit. 2016-08-10]. Dostupné z: http://ebrana.cz/magazin/optimalni-zivotni-cyklus-webu-s-ohledem-na-koncoveho-uzivatele
10. Gantt.com. *What is a Gantt chart?* [online]. [cit. 2016-07-21]. Dostupné z: http://www.gantt.com/index.htm
11. MANAGEMENTMANIA.COM. Managementmania.com. *Vnitřní výnosové procento (IRR - Internal Rate of Return)* [online]. 16. 9. 2015 [cit. 2016-08-08]. Dostupné z: https://managementmania.com/cs/vnitrni-vynosove-procento
12. Středoeveropské centrum pro finance a management. *Return on Investment (ROI) Výnos investice* [online]. 2016 [cit. 2016-07-20]. Dostupné z: http://www.finance-management.cz/080vypisPojmu.php?X=Return+On+Investment+ROI&IdPojPass=4
13. UsabilityNet. *Methods: Affinity diagramming* [online]. [cit. 2016-07-27]. Dostupné z: http://www.usabilitynet.org/tools/affinity.htm
14. Svět produktivity. *Afinitní diagram* [online]. 2012 [cit. 2016-06-04]. Dostupné z: http://www.svetproduktivity.cz/slovnik/afinitni-diagram.htm

15. COOPER, A. R. R. A. D. C. *About face 3: the essentials of interaction design*. Rev. ed. Indianapolis: Wiley, c2007. ISBN 978-0-470-08411-3.

16. ITNETWORK.CZ. ITnetwork. *2. díl - UML - Use Case Diagram* [online]. 2014 [cit. 2016-05-29]. Dostupné z: http://www.itnetwork.cz/navrhove-vzory/uml/uml-use-case-diagram/

17. RADEK, O. *Objektové metody návrhu IS (přednáška)*. 2008.

18. LTD. A. S. S. I. Adobe. *Adobe Creative Cloud* [online]. [cit. 2016-07-02]. Dostupné z: http://www.adobe.com/cz/creativecloud.html

19. GRUDL, D. Nette.org. *Nette Framework: Rychlý a pohodlný vývoj webových aplikací v PHP* [online]. 2016 [cit. 2016-07-04]. Dostupné z: https://nette.org/cs/

20. WORDPRESS. Wordpress. *Blog Tool, Publishing Platform, and CMS — WordPress* [online]. [cit. 2016-07-02]. Dostupné z: https://wordpress.org/

21. PrestaShop. *PrestaShop - Free ecommerce software* [online]. 2016 [cit. 2016-07-06]. Dostupné z: https://www.prestashop.com/

22. Redmine. *Redmine: Overview* [online]. [cit. 2016-07-12]. Dostupné z: http:// www.redmine.org/

23. JetBrains. *PhpStorm IDE : JetBrains PhpStorm* [online]. 2016 [cit. 2016-07-15]. Dostupné z: https://www.jetbrains.com/phpstorm/

24. GOOGLE. Google Drive. *Google Drive* [online]. 2016 [cit. 2016-07-14]. Dostupné z: https://www.google.com/intl/cs_CZ/drive/

25. ALLEN, D. *Getting things done: the art of stress-free productivity*. New York, N.Y.: Penguin books, 2001. ISBN 06-708-9924-0.

26. Zdroják.cz. *Seriál: Pět důvodů, proč zvolit Git* [online]. 21. 12. 2009 [cit. 2016-06-08]. Dostupné z: https://www.zdrojak.cz/serialy/pet-duvodu-proc-zvolit-git/

27. Git. *Git* [online]. 2016 [cit. 2016-07-12]. Dostupné z: https://git-scm.com/

28. GRUDL, D. Phpfashion. *FTP Deployment: nahrávejte přes FTP chytře* [online]. 12. 6. 2012 [cit. 2016-06-12]. Dostupné z: https://phpfashion.com/ftp-deployment-nahravejte-pres-ftp-chytre

29. GitHub. *FTP Deployment: smart upload* [online]. GRUDL, D. ed. 10. 6. 2016 [cit. 2016-07-12]. Dostupné z: https://github.com/dg/ftp-deployment

30. Robotstxt. *tag - The Web Robots Pages* [online]. 2007 [cit. 2016-08-01]. Dostupné z: http://www.robotstxt.org/meta.html

31. Google Developers. *Make the Web Faster* [online]. 2016 [cit. 2016-07-15]. Dostupné z: https://developers.google.com/speed/pagespeed/

32. Bugzilla. *Bugzilla* [online]. 16. 5. 2016 [cit. 2016-06-15]. Dostupné z: https://www.bugzilla.org/

33. B. *Bugzilla na operačním systému Windows - recenze a postřehy* [online]. 13. 4. 2013 [cit. 2016-06-02]. Dostupné z: http://www.blogovnik.cz/bugzilla-na-operacnim-systemu-windows-recenze-a-postrehy-201304131049.php

34. Creative Commons. *Attribution-ShareAlike 2.0 Generic* [online]. [cit. 2016-06-19]. Dostupné z: https://creativecommons.org/licenses/by-sa/2.0/

35. Wunderlist. *To-do list, Reminders, Errands - App of the Year!* [online]. 2016 [cit. 2016-05-24]. Dostupné z: https://www.wunderlist.com/

36. TODOIST. Todoist. *Todoist: To do list and task manager. Free, easy, online and mobile* [online]. 2016 [cit. 2016-06-01]. Dostupné z: https://www.todoist.com

37. Mít vše hotovo. *Dva měsíce života s Todoist* [online]. 18. 5. 2015 [cit. 2016-05-28]. Dostupné z: http://www.mitvsehotovo.cz/2015/05/dva-mesice-zivota-s-todoist/

38. Trello. *Trello* [online]. 2016 [cit. 2016-06-15]. Dostupné z: https://trello.com/

39. JIRA Software. *JIRA Software - Issue & Project Tracking for Software Teams | Atlassian* [online]. 2016 [cit. 2016-07-01]. Dostupné z: https://www.atlassian.com/software/jira

40. HELPNETSECURITY.COM. helpnetsecurity.com. *The security of the most popular programming languages* [online]. 15. 4. 2014 [cit. 2016-05-02]. Dostupné z: https://www.helpnetsecurity.com/2014/04/15/the-security-of-the-most-popular-programming-languages/

41. W3TECHS. W3techs. *Usage of server-side programming languages for websites* [online]. 16. 4. 2016 [cit. 2016-06-18]. Dostupné z: https://w3techs.com/technologies/overview/programming_language/all

42. BUILTWITH.COM. Builtwith.com. *Programming Language usage in Czech Republic* [online]. 2016 [cit. 2016-05-16]. Dostupné z: http://trends.builtwith.com/framework/programming-language/country/Czech-Republic

43. MILLARES, G. Stoneriverelearning.com. *Top 5 Programming Languages Used In Web Development* [online]. 17. 12. 2015 [cit. 2016-05-15]. Dostupné z: http://blog.stoneriverelearning.com/top-5-programming-languages-used-in-web-development/

44. DEACON, J. jdl.co.uk/. In: *Model-view-controller (mvc) architecture* [online]. 2009 [cit. 2016-05-20]. Dostupné z: http://www.jdl.co.uk/briefings/MVC.pdf

45. PONKRÁC, M. *PHP a MySQL: bez předchozích znalostí*.. Brno: Computer Press, 2007. ISBN 978-80-251-1758-3.

46. MROZEK, J. Zdroják.cz. *Začínáme s AngularJS* [online]. 30. 11. 2012 [cit. 2016-06-20]. Dostupné z: https://www.zdrojak.cz/clanky/zaciname-s-angularjs/

47. ARLOW, J. A. I. N. *UML 2 a unifikovaný proces vývoje aplikací: objektově orientovaná analýza a návrh prakticky*. 2. aktualiz. a dopl. vyd. Brno: Computer Press, 2007. ISBN 978-80-251-1503-9.

48. Dokumenty Google. *Dokumenty Google – zdarma vytvářejte a upravujte dokumenty online.* [online]. [cit. 2016-07-03]. Dostupné z: https://www.google.cz/intl/cs/docs/about/

49. KANISOVÁ, H.. M. M. *UML srozumitelně*. 2. aktualizované vydání. Brno: Computer Press, 2007. ISBN 80-251-1083-4.

Seznam obrázků

Seznam tabulek

Seznam případů užití

Seznam příloh

Příloha A: Případy užití

1 Případy užití

1.1 Host

Use Case 1: Přihlášení uživatele

Název: *Přihlášení uživatele*

Popis: Přihlášení uživatele do systému

Aktéři: *Host*

Vstupní podmínky: *Uživatel* není přihlášen do systému

Výstupní podmínky: *Uživatel* je přihlášen do systému

Základní scénář:

1. *Uživatel* s registrovanými přihlašovacími údaji přijde na stránku s přihlašovacím formulářem
2. *Systém* vyzve *uživatele* k zadání uživatelského emailu a hesla
3. *Systém* zadané údaje ověří
4. *Systém* přihlásí *uživatele*

Alternativní scénáře:

Uživatel v bodě 2 zadá nesprávné přihlašovací údaje

1. *Systém* informuje *uživatele* o nesprávných přihlašovacích údajích
2. *Systém* umožní zadat přihlašovací údaje znovu

Use Case 2: Odhlášení uživatele

Název: *Odhlášení uživatele*

Popis: Odhlášení *uživatele* ze *systému*

Aktéři: *Host*

Vstupní podmínky: *Uživatel* je přihlášen do *systému*

Výstupní podmínky: *Uživatel* je odhlášen ze *systému*

Základní scénář:

1. Přihlášený *uživatel* vyvolá akci ohlášení
2. *Systém uživatele* odhlásí

1.2 Externista

Aktér *externista* je ve většině případů zároveň i *zaměstnanec*, díky této roli může měřit časy. Role *externista* odebírá uživateli možnosti, které v aplikaci může provádět. Na rozdíl od *zaměstnance* vidí pouze přidělené projekty a přístupy.

1.3 Klient

Aktér *klient* má přístup pouze ke správě úkolů. Jeho oprávnění vychází z role *zaměstnanec*. Klient může vytvářet, upravovat a mazat úkoly. Díky tomu získává vše potřebné k práci se *systémem*.

1.4 Zaměstnanec

Use Case 3: Vyhledávání v systému

Název: *Vyhledávání v systému*

Popis: Zobrazení konkrétních informací v systému pomocí funkce vyhledávání

Aktéři: *Zaměstnanec*

Vstupní podmínky: *Uživatel* je přihlášen do *systému*

Výstupní podmínky: *Uživateli* jsou zobrazeny vyhledané informace

Základní scénář:

1. *Uživatel* do vyhledávacího pole zadá hledaný výraz
2. *Systém* zobrazí informace o všech projektech, stránkách a přístupech obsahujících vyhledávané slovo

Alternativní scénář:

S*ystém* nenalezne hledané informace, zobrazí informační hlášku o nenalezení výsledků.

Use Case 4: Vytvoření nového úkolu

Název: *Vytvoření nového úkolu*

Popis: Vytvoření nového úkolu v záložce Todo.

Aktéři: *Zaměstnanec*

Vstupní podmínky: *Uživatel* je přihlášen do *systému*, existuje vytvořený projekt

Výstupní podmínky: *Uživatel* vytvořil nový úkol

Základní scénář:

1. *Uživatel* vybere projekt ze seznamu projektů
2. *Uživatel* vybere možnost přidat nový úkol

3. *Systém* vyzve *uživatele* k vyplnění názvu úkolu
4. *Uživatel* vyplní název úkolu a potvrdí jeho vytvoření tlačítkem „Enter“
5. *Systém* vytvoří nový úkol a zobrazí ho v seznamu úkolů

Use Case 5: Zobrazení detailu úkolu

Název: *Zobrazení detailu úkolu*

Popis: Zobrazení detailu konkrétního úkolu v záložce Todo.

Aktéři: *Zaměstnanec*

Vstupní podmínky: *Uživatel* je přihlášen do *systému*, existující úkol zadaný v *systému*

Výstupní podmínky: *Uživateli* je zobrazen detail vytvořeného úkolu

Základní scénář:

1. *Uživatel* vybere položku ze seznamu úkolů
2. *Systém* zobrazí podrobné informace o zvoleném úkolu

Use Case 6: Editace úkolu

Název: *Editace úkolu*

Popis: Editace atributů vytvořeného úkolu v záložce Todo.

Aktéři: *Zaměstnanec*

Vstupní podmínky: *Uživatel* je přihlášen do *systému*, existuje vytvořený úkol

Výstupní podmínky: *Uživatel* upravil atributy úkolu

Základní scénář:

1. *Uživatel* vybere projekt ze seznamu projektů
2. *Uživatel* vybere úkol ze seznamu vytvořených úkolů kliknutím na jeho název
3. *Systém* zobrazí detail úkolu
4. *Uživatel* upraví atributy úkolu
5. *Systém* validuje správnost zadaných dat
   POKUD validace proběhla úspěšně

a) *Systém* upraví atributy úkolu a zobrazí aktualizovanou verzi v seznamu úkolů
   POKUD validace neproběhla úspěšně

b) *Systém* oznámí *uživateli* chybu v zadaných hodnotách a proces pokračuje na bod 5

Use Case 7: Vytvoření nového detailu stránek

Název: *Vytvoření nového detailu stránek*

Popis: Vytvoření nové instance stránky v záložce Stránky

Aktéři: *Zaměstnanec*

Vstupní podmínky: *Uživatel* je přihlášen do *systému*, existuje vytvořený záznam o společnosti, existuje vytvořený záznam o hostingu

Výstupní podmínky: *Uživatel* vytvořil nový detail stránky

Základní scénář:

1. *Uživatel* v menu vybere možnost vytvořit novou stránku pomocí tlačítka „nová stránka“
2. *Uživatel* vyplní všechny potřebné informace
a) *Uživatel* vyplní hostingovou společnost stránky
b) *Uživatel* vyplní společnost, ke které jsou stránky vázány
c) *Uživatel* vyplní URL adresu stránky
d) *Uživatel* vyplní informaci o typu vývojového prostředí – test nebo produkce
e) *Uživatel* doplní nepovinné údaje o stránce, pokud k nim má informace
3. *Uživatel* potvrdí zadané informace pomocí tlačítka „uložit“
4. *Systém* validuje vložené informace
5. *Systém* uloží nový záznam o stránce

Alternativní scénář:

V bodě 1 se nemusí jednat pouze o položku menu, ale také o tlačítko na výpisu stránek.

Pokud bude validace v bodě 4 neúspěšná, *systém* informuje *uživatele* o chybných parametrech a proces pokračuje znovu na bod č. 2 s rozdílem pouhého doplnění/opravení informací.

Use Case 8: Zobrazení detailu stránek

Název: *Zobrazení detailu stránek*

Popis: Zobrazení detailu konkrétní stránky v záložce Stránky.

Aktéři: *Zaměstnanec*

Vstupní podmínky: *Uživatel* je přihlášen do systému, existuje vytvořený záznam o stránce

Výstupní podmínky: *Uživateli* je zobrazen detail stránky

Základní scénář:

1. *Uživatel* vybere položku ze seznamu stránek
2. *Systém* zobrazí podrobné informace o stránce

Alternativní scénář:

Uživatel nemá přístupy k žádným stránkám a *systém* vypíše pouze informaci o nenalezení žádných záznamů. Týká se především aktéra *Externista.*

Use Case 9: Editace detailu stránek

Název: *Editace detailu stránek*

Popis: Editace detailu konkrétní stránky v záložce Stránky

Aktéři: *Zaměstnanec*

Vstupní podmínky: *Uživatel* je přihlášen do systému, existuje vytvořený záznam o stránce

Výstupní podmínky: *Uživatel* upravil detail stránky

Základní scénář:

1. *Uživatel* vybere položku ze seznamu stránek
2. *Systém* zobrazí podrobné informace o stránce

Alternativní scénář:

Uživatel nemá přístupy k žádným stránkám a *systém* vypíše pouze informaci o nenalezení žádných záznamů. Týká se především aktéra *Externista.*

Use Case 10: Zobrazení výpisu hostingových služeb

Název: *Zobrazení výpisu hostingových služeb*

Popis: Zobrazení výpisu hostingových služeb uložených v *systému*

Aktéři: *Zaměstnanec*

Vstupní podmínky: *Uživatel* je přihlášen do *systému*, existuje vytvořený záznam o hostingové službě

Výstupní podmínky: *Uživateli* je zobrazen seznam hostingových služeb

Základní scénář:

1. *Uživatel* vybere v menu položku Stránky → Hosting
2. *Systém* zobrazí seznam všech hostingových služeb

Alternativní scénář:

Uživatel s rolí *externista* nemá přístup k informacím o hostingu

---

Use Case 11: Zobrazení detailu hostingové služby

Název: *Zobrazení detailu hostingové služby*

Popis: Zobrazení detailu konkrétního hostingu v záložce Stránky → Hosting

Aktéři: *Zaměstnanec*

Vstupní podmínky:

Uživatel je přihlášen do *systému*, existuje vytvořený záznam hostingu

Výstupní podmínky: *Uživateli* je zobrazen detail hostingu

Základní scénář:

1. *Uživatel* vybere hosting ze seznamu zobrazených dle Use Case 10
2. *Systém* zobrazí informace o hostingu

Alternativní scénář:

Uživatel s rolí *externista* nemá přístup k informacím o hostingu

---

Use Case 12: Kontrola stavu webových stránek

Název: *Kontrola stavu webových stránek*

Popis: Vyvolání kontroly stavu stránek hlídajících nastavení meta tagů a rychlosti odpovědi serveru

Aktéři: *Zaměstnanec*

Vstupní podmínky: *Uživatel* je přihlášen do *systému*, existuje vytvořený záznam o stránce

Výstupní podmínky: *Uživatel* má přístup k aktualizovaným informacím o stavu webové stránky

Základní scénář:

1. *Uživatel* postupuje stejným způsobem jako u případu užití *Zobrazení detailu stránek*
2. *Uživatel* vybere možnost zkontrolovat meta data
3. *Systém* spustí kontrolu pro vybranou stránku
4. *Systém* vypíše aktualizované informace

Use Case 13: Zobrazení přístupových údajů k projektu

Název: *Zobrazení přístupových údajů k projektům*

Popis: Zobrazení přístupových údajů potřebných pro práci na projektech

Aktéři: *Zaměstnanec*

Vstupní podmínky: *Uživatel* je přihlášen do *systému*, existují vytvořené přístupy

Výstupní podmínky: *Uživateli* jsou zobrazeny přístupové údaje ke konkrétnímu projektu

Základní scénář:

3. *Uživatel* vybere v menu položku Stránky → Přístupy

4. *Systém* zobrazí seznam všech webových stránek
5. *Uživatel* vybere nebo vyfiltruje hledanou stránku a kliknutím se dostane na detail
6. *Systém* zobrazí přístupové údaje k vybrané stránce

Alternativní scénář:

Uživatelé s rolí *externista* uvidí pouze informace o projektech, ke kterým jsou přiřazeni

---

Use Case 14: Zobrazení výpisu projektů

Název: *Zobrazení výpisu projektů*

Popis: Zobrazení výpisu všech projektů

Aktéři: *Zaměstnanec*

Vstupní podmínky: *Uživatel* je přihlášen do systému, existuje vytvořený projekt

Výstupní podmínky: *Uživateli* je zobrazen seznam všech projektů

Základní scénář:

1. *Uživatel* vybere v menu položku Projekty → Výpis
2. *Systém* zobrazí seznam všech projektů

Alternativní scénář:

Uživatelé s rolí *externista* uvidí pouze informace o projektech, ke kterým jsou přiřazeni

---

Use Case 15: Zobrazení detailu projektu

Název: *Zobrazení detailu projektu*

Popis: Zobrazení detailních informací k vybranému projektu

Aktéři: *Zaměstnanec*

Vstupní podmínky: *Uživatel* je přihlášen do systému, existuje vytvořený projekt

Výstupní podmínky: *Uživateli* jsou zobrazeny informace o vybraném projektu

Základní scénář:

1. *Uživatel* postupuje stejným způsobem jako u případu užití *„Zobrazení výpisu projektů“*
2. *Uživatel* vybere nebo vyfiltruje hledaný projekt a kliknutím se dostane na jeho detail
3. *Systém* zobrazí podrobné informace k vybranému projektu

Alternativní scénář:

Uživatelé s rolí *externista* uvidí pouze informace o projektech, ke kterým jsou přiřazeni

---

Use Case 16: Vytvoření složky projektu

Název: *Vytvoření složky projektu*

Popis: Uživatel chce vytvořit novou složku pro přidávání úkolů

Aktéři: *Zaměstnanec*

Vstupní podmínky: *Uživatel* je přihlášen do systému

Výstupní podmínky: *Uživatel* vytvořil novou složku projektu

Základní scénář:

1. *Uživatel* vybere možnost vytvoření nové složky
2. *Systém uživatele* vyzve k zadání následujících údajů
   a) Název složky
   b) Výběr projektu ze seznamu existujících projektů
   c) Dobrovolný obrázek
3. *Uživatel* potvrdí vytvoření složky tlačítkem „uložit“
4. *Systém* vytvoří novou složku a zobrazí ji v seznamu

Use Case 17: Přidání nového úkolu do složky projektu

Název: *Přidání nového úkolu do složky projektu*

Popis: Uživatel chce přidat nový úkol do systému.

Aktéři: *Zaměstnanec*

Vstupní podmínky: *Uživatel* je přihlášen do *systému*, existuje vytvořená složka projektu

Výstupní podmínky: *Uživatel* vytvořil nový úkol ve vybrané složce

Základní scénář:

1. *Uživatel* vybere složku
2. *Systém uživatele* vyzve k zadání následujících údajů
   a) Název složky
   b) Výběr projektu ze seznamu existujících projektů
   c) Dobrovolný obrázek
3. *Uživatel* potvrdí vytvoření složky tlačítkem „uložit“
4. *Systém* vytvoří novou složku a zobrazí ji v seznamu

Use Case 18: Zobrazení seznamu úkolů

Název: *Zobrazení seznamu úkolů*

Popis: Zobrazení výpisu všech úkolů ve vybrané složce

Aktéři: *Zaměstnanec*

Vstupní podmínky: *Uživatel* je přihlášen do *systému*, existuje vytvořená složka

Výstupní podmínky: *Uživateli* je zobrazen seznam všech úkolů ve vybrané složce

Základní scénář:

1. *Uživatel* vybere složku na stránce Todo
2. *Systém* zobrazí seznam všech úkolů ve vybrané složce

Alternativní scénář:

Pokud v bodě 2 neexistují žádné nedokončené úkoly, *systém* upozorní *uživatele* pomocí informační hlášky.

Use Case 19: Zobrazení detailu úkolu

Název: *Zobrazení detailu úkolu*

Popis: Zobrazení detailních informací o vybraném úkolu

Aktéři: *Zaměstnanec*

Vstupní podmínky: *Uživatel* je přihlášen do *systému*, existuje vytvořený úkol

Výstupní podmínky: *Uživateli* jsou zobrazeny informace o vybraném úkolu

Základní scénář:

Uživatel se nachází v části zobrazení seznamu úkolů (viz **Název:** *Přidání nového úkolu do složky projektu*

Popis: Uživatel chce přidat nový úkol do systému.

Aktéři: *Zaměstnanec*

Vstupní podmínky: *Uživatel* je přihlášen do *systému*, existuje vytvořená složka projektu

Výstupní podmínky: *Uživatel* vytvořil nový úkol ve vybrané složce

Základní scénář:

5. *Uživatel* vybere složku
6. *Systém uživatele* vyzve k zadání následujících údajů
   a) Název složky
   b) Výběr projektu ze seznamu existujících projektů
   c) Dobrovolný obrázek
7. *Uživatel* potvrdí vytvoření složky tlačítkem „uložit“
8. *Systém* vytvoří novou složku a zobrazí ji v seznamu

---

1. Use Case 18)
2. *Uživatel* si přeje zobrazit nebo skrýt dokončené úkoly
3. *Uživatel* stiskne tlačítko určené pro zobrazení/skrytí úkolů
   a) *Systém* zobrazí dokončené úkoly v případě, že nebyly zobrazeny
   b) *Systém* skryje dokončené úkoly v případě, že byly zobrazeny

---

Use Case 20: Úprava detailu úkolu

Název: *Úprava detailu úkolu*

Popis: Upravení informací vybraného úkolu. Může se jednat o přiřazení zodpovědné osoby, přidání popisu, termínu dokončení nebo například nahrání nového souboru

Aktéři: *Zaměstnanec*

Vstupní podmínky: *Uživatel* je přihlášen do *systému*, existuje vytvořený úkol

Výstupní podmínky: *Uživatel* upravil nebo přidal nové informace k vybraném úkolu

Základní scénář:

1. *Uživatel* postupuje stejným způsobem jako u Use Case 19
2. *Uživatel* upraví nebo přidá nové informace k úkolu pomocí předem předpřipravených polí
3. *Systém* uloží aktualizované údaje a zobrazí je na výpisu úkolů

Use Case 21: Zobrazení/skrytí dokončených úkolů

Název: *Zobrazení/skrytí dokončených úkolů*

Popis: Systém umožní uživateli zobrazit nebo skrýt dokončené úkoly.

Aktéři: *Zaměstnanec*

Vstupní podmínky: *Uživatel* je přihlášen do *systému*, existuje dokončený úkol

Výstupní podmínky: *Uživateli* jsou zobrazeny/skryty úkoly se stavem *dokončen*

Základní scénář:

Uživatel v seznamu zobrazených úkolů (viz **Název:** *Přidání nového úkolu do složky projektu*

Popis: Uživatel chce přidat nový úkol do systému.

Aktéři: *Zaměstnanec*

Vstupní podmínky: *Uživatel* je přihlášen do *systému*, existuje vytvořená složka projektu

Výstupní podmínky: *Uživatel* vytvořil nový úkol ve vybrané složce

Základní scénář:

9. *Uživatel* vybere složku
10. *Systém uživatele* vyzve k zadání následujících údajů
    a) Název složky
    b) Výběr projektu ze seznamu existujících projektů
    c) Dobrovolný obrázek

11. *Uživatel* potvrdí vytvoření složky tlačítkem „uložit“
12. *Systém* vytvoří novou složku a zobrazí ji v seznamu

---

1. Use Case 18) vybere úkol
2. *Systém* zobrazí podrobné informace k vybranému úkolu

Use Case 22: Měření času

Název: *Měření času*

Popis: Uživatel chce měřit čas k vykonávané aktivitě. Čas bude zaznamenán pomocí stopek, společně s časovým údajem bude uložena také informace o projektu, popis úkolu a štítky odpovídající vykonávané aktivitě.

Aktéři: *Zaměstnanec*

Vstupní podmínky: *Uživatel* je přihlášen do systému

Výstupní podmínky: *Systém* zaznamenal informace o odpracované aktivitě

Základní scénář:

1. *Uživatel* vybere možnost stopování času
2. *Systém* vyzve uživatele k vybrání projektu, na kterém se bude pracovat
3. *Uživatel* vybere projekt ze seznamu
   a) **Pokud** *uživatel* chce začít zaznamenávat čas, započne měření pomocí tlačítka
   b) **Pokud** *uživatel* chce doplnit informace – popis a štítky je možné je vyplnit v tomto bodě
   c) **Pokud** *uživatel* chce započít práce v jiný než aktuální čas, musí upravit hodnoty počátku práce pomocí příslušného vstupu
4. *Uživatel* vykonává pracovní aktivitu
5. *Uživatel* si přeje skončit práce, učiní tak pomocí příslušného tlačítka označeného „konec práce“
6. *Systém* zaznamená čas strávený na projektu a všechny vyplněné hodnoty

---

Use Case 23: Měření času vytvořeného úkolu

Název: *Měření času*

Popis: Uživatel chce měřit čas k vybranému úkolu zadanému ve složce. Systém automaticky vyplní informace o úkolu dle zadaných informací. Zaměstnanec tedy nemusí vyplňovat projekt ani popis. Začátek práce je defaultně v daný okamžik, lze ho však upravit.

Aktéři: *Zaměstnanec*

Vstupní podmínky: *Uživatel* je přihlášen do *systému*, existuje vytvořený úkol, *uživatel* neměří aktivitu pomocí integrovaných stopek

Výstupní podmínky: *Systém* zaznamenal informace o odpracované aktivitě

Základní scénář:

1. *Uživatel* vybírá ze seznamu úkolů (viz Use Case 18)
2. *Uživatel* využije tlačítka u úkolu značícího počátek práce (symbol ▸)
3. *Systém* vloží začátek prací do stopek a vyplní informace získané z detailu úkolu
4. *Proces* pokračuje stejně jako u Use Case 22 bodem 4

Use Case 24: Zobrazení vlastních časů

Název: *Zobrazení vlastních časů*

Popis: Zobrazení podrobných informací o časech strávených zaměstnancem

Aktéři: *Zaměstnanec*

Vstupní podmínky: *Uživatel* je přihlášen do *systému*

Výstupní podmínky: *Uživateli* jsou zobrazeny informace o jeho odpracovaných hodinách s možností filtrace

Základní scénář:

1. *Uživatel* vybere v menu položku Projekty → Výpis strávených časů
2. *Systém* zobrazí podrobné informace o všech časech strávených uživatelem na projektech za aktuální měsíc
3. *Systém* umožní řadit zobrazená data dle projektů nebo stráveného času
4. *Systém* umožní prohlížet i historická data pomocí přepínání měsíců

Use Case 25: Zobrazení přehledů časů projektů

Název: *Zobrazení přehledů časů projektů*

Popis: Zobrazení podrobných informací o časech strávených na projektech

Aktéři: *Zaměstnanec*

Vstupní podmínky: *Uživatel* je přihlášen do *systému*

Výstupní podmínky: *Uživateli* jsou zobrazeny informace o aktuálním stavu projektů, odpracovaných hodinách, zbývajícím čase a termínu dokončení

Základní scénář:

1. *Uživatel* vybere v menu položku Projekty → Přehled časů
2. *Systém* zobrazí podrobné informace o všech projektech

3. *Systém* umožní rychle vyhledávat v zobrazených datech pomocí vyhledávacího vstupního elementu

Alternativní scénář:

Uživatelé s rolí *externista* uvidí pouze informace o projektech, ke kterým jsou přiřazeni

---

Use Case 26: Zobrazení připomínek

Název: *Zobrazení připomínek*

Popis: Zobrazení seznam připomínek uživatele, vlastních nebo jemu přiřazených

Aktéři: *Zaměstnanec*

Vstupní podmínky: *Uživatel* je přihlášen do *systému*

Výstupní podmínky: *Uživateli* jsou zobrazeny připomínky, které si vytvořil on, nebo mu byly přiřazeny.

Základní scénář:

1. *Uživatel* vybere v menu položku Připomínky → Výpis
2. *Systém* zobrazí všechny *uživatelské* připomínky

Use Case 27: Vytvoření nové připomínky

Název: *Vytvoření nové připomínky*

Popis: Uživatel chce vytvořit novou připomínku pro sebe nebo jiného uživatele

Aktéři: *Zaměstnanec*

Vstupní podmínky: *Uživatel* je přihlášen do *systému*

Výstupní podmínky: *Uživatel* vytvořil novou připomínku

Základní scénář:

1. *Uživatel* vybere v menu položku Připomínky → Přidat připomínku
2. *Systém uživatele* vyzve k zadání následujících údajů
   a) Název složky
   b) Připomenutí (datum pomocí funkce datepicker)
   c) Popis
   d) Kategorie (uživatel vybere ze seznamu předpřipravených možností)
   e) Klient
   f) Nabídka (nebo projekt)
   g) Výběr uživatelů z připraveného seznamu

3. *Uživatel* potvrdí vytvoření připomínky tlačítkem „uložit“
4. *Systém* vytvoří novou připomínku a zobrazí ji v seznamu

Use Case 28: Splnění připomínky

Název: *Splnění připomínky*

Popis: *Uživatel* chce splnit připomínku a tím ji označit za dokončenou

Aktéři: *Zaměstnanec*

Vstupní podmínky: *Uživatel* je přihlášen do *systému,* v *systému* existuje připomínka

Výstupní podmínky: *Uživatel* dokončil připomínku

Základní scénář:

1. *Uživatel* vybere v menu položku Připomínky → Výpis
2. *Uživatel* dokončí připomínku pomocí tlačítka označeného symbolem ✓
3. *Systém* označí úkol za hotový a přestane ho zobrazovat ve výpisu

---

1.4.1 Projektový manager

Use Case 29: Smazání složky

Název: *Smazání složky*

Popis: *Uživatel* s minimální rolí projektový manager může smazat vytvořenou složku úkolů.

Aktéři: *Projektový manager*

Vstupní podmínky: *Uživatel* je přihlášen do *systému,* existuje vytvořená složka projektu

Výstupní podmínky: *Uživatel* smazal složku projektu a úkoly v ní jsou smazány.

Základní scénář:

1. *Uživatel* vybere v menu položku Todo
2. *Systém* zobrazí všechny vytvořené složky
3. *Uživatel* vybere editaci složky pomocí tlačítka označeného ikonkou „tužky“
4. *Systém* zobrazí detail složky
5. *Uživatel* vybere možnost „smazat“ pomocí tlačítka
6. *Systém* vyzve uživatele k potvrzení smazání složky
7. *Uživatel* potvrdí smazání tlačítkem „smazat“
8. *Systém* smaže vybranou složku, všechny její podsložky a úkoly v těchto složkách

Use Case 30: Zobrazení měsíčních časových přehledů pro všechny uživatele

Use case vychází ze stejného výpisu jako zobrazení vlastních časů, pouze vypisuje informace o všech uživatelích.

Use Case 31: Přidání víceprací k projektu

Název: *Přidání vícepráce k projektu*

Popis: *Projektový manager* nebo *administrátor* chce přidat schválenou náročnost vícepráce k projektu. Tyto hodiny se budou zobrazovat u projektu a rozšíří jeho časovou náročnost.

Aktéři: *Projektový manager*

Vstupní podmínky: *Uživatel* je přihlášen do *systému,* existuje projekt

Výstupní podmínky: *Uživatel* přidal schválenou pracnost vícepráce do systému.

Základní scénář:

1. *Uživatel* vybere projekt stejným způsobem, jako u Use Case: 15
2. *Uživatel* vybere možnost přidat vícepráce
3. *Uživatel* do *systému* zadá časovou náročnost v hodinách a popis vícepráce
4. *Systém* uloží vícepráce do databáze.

Use Case 32: Přidání externích uživatelů k projektu

Název: *Přidání externích zaměstnanců k projektu*

Popis: *Projektový manager* nebo *administrátor* chce přidat *externího zaměstnance* k projektu. *Externí zaměstnanec* následně uvidí zvolený projekt ve svém výpisu projektů.

Aktéři: *Projektový manager*

Vstupní podmínky: *Uživatel* je přihlášen do *systému,* existuje projekt, existuje externí zaměstnanec

Výstupní podmínky: *Uživatel* přidal *externího zaměstnance* k systému. *Externí zaměstnanec* získal přístup ke zvolenému projektu.

Základní scénář:

1. *Uživatel* vybere projekt stejným způsobem, jako u Use Case: 15
2. *Uživatel* vybere možnost přidat externího zaměstnance
3. *Systém* vyzve k vybrání *externího zaměstnanec* pomocí nabídky select, zobrazující všechny *externí zaměstnance*.
4. *Uživatel* potvrdí výběr tlačítkem „přidat“

5. *Systém* nastaví *uživateli* přístupy k projektu a zobrazí ho ve výpisu přidaných *externích zaměstnanců*

Use Case 33: Odebírání externího zaměstnance z projektu

Název: *Odebrání externího zaměstnance z projektu*

Popis: *Projektový manager* nebo *administrátor* chce odebrat *externího zaměstnance* z projektu. *Externí zaměstnanec* následně neuvidí zvolený projekt ve svém výpisu projektů.

Aktéři: *Projektový manager*

Vstupní podmínky: *Uživatel* je přihlášen do *systému,* existuje projekt, existuje externí zaměstnanec

Výstupní podmínky: *Uživatel* odebral *externího zaměstnance* k systému. *Externímu zaměstnanci* byl odebrán přístup ke zvolenému projektu.

Základní scénář:

1. *Uživatel* vybere projekt stejným způsobem, jako u Use Case: 15
2. *Uživatel* vybere uživatele ve výpisu přidaných *externích zaměstnanců*
3. *Uživatel* odebere *externího uživatele* kliknutím na ikonu křížku vedle jména *externího zaměstnance*
4. *Systém* odebere *externímu zaměstnanci* práva pro přístup k danému projektu, *externí zaměstnanec* je odebrán z výpisu přidaných *externích zaměstnanců*

Use Case 34: Vytvoření záznamu nového klienta

Název: *Vytvoření záznamu nového klienta*

Popis: Vytvoření nového záznamu o *klientovi* v záložce klienti

Aktéři: *Projektový manager, administrátor*

Vstupní podmínky: *Uživatel* je přihlášen do *systému*, existuje společnost související s klientem

Výstupní podmínky: *Uživatel* vytvořil nový záznam o klientovi

Základní scénář:

1. *Uživatel* vybere na stránce se správou klientů možnost vytvořit nového klienta
2. *Uživatel* vyplní všechny potřebné informace
   a) *Uživatel* vyplní jméno klienta
   b) *Uživatel* vyplní emailovou adresu klienta
   c) *Uživatel* vyplní telefonní kontakt na klienta
   d) *Uživatel* vybere společnost, ke které klient patří
   e) *Uživatel* v případě potřeby doplní nepovinné pole poznámka

3. *Uživatel* potvrdí zadané informace pomocí tlačítka „uložit“
4. *Systém* validuje vložené informace
5. *Systém* uloží nový záznam o hostingu

Alternativní scénář:

Pokud bude validace v bodě 4 neúspěšná, *systém* informuje *uživatele* o chybných parametrech a proces pokračuje znovu na bod č. 2 s rozdílem pouhého doplnění/opravení informací.

Use Case 35: Zobrazení detailu klienta

Název: *Zobrazení detailu klienta*

Popis: Zobrazení detailu konkrétní stránky v záložce Stránky

Aktéři: *Projektový manager, administrátor*

Vstupní podmínky:

Uživatel je přihlášen do *systému*, existuje vytvořený záznam o klientovi

Výstupní podmínky: *Uživateli* je zobrazen detail klienta

Základní scénář:

1. *Uživatel* vybere položku ze seznamu klientů
2. *Systém* zobrazí podrobné informace o klientovi

Use Case 36: Vytvoření nového záznamu o společnosti

Název: *Vytvoření nového záznamu o společnosti*

Popis: Vytvoření nového záznamu o *společnosti* v záložce klienti

Aktéři: *Projektový manager, administrátor*

Vstupní podmínky: *Uživatel* je přihlášen do *systému*

Výstupní podmínky: *Uživatel* vytvořil nový záznam o *společnosti*

Základní scénář:

1. *Uživatel* vybere na stránce se správou klientů možnost přidat společnost
2. *Uživatel* vyplní všechny potřebné informace
   a) *Uživatel* vyplní jméno společnosti
   b) *Uživatel* v případě potřeby doplní nepovinné pole poznámka
3. *Uživatel* potvrdí zadané informace pomocí tlačítka „uložit“
4. *Systém* validuje vložené informace
5. *Systém* uloží nový záznam o společnosti

Alternativní scénář:

Pokud bude validace v bodě 4 neúspěšná, *systém* informuje *uživatele* o chybných parametrech a proces pokračuje znovu na bod č. 2 s rozdílem pouhého doplnění/opravení informací.

Use Case 37: Zobrazení detailu společnosti

Název: *Zobrazení detailu společnosti*

Popis: Zobrazení detailu konkrétní společnosti v záložce klienti → společnosti

Aktéři: *Projektový manager, administrátor, zaměstnanec*

Vstupní podmínky:

Uživatel je přihlášen do *systému*, existuje vytvořený záznam o společnosti

Výstupní podmínky: *Uživateli* je zobrazen detail společnosti

Základní scénář:

1. *Uživatel* vybere záložku klienti → společnosti v menu
2. *Systém* zobrazí výpis existujících společností
3. *Uživatel* vybere detail konkrétní společnosti kliknutím na její název
4. *Systém* zobrazí podrobné informace o společnosti

Use Case 38: Editace detailu společnosti

Název: *Editace detailu společnosti*

Popis: Editace detailu konkrétní společnosti v záložce klienti → společnosti

Aktéři: *Projektový manager, administrátor*

Vstupní podmínky:

Uživatel je přihlášen do *systému*, existuje vytvořený záznam o společnosti

Výstupní podmínky: *Uživatel* upravil detail společnosti

Základní scénář:

1. *Uživatel* vybere záložku klienti → společnosti v menu
2. *Systém* zobrazí výpis existujících společností
3. *Uživatel* vybere konkrétní společnost, kterou chce upravovat kliknutím na ikonu označující možnost editace
4. *Systém* zobrazí editovatelné informace o společnosti
5. *Uživatel* změní informace o společnosti
6. *Uživatel* uloží změny pomocí tlačítka určeného pro uložení změn
7. *Systém* validuje změny a pokud jsou změny validní uloží je do databáze

Use Case 39: Zobrazení budoucího plánu

Název: *Zobrazení budoucího plánu*

Popis: Zobrazení informací o stavu lidských zdrojů a náročnosti projektů

Aktéři: *Administrátor, projektový manager*

Vstupní podmínky: *Uživatel* je přihlášen do *systému*

Výstupní podmínky: *Uživatel* vidí informace o lidských zdrojích

Základní scénář:

1. *Uživatel* v menu vybere možnost Projekt → Budoucí plán
2. *Systém* zobrazí informace o stavu lidských zdrojů a práce na další měsíce

---

Use Case 40: Filtrování budoucího plánu

Název: *Filtrování budoucího plánu*

Popis: Filtrování informací dle parametrů přehledu budoucího plánu

Aktéři: *Administrátor, projektový manager*

Vstupní podmínky: *Uživatel* je přihlášen do systému

Výstupní podmínky: *Uživatel* vidí informace v přehledu budoucího plánu filtrované podle časového úseku (den, týden, měsíc)

Základní scénář:

1. *Uživatel* v menu vybere možnost Projekt → Budoucí plán
2. *Uživatel* vybere ve filtru možnost den, týden, měsíc dle potřeby
3. *Uživatel* potvrdí filtr
4. *Systém* zobrazí informace o stavu lidských zdrojů a práce na další časové období v intervalu zvoleném ve filtru

---

Use Case 41: Vytvoření nového hostingu

Název: *Vytvoření nového hostingu*

Popis: Vytvoření nového hostingu v záložce Hostingy

Aktéři: *Projektový manager*

Vstupní podmínky: *Uživatel* je přihlášen do *systému*

Výstupní podmínky: *Uživatel* vytvořil nový záznam o hostingu

Základní scénář:

1. *Uživatel* vybere na stránce se správou hostingu možnost „vytvořit nový hosting“

2. *Uživatel* vyplní všechny potřebné informace
   a) *Uživatel* vyplní jméno hostingové společnosti
   b) *Uživatel* vyplní URL adresu administrace hostingu
   c) *Uživatel* vyplní cenu hostingu za rok, pokud se jedná o hosting společnosti
   d) *Uživatel* vyplní přihlašovací údaje do administrace hostingu
   e) *Uživatel* doplní nepovinné údaje (např. logo)
3. *Uživatel* potvrdí zadané informace pomocí tlačítka „uložit“
4. *Systém* validuje vložené informace
5. *Systém* uloží nový záznam o hostingu

Alternativní scénář:

Pokud bude validace v bodě 4 neúspěšná, *systém* informuje *uživatele* o chybných parametrech a proces pokračuje znovu na bod č. 2 s rozdílem pouhého doplnění/opravení informací.

Use Case 42: Editace hostingů

Název: *Editace vytvořeného hostingu*

Popis: Editace údajů o hostingu

Aktéři: *Projektový manager*

Vstupní podmínky: *Uživatel* je přihlášen do *systému*, existuje záznam o hostingu

Výstupní podmínky: *Uživatel* upravil údaje o hostingu

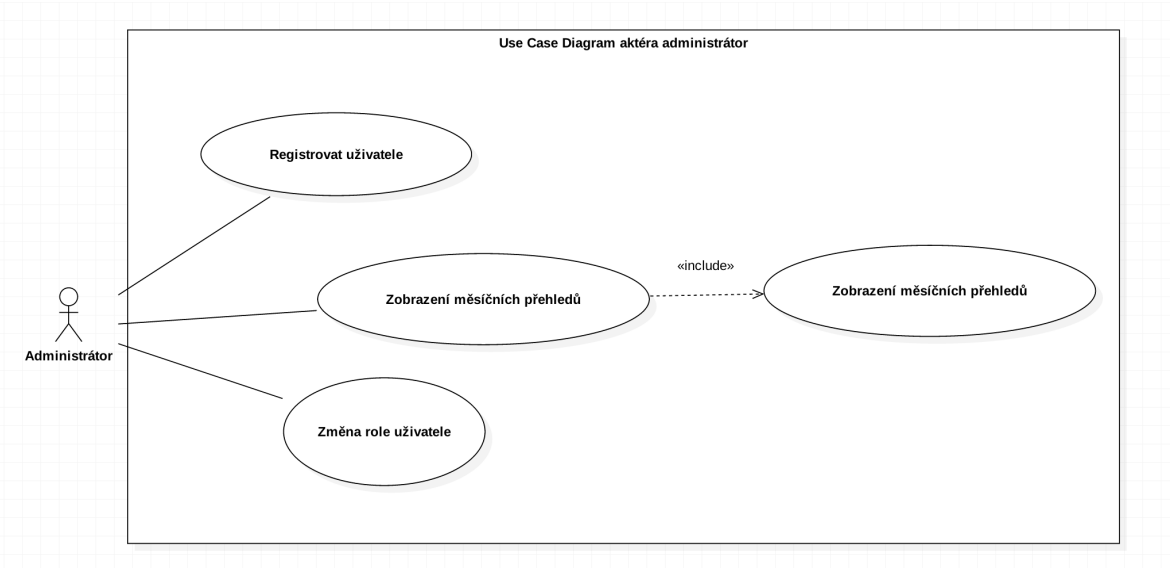
Základní scénář:

1. *Uživatel* zobrazí seznam hostingů viz. Use Case 11
2. *Uživatel* vybere položku, kterou chce upravovat a kliknutím na ikonu „tužky“
3. *Systém* vyzve uživatele k vyplnění informací o hostingu
4. *Uživatel* upraví záznam a uloží ho pomocí tlačítka „uložit“
5. *Systém* uloží změny v záznamu o hostingu

Alternativní scénář:

Pokud bude validace v bodě 4 neúspěšná například hesla se neshodují, *systém* informuje *uživatele* o chybných parametrech a proces pokračuje znovu na bod č. 3 s rozdílem pouhého doplnění/opravení informací.

1.5 Administrátor



Obrázek 45: Use Case diagram aktéra administrátor

Use Case 43: Registrace uživatele

Název: *Registrace uživatele*

Popis: Registrace uživatele do systému

Aktéři: *Administrátor*

Vstupní podmínky: *Uživatel* není registrován v systému

Výstupní podmínky: *Uživatel* je registrován v systému

Základní scénář:

1. *Administrátor* chce registrovat nového *uživatele* do *systému*
2. *Systém* vyzve *uživatele* k zadání uživatelského emailu a jména
3. *Administrátor* vyplní email a jméno *uživatele*
4. *Systém* vyzve *uživatele* k vyplnění hesla a hesla pro ověření, resp. ověří, zda jsou hesla stejná
5. *Administrátor* vygeneruje bezpečné heslo pro nového *uživatele* a zadá ho ve dvou kopiích
6. *Systém* ověří jedinečnost uživatelského emailu a shodnost hesel
7. *Systém* vyzve *administrátora* k vyplnění uživatelských rolí
8. *Administrátor* vybere jednu nebo více uživatelských rolí
9. *Administrátor* registraci potvrdí
10. *Systém* registruje nového *uživatele*

Alternativní scénáře:

Administrátor v bodě 5 zadá nedostačující nebo neshodné heslo

1. *Systém* informuje *administrátora* o nedostačujícím/chybném zadání hesla nebo o neshodnosti hesel, scénář pokračuje znovu bodem 4. *Administrátor* v bodě 3 zadá již existující emailovou adresu
2. *Systém* informuje *administrátora* o existenci účtu se stejnou emailovou adresou, vyzve ho k zadání nové adresy a scénář pokračuje bodem 3

Use Case 44: Změna role uživatele

Název: *Změna role uživatele*

Popis: Změna uživatelské role

Aktéři: *Administrátor*

Vstupní podmínky: *Uživatel* je přihlášen do *systému*

Výstupní podmínky: *Uživatel* má přiřazené nové role

Základní scénář:

1. *Administrátor* vybere účet *uživatele*, kterému chce změnit roli
2. *Administrátor* změní role
3. *Administrátor* změnu potvrdí
4. *Systém* upraví uživatelské role

Use Case 45: Zobrazení měsíčních přehledů

Název: *Zobrazení měsíčních přehledů*

Popis: Zobrazení měsíčních přehledů

Aktéři: *Administrátor*

Vstupní podmínky: *Uživatel* je přihlášen do *systému*

Výstupní podmínky: *Uživatel* může prohlížet měsíční přehledy

Základní scénář:

1. *Administrátor* na hlavní stránce aplikace vidí informace o aktuálním a následujícím měsíci
2. *Administrátor* přejde na stránku s přehledy pomocí tlačítka „měsíční přehled“
3. *Administrátor* vidí informace o následujících měsících

Use Case 46: Filtrování měsíčních přehledů

Název: *Filtrování měsíčních přehledů*

Popis: Filtrování informací dle parametrů měsíčních přehledů

Aktéři: *Administrátor*

Vstupní podmínky: *Uživatel* je přihlášen do systému

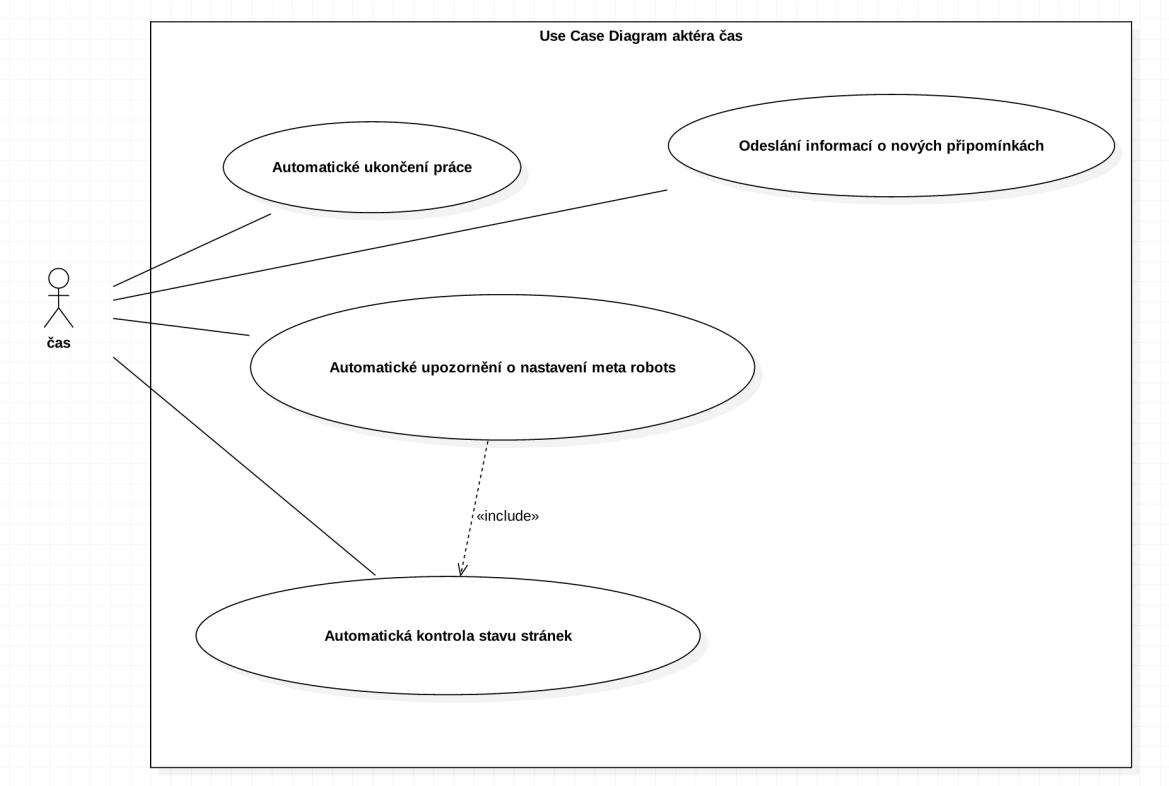
Výstupní podmínky: *Uživatel* vidí pouze přesněji specifikované projekty

Základní scénář:

1. *Administrátor* na stránce měsíčních přehledů specifikuje filtr
2. *Systém* zobrazí pouze projekty odpovídající zvolenému filtru

---

1.6 Čas



Obrázek 46: Use Case diagram aktéra Čas

Use Case 47: Automatické ukončení práce

Název: *Automatické ukončení práce*

Popis: V případě zapomenutí vypnutí měření času systém ve stanovenou noční hodinu vypne všechny běžící stopky. Ke každému úkolu vypnutému tímto způsobem přidá poznámku o automatickém ukončení měření.

Aktéři: *Čas*

Vstupní podmínky: Nastala doba automatického ukončení běžících úkolů, existuje úkol, ke kterému se stále měří čas

Výstupní podmínky: Čas úkolu je pozastaven a je k němu přiřazena poznámka o ukončení

Základní scénář:

1. *Uživatel* využívá funkce inteligentních stopek
2. *Uživatel* neukončil práce ke stanovené hodině
3. *Systém* ukončuje všechny práce a ke každému úkolu přiřazuje poznámku o automatickém ukončení

Use Case 48: Automatická kontrola stavu stránek

Název: *Automatická kontrola stavu stránek*

Popis: Systém každý den v nastavenou hodinu provádí kontrolu webových stránek. Systém aktualizuje informace v systému dle aktuálního stavu.

Aktéři: *Čas*

Vstupní podmínky: Existuje záznam o webové stránce

Výstupní podmínky: Všechny stránky jsou zkontrolovány, informace o stavu jsou aktualizovány

Základní scénář:

1. *Systém* pro každý záznam ze seznamu webových stránek spustí skript kontrolující její nastavení a stav
2. *Systém* aktualizuje údaje dle získaných hodnot

Use Case 49: Automatické upozornění o nastavení meta robots

Název: *Automatické upozornění o nastavení meta robots*

Popis: Systém každý den v nastavenou hodinu provádí kontrolu webových stránek (viz

Use Case 48). Systém pravidelně odesílá informace o stránkách s nesprávným nastavením meta tagů na administrátorský email.

Aktéři: *Čas*

Vstupní podmínky: *Systém* provedl kontrolu webových stránek, existuje stránka s nesprávným nastavením meta tagů

Výstupní podmínky: *Systém* odešle administrátorovi email obsahující seznam všech stránek s nesprávným nastavením a informací o nekorektním stavu

Základní scénář:

1. *Systém* provede kontrolu webových stránek (viz.
2. Use Case 48)
3. *Systém* nalezl minimálně jednu webovou stránku s nesprávným nastavením
4. *Systém* odesílá informační email *administrátorovi*.

Use Case 50: Odeslání informací o nových připomínkách

Název: *Odeslání informací o nových připomínkách*

Popis: Systém každý den v nastavenou hodinu provádí kontrolu připomínek. V případě, že existují nové připomínky, informuje přiřazené uživatele pomocí emailu.

Aktéři: *Čas*

Vstupní podmínky: *Systém* provedl kontrolu připomínek, existuje nová připomínka

Výstupní podmínky: *Systém* odešle uživatelům s novou připomínkou email

Základní scénář:

1. *Systém* provede kontrolu připomínek
2. *Systém* nalezl minimálně jednu novou připomínku
3. *Systém* odesílá souhrnný email každému uživateli s novou připomínkou

---

Příloha B: Instalační příručka

Aplikace pro svůj běh potřebuje server s verzí PHP alespoň 7.0. Databázový server podporující MySQL v nejnovější verzi.

Všechny soubory potřebné pro Composer jsou nainstalovány, v případě problémů je nutno nainstalovat balíčkovací systém Composer ze stránek https://getcomposer.org/. Následně pomocí terminálu nebo console přejít do adresáře na ~/Aplikace/ a spustit v příkazové řádce příkaz *„composer install“*.

Pokud aplikaci spouštíte na lokálním prostředí je potřeba nahrát data ze složky db do nově vytvořené databáze.

Přístupy k vytvořené databázi je potřeba vyplnit do souboru *config.local.neon*, k nalezení v adresáři: *~/Aplikace/pipi_app/app/config/*

Lokální php server je potřeba nasměrovat do složky *~/Aplikace/pipi/*

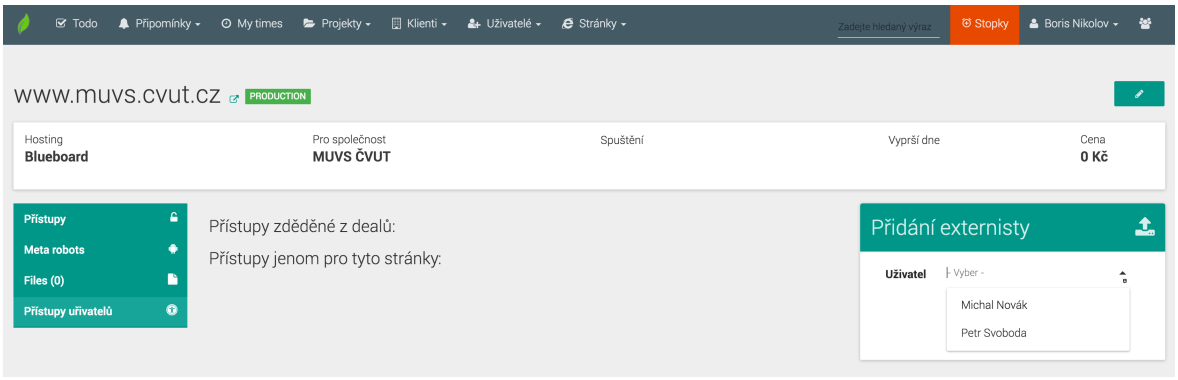
Základní přístupové údaje do aplikace jsou následující:

Přihlašovací jméno: admin@ipip.cz
Heslo: pQoy3FGph0PYDYCttoyOJ

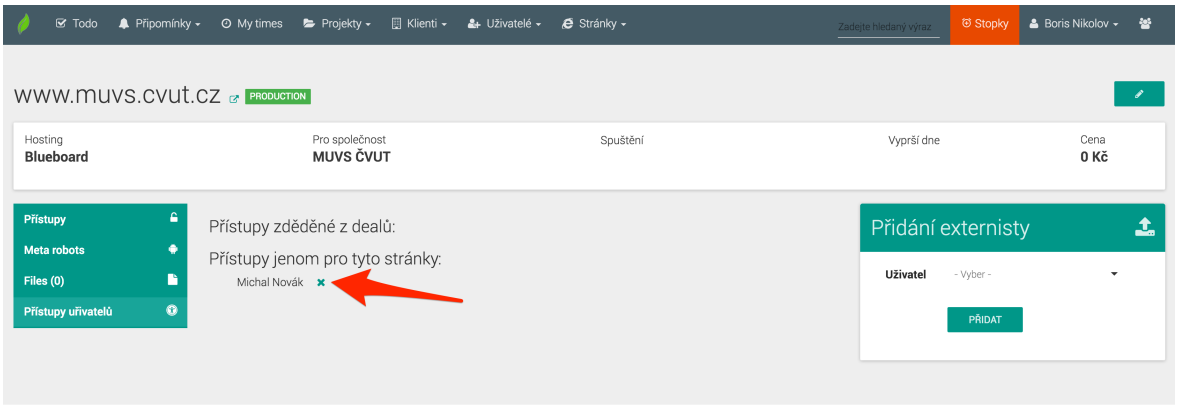
Pro testovací účely bude omezenou dobu vytvořeno testovací prostředí od 1.6.2017 na webové stránce http://dp.borisnikolov.cz

Pro přihlášení budou fungovat přístupové údaje výše.

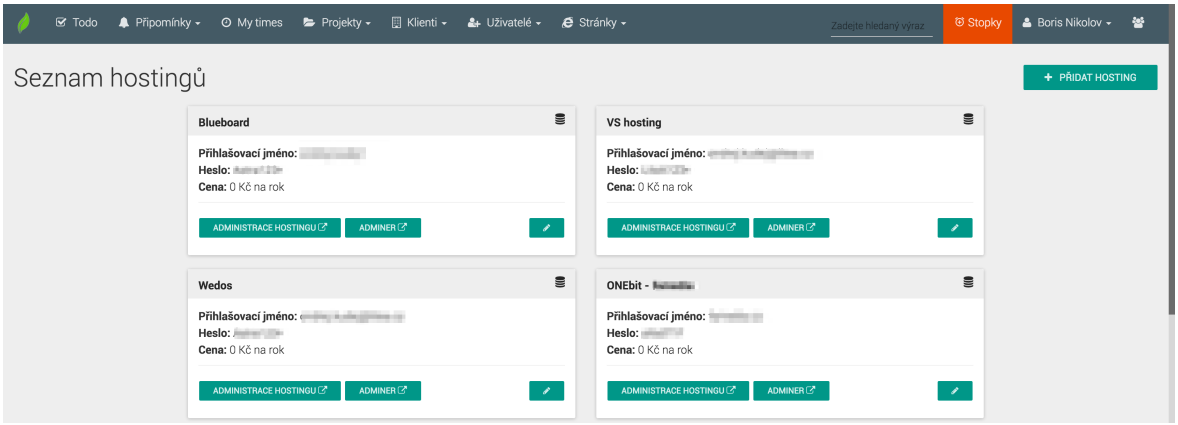
Příloha C: Uživatelské rozhraní a obrázky



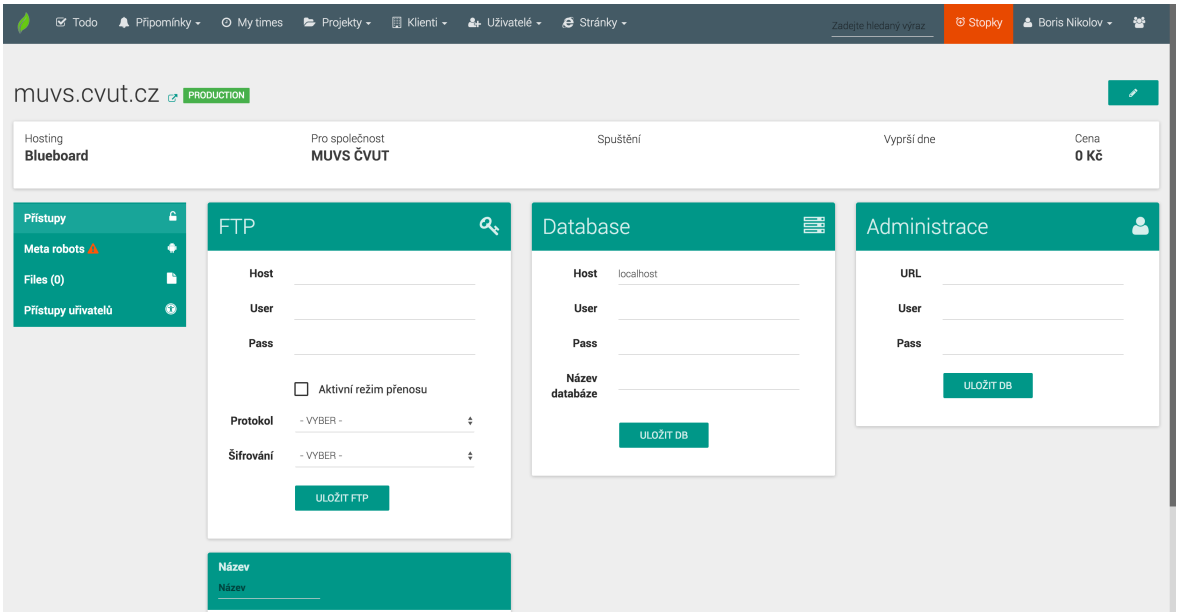
Obrázek 47: Přidání externího zaměstnance ke stránce



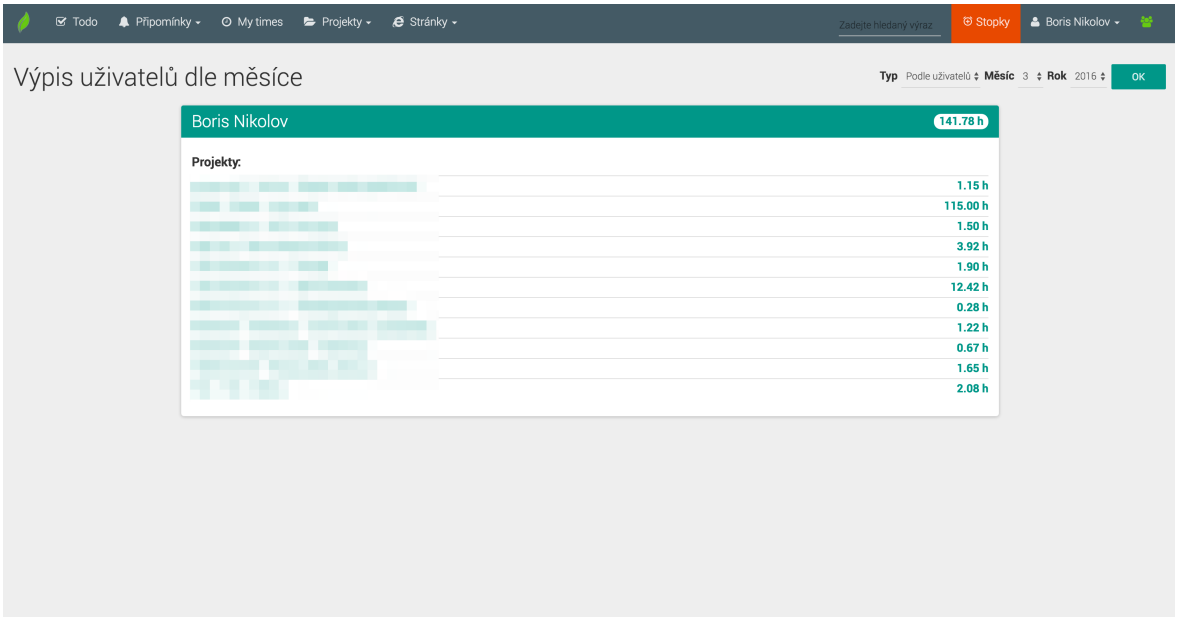
Obrázek 48: Odebrání přístupu externímu zaměstnanci k vybrané webové stránce



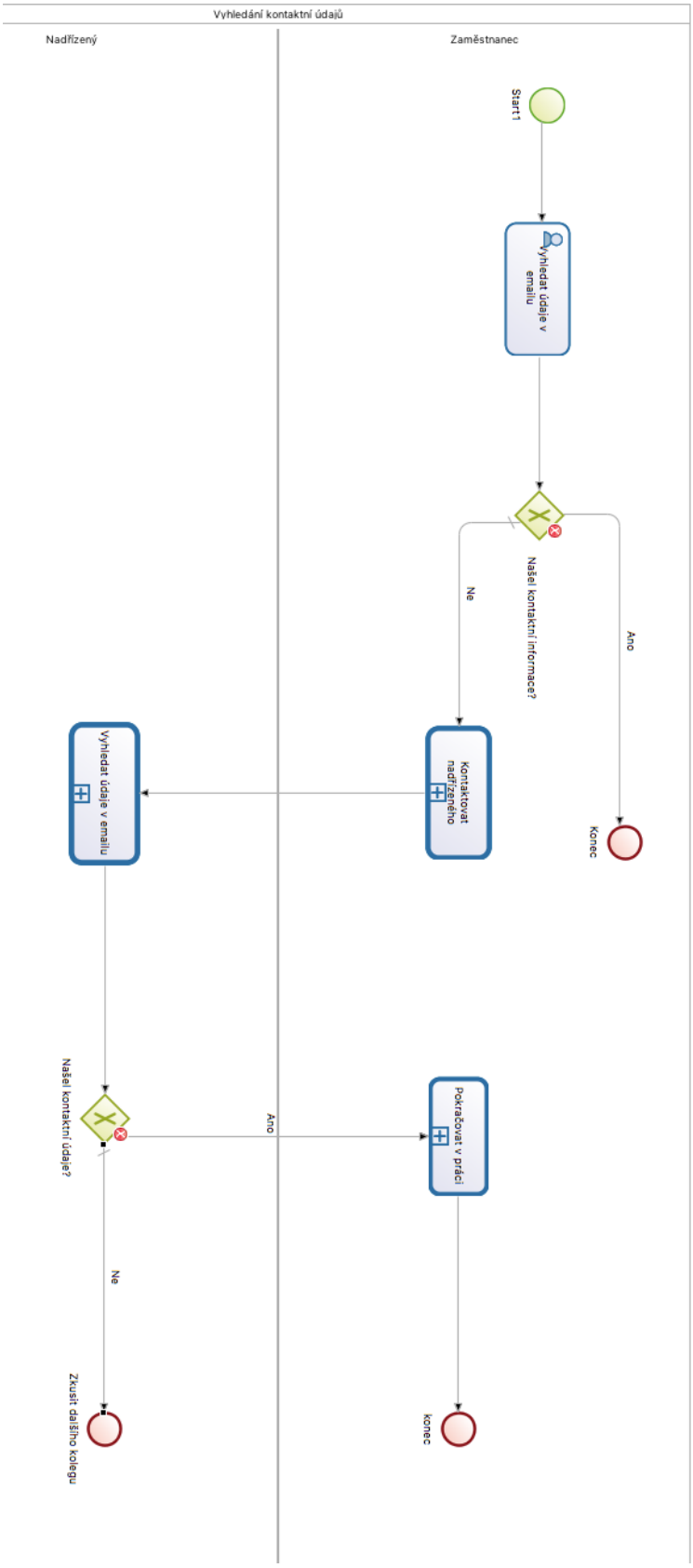
Obrázek 49: Výpis hostingových služeb



Obrázek 50: Editace detailu webové stránky



Obrázek 51Výpis odpracovaných hodin uživatel, zobrazeno dle měsíce



Obrázek 52: Schéma získávání informací

Příloha D: Obsah přiložených souborů

~/Poster.pdf – Poster k této práci

~/NikolovBoris_DP.pdf – text této práce ve formátu PDF

V souboru aplikace.zip jsou následující důležité soubory:

~/Readme.txt – instrukce k instalaci aplikace

~/Aplikace – soubory aplikace

~/Aplikace\composer.json – soubor obsahující konfiguraci potřebných balíčků pro Composer

~/Aplikace\db – databáze potřebná pro běh systému

~/Aplikace/pipi_app/app/config.local.neon – nastavení databáze pro lokální prostředí

Evidence výpůjček

Prohlášení:
Dávám svolení k půjčování této diplomové práce. Uživatel potvrzuje svým podpisem, že bude tuto práci řádně citovat v seznamu použité literatury.

Jméno a příjmení

V Praze dne: podpis: ……

Jméno	Katedra / Pracoviště	Datum	Podpis