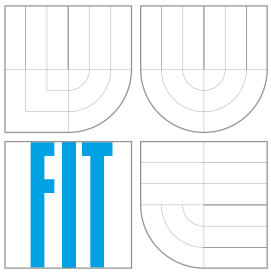
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

WEBOVÝ NÁSTROJ PRO TVORBU A SPRÁVU ČINNOSTÍ A CÍLŮ PROJEKTŮ

WEB TOOL FOR DESIGN AND MANAGEMENT OF PROJECT ACTIVITIES AND GOALS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — GABRIEL BRANDERSKÝ
AUTHOR

VEDOUCÍ PRÁCE — Ing. VÍTĚZSLAV BERAN, Ph.D.
SUPERVISOR

BRNO 2014

Abstrakt

Táto práca sa zaoberá plánovaním činností a cieľov projektov vedúce na ich efektívne a úspešné zvládnutie. Výsledkom je webový nástroj, ktorý má zakomponované dobré princípy z riadenia a manažovania projektov a gamifikáciu. Cieľovou skupinou sú predovšetkým študenti vysokých škôl, tí sa počas svojho štúdia stretávajú s mnohými projektami. Medzi najdôležitejšie rozhodne patrí bakalárska a diplomová práca, preto im je venovaná zaslúžená pozornosť, aby nástroj odpovedal ich špecifickým potrebám.

Abstract

This thesis deals with planning of project activities and goals which leads to effective and successful outcomes. The result is the web tool which embeds good principles of project management and gamification which is targeted especially to college students. They are dealing with many projects during their studies. The most important of these projects are bachelor and diploma thesis and that is a reason why extra attention is devoted to these projects and their specific needs.

Klíčová slova

Manažovanie projektov, Webové technológie, Gamifikácia, Užívateľské rozhranie, Ruby on Rails

Keywords

Project management, Web technologies, User interface, Ruby on Rails

Citace

Gabriel Branderský: Webový nástroj pro tvorbu a správu činností a cílů projektů, bakalářská práce, Brno, FIT VUT v Brně, 2014

Webový nástroj pro tvorbu a správu činností a cílů projektů

Prohlášení

Prehlasujem, že som bakalársku prácu vypracoval samostatne pod vedením pána Ing. Vítězslava Berana, Ph.D. Uviedol som všetky literárne pramene a publikácie z ktorých som čerpal.

.....
Gabriel Branderský
19. května 2014

Poděkování

Rád by som sa poďakoval pánu Ing. Vítězslavu Beranovi, ktorý mi poskytol možnosť pracovať na tejto zaujímavej práci, viedol ma správnym smerom a predal mnoho cenných rád.

Obsah

Kapitola 1

Úvod

Žijeme vo svete zvyšujúcej sa komplexity s ktorou sa neustále stretávame, či už v každodennom živote alebo v rámci pracovných projektoch. Preto nie je prekvapujúce, že na zvládnutie tejto komplexity sa využíva výpočetná sila počítačov, ktorá je oproti ľudskému mozgu vhodnejšia na ukladanie veľkého množstva presných informácií. Využitie informačných technológií pomôže nielen „uvolniť“ kapacitu v našich hlavách, ale aj získať lepšiu kontrolu nad ohromujúcim závalom informácií súčasnosti.

Táto bakalárska práca sa zaoberá vytvorením webového nástroja na tvorbu a správu činností a cieľov projektov vedúce k ich efektívnemu zaobchádzaniu a úspešnému zvládnutiu. Cieľovou skupinou sú predovšetkým študenti vysokých škôl, ktorí sa počas svojho štúdia stretávajú s mnohými projektami. Medzi najdôležitejšie rozhodne patrí bakalárska a diplomová práca, preto je venovaná zaslúžená pozornosť. Tieto projekty okrem iného vyžadujú dobrú organizáciu a plánovanie avšak študenti nie vždy majú dostatočné schopnosti v týchto oblastiach. A to je hlavnou motiváciou na vytvorenie nástroja, ktorý bude mať v sebe zakomponované princípy z riadenia projektov a techniky efektívneho dosahovania cieľov. Nutno podotknúť, že použitie na iné účely ako sú napríklad bežné povinnosti nie je vylúčené. Priam naopak, je veľmi žiadané, aby si študenti osvojili tieto postupy a preniesli ich do každodenného života. To im v konečnom dôsledku pomôže dosahovať lepšie výsledky na projektoch.

Prvá kapitola poskytne teoretický základ, ktorý predstaví webové technológie, metodológie a systémy riadenia projektov, zásady a postupy pri tvorbe a testovaní užívateľského rozhrania, moderný trend gamifikácie, súvisiacu psychológiu a tiež prehľad existujúcich riešení. Návrh zahŕňa definíciu cieľov, vysvetlenie návrhu užívateľského rozhrania a dátovej vrstvy, ktoré sa predstavia s využitím modelovacích techník. V ďalších kapitolách popíšem spôsob akým som realizoval aplikáciu a ako som postupoval pri implementácií a testovaní aplikácie ako aj jednotlivé komponenty aplikácie. V neposlednej rade popíšem aké experimenty som vykonal aké výsledky, ktoré to prinieslo, čo zhodnotím a zhrniem v závere práce.

Kapitola 2

Teória

Táto kapitola zoznamuje s teóriou na ktorej je práca postavená. V prvom rade sa pozrieme na technológie, ktoré sa využívajú na tvorbu moderných webových aplikácií. Uvedené technológie predstavujú jednu z mnohých vhodných kombinácií, samozrejme nie jedinú možnú. Rovnako existujú ďalšie spôsoby riadenia projektov a budú spomenuté len tie, ktoré sa týkajú tejto práce.

Ak realizovaná aplikácia neposkytuje to, čo užívateľ potrebuje alebo vo forme ktorej rozumie, tak je nepoužiteľná. Preto sa zaoberáme tvorbou a testovaním užívateľského rozhrania. Aj gamifikáciu je možné chápať ako vytváranie užívateľského rozhrania s atribútmi, ktoré sú dôležité pre ľudí alebo tiež ako techniku efektívneho dosahovania cieľov, čo bude tiež predstavené v psychológií. Na záver sa pozrieme na prehľad existujúcich riešení.

2.1 Webové technológie

V tejto sekcií postupne predstavím rôzne skupiny technológií, ktoré sa líšia zložitosťou a úlohou, ktorú plnia. Tá závisí aj na umiestnení v architektúre webových aplikácií, ktorá pozostáva z klienta (webový prehliadač), ktorý posiela serveru požiadavky sieťovým protokolom HTTP a server ich spracováva a posiela odpovede. Na zodpovedanie požiadavku sa môže podieľať viacej serverov najčastejšie sa používa trojvrstvová architektúra, ktorá oddeľuje aplikačný server a databázový server. Preto na vytvorenie webovej aplikácie je potrebné zaoberať klientskými aj serverovými technológiami, ktoré sa s rozširovaním Internetu vyvíjajú a stávajú sa sofistikovanejšie. Táto sekcia čerpá z [9].

Klientské technológie

HTML (HyperText markup language) je hypertextový značkovací jazyk, ktorý definuje štruktúru a obsah dokumentu. Podporuje aj úpravu vzhľadu avšak k tomu sa v súčasnosti už nepoužíva. Pretože je vhodné definíciu štruktúry a vzhľadu oddeliť.

Oddelenie vzhľadu sa dosahuje pomocou CSS (Cascading Style Sheets), ktoré je tiež nazývané ako kaskádové štýly, ktoré pridávajú dokumentu štýl a vzhľad. Oddelený štýl je tak ľahšie udržiavateľný a navyše môžu existovať rôzne vzhľady pre rovnaký HTML dokument. To sa využíva predovšetkým na prispôsobenie rôznym zariadeniam ako sú notebooky, inteligentné telefóny a iné. Avšak štýlové súbory sa s vývojom stávajú väčšie a ťažko udržiavateľné. Preto nastúpili pre-procesory, ktoré podporujú programové konštrukcie na uľahčenie zápisu v CSS.

Jazyk SASS [1] (Syntactically Awesome Stylesheets) podporuje okrem iného dedičnosť, vnorenie a premenné. Pre-procesor jazyka SASS vytvorí opäť CSS súbor, ktorý je spracovateľný prehliadačom.

Na rozdiel od HTML a CSS, ktoré vytvárajú len statické stránky, Javascript umožňuje pridať interaktívne prvky. Je to dynamický objektovo-orientovaný jazyk, ktorého interpret je možné nájsť v takmer každom prehliadači, avšak existujú rôzne verzie Javascriptu v rôznych prehliadačoch. Aj preto sa častejšie používajú Javascriptové knihovny ako je jQuery [2], ktoré poskytuje aplikačné rozhranie (API) nezávislé na prehliadači a zjednodušuje manipuláciu s HTML dokumentom, spracovanie udalostí, vytváranie animácií a iné.

CoffeeScript [3] je jednoduchý jazyk, ktorý sa prekladá do Javascriptu. Jeho syntax je inšpirovaná modernými jazykmi ako sú Python a Ruby, čo je možné vidieť aj na syntaktickom zápise bez nutnosti používania bodkočiarok za príkazmi alebo zátvoriek (ak je to jednoznačné). Taktiež sa používa indentácia na vyznačenie programových konštrukcií.

Bootstrap [4] je veľmi populárny framework, ktorý urýchľuje vytváranie klientských častí s dôrazom na podporu rôznych zariadení (tzv. responzívny dizajn) a rozšíriteľnosť. Poskytuje rôzne šablóny, komponenty, Javascriptové pluginy na zjednodušenie častých činností.

Serverové technológie

Medzi najrozšírenejší a zároveň najdlhšie používaný serverový jazyk patrí jazyk PHP [5] (Hypertext Preprocessor). Od svojho vzniku prešiel dlhým vývojom napríklad stal sa objektovo orientovaným jazykom, aby poskytol prostriedky na vytváranie moderných stránok, čo sa mu úspešne darí. Avšak tento vývoj zanechal stopy a prichádzajú nové technológie ako Ruby s modernými vlastnosťami a charakteristikami a preto sa stávajú čoraz častou voľbou.

Ruby je interpretovaný a objektovo orientovaný jazyk, ktorého syntax je navrhnutá na prirodzené čítanie a zápis. Takmer všetko tj. čísla, reťazce a iné v Ruby je objekt, vďaka čomu je jazyk veľmi flexibilný. Získal si popularitu aj vďaka webovému frameworku Ruby on Rails [6], ktorý je vyvíjaní ako otvorený softwér a poskytuje prostriedky špecifické pre webové aplikácie ako aj mnoho knihovien tzv. gemy.

Ruby on Rails je MVC (model, view, controller) framework a pri tejto architektúre je aplikácia rozdelená do troch komponent, ktoré sa nazývajú model, kontrolér a pohľad. Komponenty sú vyvíjané ako separátne časti a framework ich na základe inteligentných implicitných nastavení spojí dokopy. Presne podľa filozofie Ruby on Rails: „Uprednostňovať konvenciu pred konfiguráciou“. Model sa stará o zachovávanie stavu aplikácie, či už prechodného alebo trvalého. Navyše sa stará o dodržiavanie obmedzení dát špecifických pre aplikáciu. Pohľad sa stará o zobrazenie dát z modelu, prípadne viacerých modelov. V okamihu keď sú dáta zobrazené úloha pohľadu končí tj. nestará sa o spracovanie vstupov, nato slúži model. A nakoniec kontrolér, ktorý sa

[1] http://sass-lang.com/
[2] http://jquery.com/
[3] http://coffeescript.org/
[4] http://getbootstrap.com/
[5] http://www.php.net/
[6] http://rubyonrails.org/

stará o spracovávanie a koordináciu akcií užívateľa, komunikuje s modelmi a zobrazuje príslušný pohľad. Nato, aby jednotlivé komponenty zodpovedali HTTP požiadavok je potrebné vykonať postupnosť operácií. Najprv požiadavok prechádza routerom, kde sú definované pravidlá ako požiadavok spracovať. Prvé odpovedajúce pravidlo identifikuje akciu (metódu) kontroléra, ktorá sa má vykonať. Spustená akcia väčšinou komunikuje s modelom a tieto data následne predáva do príslušného pohľadu.

Ruby on Rails používa tiež mapovanie relačnej databáze na objekty aplikácie, skrátene ORM (Object-relational mapping). Je to technika pri ktorej sú atribúty a vzťahy objektov uložené v databázi. Sú jednoducho opätovne prístupné bez nutnosti písať SQL dotazy a celkovo s menším počtom prístupov k databázi.

SQLite obsahuje celú databázu v jednom súbore, nevyžaduje konfiguráciu a server. Je to implicitné nastavenie databáze vo vývojovom a testovacom prostredí v Ruby on Rails aplikáciach. Avšak Ruby on Rails sa snaží spraviť aplikáciu nezávislú od databáze a jednoducho tak databázu zmeniť predovšetkým v produkcií.

2.2 Agilné riadenie projektov

Riadenie projektov sa potýka s mnohými problémami ako sú komplexita (projekty sú zložité systémy skladajúce sa z mnohých častí, ktoré sú medzi sebou prepojené) a premenlivosť (svet je predmetom neustálych zmien) a ich nezvládnutie viedlo k neúspechu mnohých projektov. Agilné riadenie referuje k hodnotám a princípom (ako napríklad iteratívny a inkrementálny vývoj), ktoré naznačujú ako postupovať, aby projekt dosiahol úspešný koniec. Kanban a Scrum sú dve konkrétne metódy, ktoré ilustrujú ako využiť tieto princípy v praxi a budú využité aj pri návrhu. [1]

Kanban [1]

Kanban je štíhly (lean) prístup k agilnému vývoji. Vznikol ako jednoduchý spôsob vizualizácie progresu pre pracovníkoch v továrňach. Jeho použitie pri vývoji software je celkom nové avšak v továrňach sa používa už vyše pol storočia.

Zakladá sa na jednoduchých pravidlách: vizualizuj prácu a limituj rozpracovanú prácu. Aj napriek jednoduchým pravidlám ponúka veľa možnosti. Na vizualizáciu sa často používa tabuľa s pomenovanými stĺpcami v ktorom sa položky posúvajú zľava doprava.

V časti o existujúcich riešenia bude popísaná aplikáciu Trello na obrázku 2.2, ktorá predstavuje ukážku vizualizácie práce pomocou Kanban systému.

Scrum [1]

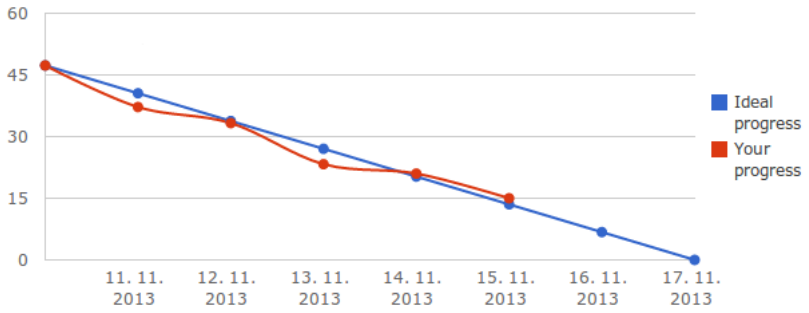
Scrum je jedna z najpopulárnejších agilných metodológií v súčasnosti. Medzi základné vlastnosti patrí viac rolový tým, denné stretnutia a plánovanie po tzv. šprintoch. Viac rolový tým pozostáva typicky zo Scrum lídra, vývojárov, testerov a analytikov. Avšak Scrum môže byť použitý aj jednou osobou, čo je navyše veľmi bežné.

Životný cyklus Scrum začína zberom požiadavkov, ktoré sú nazývané užívateľské príbehy. Scrum je iteratívna metodológia, ktorá umožňuje tieto požiadavky meniť. Na začiatku šprintu sa vyberú najdôležitejšie príbehy na ktorých sa v daný šprint bude pracovať. Každý deň šprintu sa začína krátkym stretnutím členov týmu na ktorom sa

preberá, čo sa podarilo spraviť a ako sa bude pokračovať. Rovnako ako pri Kanban sú vizualizované pomocou tabule.

Po šprinte nasleduje retrospektíva na ktorej sa každý člen tímu zamyslí nad tým, čo sa dá zlepšiť. Typickou formou sú zoznamy toho, čo by sa malo prestať robiť, v čom pokračovať a čo začať robiť. Následne sa vyberú nové príbehy do ďalšieho šprintu a vykoná sa ďalšia iterácia.

Scrum používa premyslený spôsob odhadovania práce a sledovania progresu. Odhadujú sa jednotlivé úlohy v šprinte, čím získame celkový odhad doby trvania šprintu. Na základe toho sa práca rovnomerne rozdelí do jednotlivých dní šprintu. Progres je vizualizovaný pomocou spaľovacieho grafu 2.1, ktorý na y-ovej osi zobrazuje zostávajúcu prácu a na x-ovej osi deň šprintu. Každým dnom šprintu zostávajúca práca ubúda v až a v posledný deň šprintu by mala skončiť na nule. V ideálnom prípade by zostávajúca práca ubúda lineárne, čo je tiež naznačené v spaľovacom grafe.



Obrázek 2.1: Spaľovací graf na meranie progresu v šprinte

2.3 Tvorba a testovanie užívateľského rozhrania

Pri tvorbe užívateľského rozhrania (user interface, skrátene UI), ktoré efektívne komunikuje s užívateľom je nutné zaoberať sa interakciou človeka s počítačom. Pozornosť je zameraná na vlastnosti systému tak ako ich vníma užívateľ, aby používanie UI bolo príjemné. Nielsenovy heuristiky obsahujú základné doporučenia, ktorými sa riadiť pri tvorbe UI. [7]

1. Viditeľnosť - Užívateľovi je poskytovaná spätná odozva, aby vedel v akom stave sa systém nachádza a akú činnosť vykonáva.
2. Súhlas reálneho sveta a systému - Systém by mal logicky prezentovať informácie tak ako sú známe užívateľovi známe z reálneho sveta. Mal by hovoriť rečou a spôsobom, ktorému užívateľ rozumie.
3. Kontrola a voľnosť - Užívateľ ma možnosť skúmať a experimentovať. V prípade chyby systém ponúka možnosť „naspäť“.
4. Konzistencia a štandardy - Systém je jednotný a používa prvky na ktoré je užívateľ zvyknutý. Akákoľvek odchýlka spôsobuje zmätenie a neistotu.
5. Prevencia chýb - V najlepšom prípade by systém nemal dovoliť urobiť chybu alebo by im mal predchádzať, čo je dôležitejšie ako informatívne chybové hlášky.

6. Flexibilita a efektívnosť - Prispôsobenie úrovni znalostí užívateľa v podobe klávesových skratiek pre experta a zároveň jednoduché ovládanie pre začiatočníka.

7. Minimalistický dizajn - Akákoľvek nepodstatná informácia zaberá miesto a komplikuje nájdenie relevantnej informácie.

8. Nápoveda a dokumentácia - Ak sa vyskytnú problémy je dostupná dokumentácia a nápoveda, ktorá v presných krokoch popisuje ako postupovať pri riešení.

Zatiaľ, čo Nielsenovy heuristiky predstavujú obecné princípy, ktorými sa pri riadiť pri vytváraní UI, tak teraz popíšem konkrétne prvky a ako prispievajú k lepšiemu UI. V týchto prvkoch je tiež možné vidieť obecnejší princíp - „vykonaj to priamo.“ [10]

1. Drag and drop - Ťahanie a pustenie ponúka spôsob priamej manipulácie s objektami. Z reálneho sveta sme zvyknutý pracovať s objekty priamo a presúvať ich. Preto je tento spôsob viac intuitívnejší.

2. Editovanie na mieste - Typicky existujú samostatné stránky alebo sekcie pre zobrazovanie a editovanie položky. Zvyčajne existuje tlačidlo editovať, ktoré je však vzdialené od samotnej položky, ktorú chceme zmeniť. Editovanie na mieste ponúka spôsob, ktorý je viac priamočiary.

Zaoberali sme sa obecnými doporučeniami a prvkami avšak to nám neposkytuje zodpovedanie otázok ako k čomu má slúžiť, čo obsahovať a ako ho vnímajú užívatelia. K tomu sa používajú rôzne postupy a techniky, aby sme predišli typickým problémom a vytvorili UI vhodné pre užívateľa.

Pozorovanie práce a kontextuálny rozhovor [4]

Pozorovanie práce a kontextuálny rozhovor sú techniky, ktoré sa vykonávajú v rámci prieskumu u užívateľov na zistenie ich potrieb užívateľov. Zakladajú sa na tom, že užívatelia nevedia ako vyriešiť ich problémy, často si ani neuvedomujú aké sú ich problémy. A aj keď vedia aké sú ich problémy tak ich zahrnú množstvom ďalších nepotrebných požiadavkov. Preto sa ich nemôžeme jednoducho spýtať, čo potrebujú.

Tieto techniky sa najprv snažia pochopiť, čo užívateľ robí a na základe toho navrhnúť vhodné riešenie. Hodia sa najmä v prípade ak sa produkt vytvára pre špecifickú skupinu užívateľov. Pri pozorovaní práce sledujeme, čo užívateľ vykonáva a v rámci kontextuálneho rozhovoru sa pýtame špecifické otázky týkajúce sa ich činnosti a snažíme sa vyhýbať otázkam na ich názor. Ako praktické sa ukazuje klásť tieto otázky už pri pozorovaní, pretože často nevieme čo presne užívateľ robí alebo prečo to robí. Samozrejme sa to nesmie preháňať, aby sme nenarušili bežný tok práce užívateľa.

Persóny [4]

Persóny nie sú reálny ľudia, ale hypotetické archetypy skutočných užívateľov, ktoré reprezentujú ich charakteristiky, potreby a ciele. K vytvorenie persón je potrebné spraviť prieskum u užívateľov napríklad pomocou pozorovania alebo kontextuálneho rozhovoru. Aj návrhári sú ľudia, ktorí sa dopúšťajú chýb ako je na seba orientovaný dizajn. Táto technika vytvára empatiu pre špecifických užívateľov a pomáha vytvoriť produkt, ktorý je viac zameraný na užívateľov.

Myslenie nahlas [7]

Tiež nazývaný hlasitý pohovor je jednoduchý spôsob testovania užívateľského rozhrania. Užívateľ je posadený pred aplikáciu, ktorú má preskúmať, taktiež môže mať zadané úlohy, ktoré má splniť. Počas činnosti užívateľ rozpráva nahlas svoje myšlienky. Tie sú pozorovateľom analyzované (prípadne nahrávané a následne analyzované), aby odhalili špecifické problémy pri používaní rozhrania. Je dôležité ubezpečiť užívateľa, že testujeme aplikáciu a nie jeho. A že by nemal šetriť alebo báť sa povedať akýkoľvek názor alebo pripomienky, pretože o to nám pri testovaní ide. Avšak podstatnejšie ako nájdenie problémov je ich napravenie. K riešeniu je obecne lepšie spraviť, čo najmenej je možné a vyhnúť sa tak veľkým úpravám.

A/B testovanie [7]

Pri tomto druhu testovania sa porovnávajú rôzne dizajny, návrhy alebo implementácie. Príkladom je vytvorenie rôzneho rozloženie prvkov. Pri subjektívnej metóde môže rozhodnúť zákazník alebo vývojár variantu, ktorú považuje za lepšiu.

Pri objektívnej metóde meriame ako pôsobia rôzne verzie na sledovanú (závislú) premennú. Pričom sme limitovaný atribútmi, ktoré sa dajú objektívne merať. Častým je meranie profitu, doba strávená na stránke, frekvencia použitia a iné. Malá zmena vo formulácií vety, zvýraznenie alebo presunutie môže mať významný efekt na tieto parametre. Často sa testuje aj viac ako dve varianty súčasne, čo je však časovo náročnejšie.

Užívateľské názory [4]

Najjednoduchší spôsob ako zistiť názor užívateľa je spýtať sa a to napríklad formou dotazníku alebo v rozhovore. Jedná o subjektívnu metódu, ktorú je možné použiť v akejkoľvek fázi vývoja avšak je treba byť vedomý chýb, ktorých sa ľudia dopúšťajú napríklad systematických chýb pri predstavovaní si budúcnosti tj. aké by to bolo ak by aplikácia obsahovala túto funkciu. Práve preto sa hodí skôr k neskorším fázam keď má užívateľ pred sebou niečo konkrétne.

2.4 Psychológia

Aj v predchádzajúcej časti bolo demonštrované, že psychológia človeka hrá dôležitú úlohu pri interakcií s počítačom. V tejto časti zameriam na psychológiou cieľov a motivácie, ktoré sú veľmi úzko späté so správou činností a cieľov projektov.

Stanovenie cieľov [6]

K tomu, aby sme dosahovali ciele projektov je najprv potrebné ciele stanoviť. Už samotné stanovanie cieľa má vplyv nato, či sa nám cieľ vôbec podarí dosiahnuť. Platí, že ciele, ktoré sú orientovaná na zisk sú ľahšie dosiahnuteľné ako ciele z obavy o stratu. Rovnako aj ciele, ktoré majú za cieľ získanie schopností (growth mindset) ako demonštrovanie (fixed mindset), ciele z vnútornej motivácie ako ciele zamerané na externé odmeny sú ľahšie dosiahnuteľné. Dosiahnutie cieľa závisí predovšetkým na očakávaniach úspechu a od toho sa odvíja oddanosť a záväzok k cieli. Avšak to môžeme ovplyvniť tým ako o cieli premýšľame a ako ho stanovíme.

1. Obývanie reality (dwelling) - Pri tomto spôsobe zvažujeme len „negatívne“ aspekty reality tj. to, že dosiahnutie bude vyžadovať prácu apod.

2. Dopriavanie si (indulging) - Naopak pri tomto spôsobe rozmýšľame len pozitívne aspekty budúcnosti, ktorú si prajeme.
3. Mentálne kontrastovanie - Najprv zvážime pozitívne a potom negatívne aspekty.

Mentálne kontrastovanie sa ukázalo ako najefektívnejší spôsob dosahovania cieľov a tí, ktorý použili tento postup dosahovali najlepšie výsledky oproti ostatným skupinám. Avšak v prípade, že sa pri kontrastovaní zvažovali najprv negatívne a potom pozitívne, tak sa výsledky skupiny zhoršili. Čiže aj drobnosti ako zmena v poradí môžu mať významný vplyv.

Obecne sa psychológia cieľov sa delí na stanovanie a ich nasledovanie. Pri nasledovaní sa potýkame s mnohými problémami a štúdie ukázali, že vytvorenie tzv. implementačného zámeru je efektívny spôsob ako sa s nimi vysporiadať. Implementačný zámer je tvrdenie vo forme `Ak nastane situácia, tak spravím...` a tým sa snažíme dopredu pripraviť na problémy, ktoré sa môžu vyskytnúť.

Motivácia [13]

Existuje niekoľko modelov ako vysvetliť ľudské správanie. Jedným z najznámejších modelov je hierarchia potrieb, ktorá bola vytvorená známym psychológom Abraham H. Maslow v roku 1943. Podľa tohoto modelu sa ľudia snažia uspokojiť svoje fyziologické a psychologické potreby.

1. Fyziologické potreby (vzduch, jedlo, voda, atď.)
2. Bezpečnosť (zdravie, finančná bezpečnosť a zaistenie práce, atď.)
3. Potreba patriť (láska, priateľstvo, rodina, spoločnosť, atď.)
4. Úcta (sebavedomie, úspechy, uznanie, atď.)
5. Seba-realizácia (kreativita, záľuby, morálka, atď.)

Avšak Daniel Pink tvrdí, že v súčasnosti je väčšina z týchto potrieb ľudí je viacmenej uspokojená. Preto sa do hry dostavajú viac vnútorné faktory pričom zdôrazňuje predovšetkým:

- Autonómia - Mať veci pod kontrolou a samostatne sa rozhodovať.
- Majstrovstvo - Zlepšovať si schopnosti a stávať sa expertom.
- Zmysel - Považovať to za zmysluplné a vidieť v tom väčší význam.

2.5 Gamifikácia

Táto sekcia čerpá z [12]. Samotné slovo gamifikácia sa rozšírilo len od roku 2010 a stalo sa populárnym trendom. Interpretácia tohoto nového pojmu sa stretla s rôznou interpretáciou a rôznym chápaním. V širšom zmysle ide o navrhovanie systémov, ktoré sa inšpirujú z hier. Definícia od Kevina Werbacha znie: *„Gamifikácia je spôsob použitia prvkov z hier a techník návrhu a dizajnu hier v inom kontexte ako pri hraní hier.“* V definícií sa nachádzajú tri kľúčové časti:

- Herné prvky
- Techniky návrhu a dizajnu hier

- Nie herný kontext

Herné prvky sú znova použiteľné elementy z hier. Nie hry celé, ale len malé kúsky, ktoré sa dajú opakovane použiť v rôznych situáciach. Medzi typické herné prvky patrí:

- Body
- Odznaky
- Levely
- Rebríčky

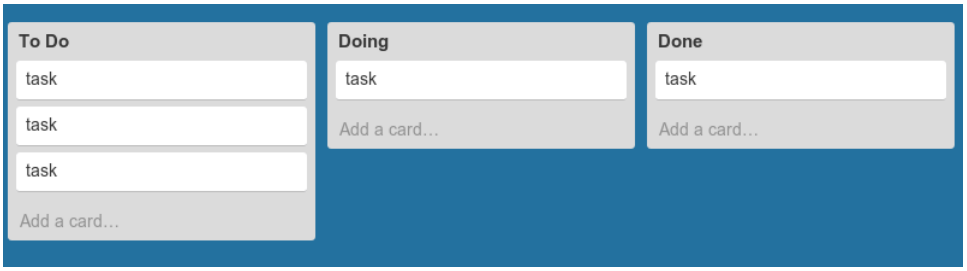
Techniky návrhu a dizajnu vyžadujú myslenie návrhára hier, ktoré zasadí prvky do kontextu (nie nutne herného), pridá im význam napríklad vo forme príbehu a pod. Táto oblasť je viac umenia ako veda, zahŕňa hudbu, dizajn, príbeh a celkovo umelecký zážitok. Poslednou časťou definície je nie herný kontext, čiže účelom je použitie mimo hier v reálnom živote. Gamifikácia nachádza použitie v biznise, na pracoviskách, školách a v aktivitách súvisiacich so zdravým životným štýlom. Tento zoznam však nie je úplný a gamifikácia preniká aj do iných oblastí. Na záver je však potrebné varovať, že nesprávne aplikovaná gamifikácia môže mať negatívny dopad. Preto treba zvážiť použitie a vhodnosť jednotlivých prvkov a techník.

2.6 Existujúce riešenia

Na trhu je veľké množstvo produktov, ktoré sa líšia zameraním na cieľovú skupinu, ktorou sú spoločnosti rôznych veľkostí alebo jednotlivci na osobný manažment. Poskytujú podporu rôznych zariadení a operačných systémov. V neposlednej rade sa líšia cenou, niektoré sú zadarmo, iné ponúkajú skúšobnú dobu s omedzenou funkcionalitou. Nasleduje zoznam nástrojov, ktoré majú podobný cieľ a sú použiteľné pre študentov. Dôraz bol kladený na rozmanitosť, nachádzajú sa tu jednoduché nástroje až po nástroje s plne integrovanou metodológiou.

Trello [7]

Trello je nástroj určený na spoluprácu na projektoch, ale je vhodný aj pre jednotlivcov. Organizuje projekty do tabúľ na ktorých je na prvý pohľad vidieť, čo je hotové, rozpracované a čo sa ešte musí dokončiť. Užívateľské rozhranie umožňuje rýchlu manipuláciu s objektami ťahaním a púšťaním položiek, väčšina operácií sa vykonáva. bez toho, aby sa stránka musela znova načítať.



Obrázek 2.2: Trello tabuľa - zoznamy s položkami

[7] https://trello.com/

Workflowy [8]

Hlavnou funkciou je vytváranie hierarchických zoznamov, ktoré je možné zdieľať. Položky zoznamov sa dajú označiť ako dokončené. Ďalšími užitočnými funkciami sú vyhľadávanie, pridávanie tagov a obľúbených stránok. Veľkou výhodou tohoto nástroja je jednoduchosť a rýchlosť ovládania. Každá operácia má klávesovú skratku, čo skúseným užívateľom veľmi urýchľuje ovládanie. Voľnosť, ktorá aplikácia ponúka ľahko spôsobí, že neskúsení užívatelia hierarchický zoznam s nesprávnou štruktúrou, ktorý je potom veľmi neprehľadný.

RedCritter [9]

Gamifikovaná aplikácia na manažovanie projektov pre firmy. Zahŕňa body, rebríčky a sadu odznakov s možnosťou pridávať vlastné. Účelom je hlavne zvýšenie produktivity a motivácie zamestnancov. Čomu prispieva aj obchod kde si zamestnanci za získané body kupujú veci. Záleží len na firme, čo im do obchodu vloží.

Scrumwise [10]

Scrumwise je veľmi intuitívna aplikácia založená na Scrum metodológií, obsahuje funkcie ako sú backlog, šprinty, Scrum tabuľu a spaľovacie diagramy. Zameraná je hlavne na spoločnosti keďže poskytuje vytváranie rôznych tímov a iné tímové funkcie. Aj napriek tomu je použiteľná aj jednotlivcami.

Google Kalendár [11]

Google kalendár je veľmi rozšírená služba, ktorá je určená predovšetkým k plánovaniu udalostí a stretnutí, ale aj úloh. Poskytuje mnohé nastavenia ako aj zadávať opakujúce sa udalosti, zdielať kalendáre a iné. Veľkou výhodou tejto služby je integrácia s ďalšími službami firmy Google.

[8] https://workflowy.com/
[9] http://www.redcrittertracker.com/
[10] https://www.scrumwise.com/
[11] https://www.google.com/calendar

Kapitola 3

Návrh riešenia

Popis existujúcich riešení 2.6 prezentuje široké spektrum oblastí zamerania v radách podobných aplikácií. Aby som prezentoval akým smerom sa orientuje táto práca definujem na začiatku kapitoly konkrétne ciele a tým vysvetlím ako zapadá do tohoto spektra. Následne predstavím návrh užívateľského rozhrania v ktorom budú tieto ciele zohľadnené a s pomocou prípadov použitia zhrniem, čo výsledná aplikácia poskytne užívateľovi. Ďalšou modelovacou technikou predstavím návrh dátovej vrstvy, ktorá je potrebná pre prácu s jednotlivými objektami aplikácie.

3.1 Definícia cieľov

Ako už názov práce napovedá cieľom je vytvoriť nástroj na správu činností a cieľov projektov, ktorý je určený najmä pre vysokoškolských študentov. Nápomocná im bude v rôznych oblastiach v ktorých budú ciele predstavené spolu s otázkami, ktoré je nutné pri návrhu zodpovedať k splneniu príslušného cieľa.

Stanovenie cieľov a plánovanie projektov

Bakalárske a diplomové práce predstavujú širokú škálu projektov a navyše sa predpokladá, že študenti budú chcieť využiť tento nástroj aj na správu školských projektov. Preto aplikácia bude mať zakomponované techniky efektívneho dosahovania cieľov a postupy z riadenia projektov, ktoré sú vhodné predovšetkým na charakter týchto projektov. Pomocou zakomponovaných techník a postupov bude môcť užívateľ stanoviť cieľ projektu a naplánovať projekt, čo znamená rozdeliť si ho na menšie časti ku ktorým sú priradené úlohy a míľniky. Pričom aj užívateľ, ktorý tieto postupy nepozná bude schopný aplikáciu používať.

1. Aké postupy, metodológie z riadenia projektov použiť? Ktoré sú najviac vhodné pre tieto projekty?
2. Ako sa budú stanovovať ciele a míľniky?
3. Ako sa bude vytvárať plán? Ako sa bude tento plán členiť, čo bude obsahovať a ako ho deliť na menšie časti?

Získavanie, organizovanie a zdieľanie informácií

Plánovanie sa nezaobíde bez potrebných informácií ako sú napríklad termíny. Aj napriek tomu, že informácie sú známe a dostupné, tak nebývajú prezentované v najjasnejšej

forme preto je potrebné zjednodušiť získanie, prehľadnosť a organizáciu informácií. Prvý pohľad na plán by mal poskytnúť dôležité informácie a zodpovedať špecifické otázky ako koľko mi ostáva času apod.

1. Ako zjednodušiť získavanie a zdieľanie informácií?
2. Ako organizovať informácie v projekte? Aké informácie zahrnúť? Ako ich prehľadne zobraziť?

Jednoduchosť a efektívnosť ovládania

V prvom rade užívatelia musia byť schopný sa rýchlo naučiť aplikáciu používať a ovládanie musí byť jednoduché a rýchle. K tomu prispejú prvky ako editovanie na mieste, ťahanie a pustenie a dodržiavanie obecných Nielsenovy heuristík. Pomocou testovania UI sa zistia prípadné problémy a nájdu lepšie riešenia.

1. Ako zjednodušiť ovládanie aplikácie? Bude užívateľ vedieť používať aplikácie bez nutnosti zaúčania?
2. Ktoré operácie bude užívateľ očakávať a ako ich bude chcieť ovládať?
3. Ktoré operácie bude najčastejšie používať a ako ich urýchliť?

Motivácia a spätná väzba

Motivácia a poskytnutie spätnej väzby pri činnostiach užívateľa je nepochybne dôležité pri vykonávaní projektu. Niekedy sú to jednoduché veci ako informovanie užívateľa o výsledku operácií až po použitie herných prvkov a iných prostriedkov gamifikácie.

1. Ako zvýšiť motiváciu užívateľov pracovať na projektoch?
2. Ktoré herné prvky použiť? Ako zakomponovať techniky návrhu hier?

3.2 Návrh užívateľského rozhrania

„Dobrý návrh sa nezačína obrázky. Začína snahou pochopiť užívateľa: čo majú radi, prečo daný systém používajú a prečo by s ním mohli interagovať. Čím viac o nich vieme, tým viac s nimi súcitíme, tým efektívnejšie môžeme pre nich navrhovať.“ [11]

Pochopenie užívateľa je základ pri návrhu zameraného na užívateľa. Preto som definoval základné charakteristiky, potreby a ciele užívateľov vo forme persóny, ktorá predstavuje výsledok pozorovaní a rozhovorov. Nasledujúca persóna by mala, čo najviac reprezentovať typického užívateľa.

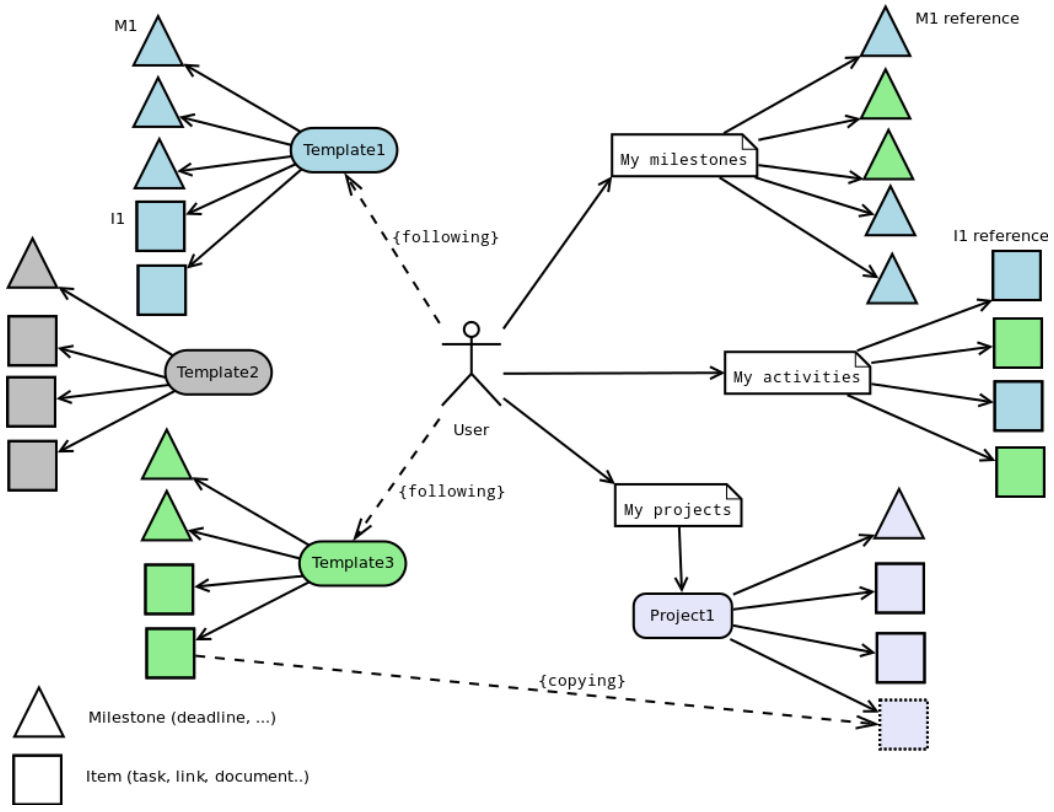
> Michal má 22 rokov. Je to vysokoškolský študent na fakulte informačních technologií Vysokého učení technického v Brně. Okrem školy si niekedy privyrába na brigádach, aby mal peniaze na jeho koníčky a záľuby. Najviac času však venuje škole, pretože dokončenie školy a získanie titulu je jeho prioritou.
>
> Projekty plánuje tak, že si najprv zistí základné informácie, aby sa rozhodol čomu je potrebné venovať pozornosť a čím začne. Často si vytvára zoznam termínov, ktorý má buď na papieri alebo v elektronickej podobe. Na začiatku projektu nezvykne rozmýšľať o osobnom cieli projektu a za ciel považuje len splnenie zadania. Začína hľadaním študijných materiálov. S kamarátmi často o projektoch komunikuje (osobne alebo online), aby získal rady, prípadne im tiež poradil. Vytváranie plánov považuje za náročné, preto väčšinou pracuje na projektu bez plánu a niekedy sa mu stáva, že na niečo zabudne alebo nestíha.

Čiže mnohí užívatelia začínajú zistením najbližšieho termínu alebo si dokonca napísali všetky termíny projektov. Taktiež vyhľadávali informácie a študijné materiály týkajúce sa projektu, ktorý riešili. Aby sme im zjednodušili získanie týchto informácií a každý užívateľ nemusel tieto údaje opakovane zadávať do aplikácie do svojich projektov, tak som navrhol tzv. šablóny.

Šablóna je obecný vzor projektu, ktorý obsahuje základné informácie, materiály, atď. podľa ktorých si užívatelia môžu vytvárať vlastné konkrétne inštancie projektu. Princíp činnosti je ilustrovaný na obrázku 3.1, na ľavej strane sa nachádzajú tri šablóny, ktoré v sebe obsahujú míľniky (trojuholníky) a položky (štvorce). Položky sú najčastejšie konkrétne úlohy, ale môže sa jednať napríklad o odkaz na študijný materiál.

Užívateľ môže začať sledovať šablónu pričom si vyberá len šablóny, ktoré ho zaujímajú. Keď užívateľ začne sledovať šablónu vznikne spojenie (following), ktoré zaistí, že všetky míľniky zo šablóny sú zobrazené v míľnikoch užívateľa zoradené od najbližšieho míľniku po najvzdialenejší. Užívateľ je tiež informovaný o činnosti na šablóne ako je pridávanie nových položiek, čo sa mu zobrazuje v jeho aktivitách.

Ďalšou operáciou (copying), ktorá je naznačená na obrázku je kopírovanie položky. Pri tejto operácií sa zadáva do akého projektu sa má položka umiestniť. Na základe toho sa na príslušnom mieste vytvorí nová kópia položky, ktorú si užívateľ môže ľubovoľne modifikovať.



Obrázek 3.1: Operácie - sledovanie šablóny a kopírovanie položky

Takže na rozdiel od sledovania, ktoré vytvára len odkaz na šablónu na základe ktorého sa mu sprístupňujú míľniky a aktivity, tak pri kopírovaní sa vytvára nový objekt bez referencie na objekt z ktorého bol vytvorený. Ďalší rozdiel je, že položky je možné kopírovať nie len zo šablóny, ale aj z konkrétnych projektov iných užívateľov.

Využitie budem demonštrovať na konkrétnom príklade študentov, ktorí pracujú na svojich bakalárskych prácach. Čiže v aplikácií existuje šablóna "Bakalárska práca", ktorú vytvoril vedúci týchto študentov. Do tejto šablóny im zadal priebežné míľniky a doporučené činnosti ako sú napríklad položky v prvom stĺpci tabuľky 3.1.

Názov položky	Schopnosti
Stanoviť si cieľ	Predstavivosť, plánovanie
Vytvoriť si plán	Plánovanie
Naučiť sa a používať verzovací systém GIT	GIT

Tabulka 3.1: Ukážka položiek so súvisiacimi schopnosťami

Študenti, ktorí túto šablónu sledujú sú informovaný o blížiacich sa míľnikoch a novinkách ako je pridanie novej položky. Kliknutím na položku si zobrazia podrobný popis a prípadne si ju skopírujú do svojho projektu. Študenti si navyše do svojich projektoch spracovávajú študijné materiály a navzájom si ich medzi sebou kopírujú.

K podporeniu záujmu vytvárať položky, ktoré sú hodnotné aj pre iných užívateľov som navrhol reputáciu. Keď niekto skopíruje položku do svojho projektu znamená to, že bola pre neho hodnotná a ten kto ju vytvoril získa reputáciu (bod).

Ďalší gamifikačný prvok, ktorý som použil sú schopnosti. Tento prvok je zameraný na vnútornú motiváciu keďže v kapitole o psychológií 2.4 bolo spomenuté, že vnútorná motivácia je dôležitejšia pri dosahovaní cieľov. Zlepšovanie schopností je navyše jedným pilierom modelu motivácie Daniela Pinka. Užívateľ si môže k jednotlivým položkám definovať schopnosti, ktoré daná položka zlepšuje. Keďže schopnosti je možno definovať aj na položkách šablóny využijem predchádzajúci príklad a v tabuľke 3.1 sú zobrazené ukážky schopnosti ku každej položke. V profile užívateľa sa zobrazuje aktuálny level schopností a koľko ostáva na dosiahnutie ďalšieho levelu v podobe progres baru. Zvolil som tieto levely:

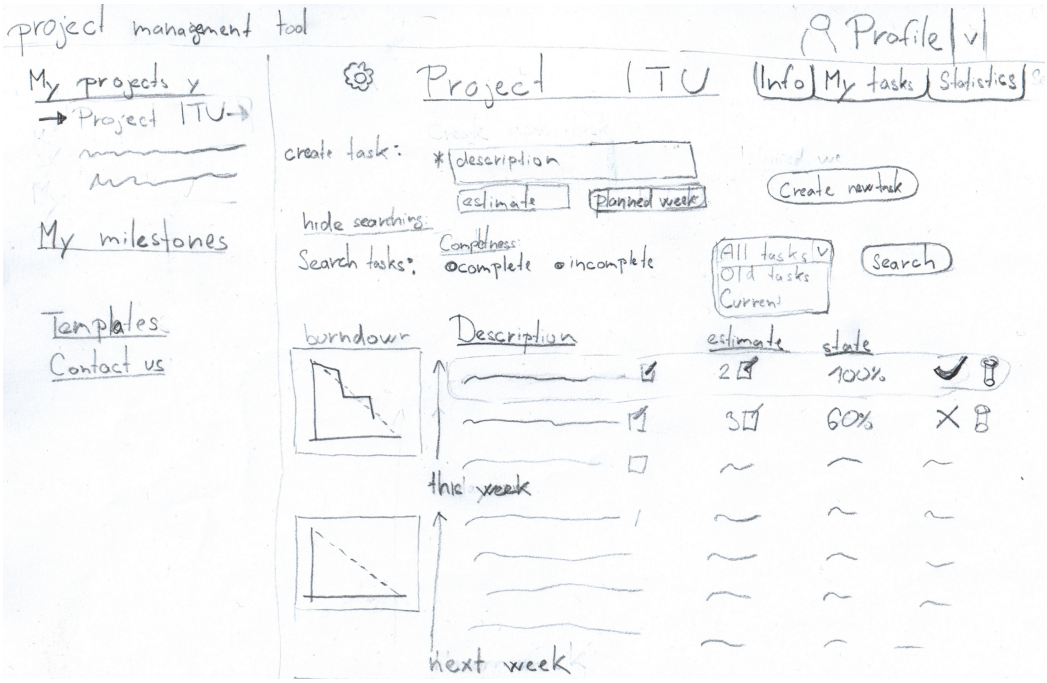
- Level 1 - Začiatočník (Beginner)
- Level 2 - Dobrý (Good)
- Level 3 - Mierne pokročilý (Intermediate)
- Level 4 - Pokročilí (Advanced)
- Level 5 - Výnimočný (Exceptional)
- Level 6 - Expert (Master)

Ďalší prvok, ktorý má podporiť odhodlanie pracovať na projektu je technika mentálneho kontrastovania a implementačných zámerov 3.1. Pri vytváraní nového projektu je nápoveda s inštrukciami a príkladom ako stanoviť cieľ.

> Čo je cieľom projektu a čo pozitívne ti práca a dosiahnutie cieľa prinesie? Aké prekážky a problémy budeš musieť prekonať? Popíš ako sa s nimi plánuješ vysporiadať vo forme: ak sa stane to, tak spravím to.
>
> Príklad: Chcem si zlepšiť schopnosti a vytvoriť niečo na čo môžem byť pyšný. Budem musieť čeliť nedostatku času vo vyťažených obdobiach a lenivosti. Ak sa objavia tieto problémy, tak si vyhradím čas ráno keď mám najviac energie a budem pracovať aspoň niekoľko desiatok minút.

Aj pri návrhu je vhodné postupovať štruktúrovane tj. začať s určitou úrovňou abstrakcie a postupne pridávať detaily, preto mojím ďalším krokom bolo vytvorenie náčrtov UI. Tým som si ujasnil základnú štruktúru a prvky UI a neriešil som farby, presný obsah a dizajn prvkov. Ceruzka, papier a guma, ktoré dovoľujú jednoducho experimentovať s prvkami rozhrania, ich umiestnením, veľkosťou atď. Pričom zmeniť náčrt je oproti zmene v skutočnej aplikácie oveľa jednoduchšie. V reálnej aplikácií by každá zmena mohla vyžadovať veľa ďalších závislých zmien.

„Fixovaním sa na konkrétne riešenie, ktoré máme v hlave alebo na náčrte nám môže uniknúť lepšie riešenie.“ [4] Avšak náčrty predstavujú len štartujúci bod z ktorého sa bude vychádzať a treba byť otvorený novým nápadom, ktoré sa objavia. Vzhľadom nato, že dôležitou súčasťou aplikácie je zobrazenie detailu projektu, tak obrázok 3.2 prezentuje náčrt práve tejto časti.



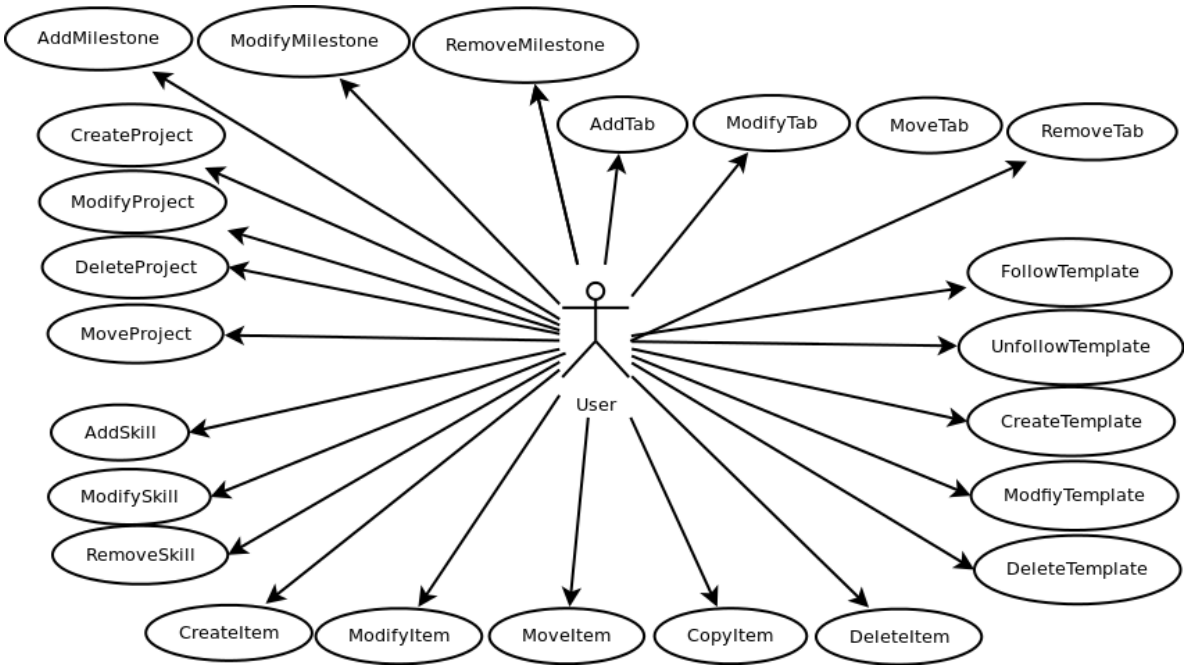
Obrázek 3.2: Nákres detailu projektu

Náčrt zobrazuje detail projektu so zoznamom úloh, ktoré je možné naplánovať do súčasného alebo nasledujúcich týždňov. Využíva sa odhadovanie úloh podľa Scrum metodológie, aby užívatelia lepšie odhadli dobu trvania, úlohy si rovnomerne rozložili a mohli sledovať svoj progres v spaľovacom grafe. Na základe tohoto náčrtu bola vytvorená prvá verzia aplikácie na obrázku 5.1, ktorá bola použitá na experimentmi z užívateľmi. Tento návrh sa však ukázal ako nepostačujúci z dôvodov, ktoré budú uvedené v príslušnej časti a preto bol použitý nový návrh inšpirovaný Kanban systémom, ktorý je vidieť na obrázku 5.3. Položky sú v tomto prípade rozdelené do tabov medzi ktorými sa môžu ľubovolne presúvať s využitím ťahania a púšťania. Taby je navyše možno neobmedzene vytvárať a tak si projekt prispôsobovať, organizovať a plánovať.

3.3 Prípady použitia

Prípady použitia reprezentujú jednotlivé akcie, ktoré môže užívateľ v aplikácií vykonať. Budú zhrnuté niektoré akcie, ktoré boli popísané už v rámci návrhu UI a taktiež budú predstavené ďalšie bežné akcie. Medzi najčastejšie akcie, ktoré môže užívateľ sú vytvorenie, modifikovanie, vymazanie objektu.

Neprihlásený užívateľ sa môže len prihlásiť, registrovať alebo vygenerovať náhodného užívateľa, aby mohol aplikáciu vyskúšať bez nutnosti registrácie. Na obrázku 3.3 sú zobrazené akcie pre prihláseného užívateľa pričom je naznačené zoskupenie k objektom. Pri popise budem postupovať podľa objektov ku ktorým sa jednotlivé prípady použitia vzťahujú.



Obrázek 3.3: Prípady použitia

- Šablóny (Template) - Prehľad detail šablón je prístupný každému na základe čoho si užívateľ vyberá, ktoré šablóny chce sledovať, prípadne prestať sledovať. Užívateľ môže vytvoriť novú šablónu, ktorú má potom pod kontrolou tj. môže modifikovať základné informácie ako je názov a popis alebo môže šablónu vymazať.
- Projekt (Project) - Každý užívateľ si vytvára vlastné projekty, ktoré si sám spravuje (vytvára, modifikuje a vymazáva). V závislosti na nastavení sú prístupné len užívateľovi samotnému alebo sú verejné. Aby si užívateľ mohol radiť projekty podľa dôležitosti je pridaná akcia na prehodenie poradia.
- Míľnik (Milestone) - K projektom a šablónam je možné pridávať tzv. míľniky, ktoré reprezentujú priebežne ciele a termíny. Vytvorené míľniky sa dajú upraviť, prípadne vymazať.
- Tab (List) - Informácie v projektoch a šablónach sú organizované v taboch, tie je možno neobmedzene pridávať, meniť a mazať a to v jednotlivých šablónach a projektoch.

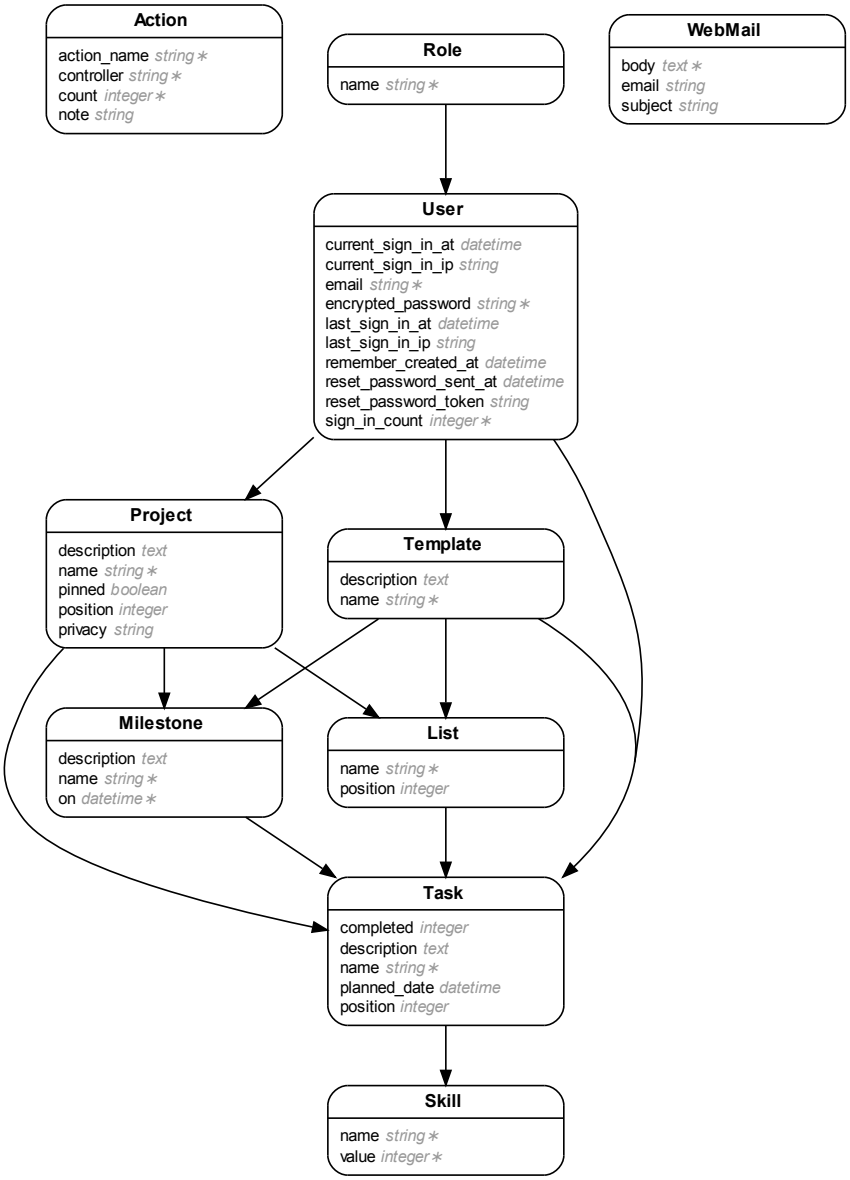
- Položka (Item) - Rovnako ako tab a míľnik je položka vytváraná, upravovaná a vymazaná v rámci projektu alebo šablóny. Položky sa dajú zo šablóny kopírovať, ale aj z verejných projektov iných užívateľov. V rámci tabu je možné položky prehadzovať a tak radiť podľa dôležitosti rovnako ako aj prehodiť položku do iných tabov.
- Schopnosť (Skill) - Schopnosti sú pridávané k jednotlivým položkám. Tie sa dajú upraviť alebo vymazať.

3.4 Návrh dátovej vrstvy

V predchádzajúcej sekcií boli spomenuté niektoré objekty s ktorými užívateľ môže pracovať. K tomu je potrebné tieto objekty navrhnúť (zvoliť vhodné atribúty, vzťahy apod.) a následne vhodným spôsobom reprezentovať (tým sa budem zaoberať až v realizácií 4.1). Teraz postupne popíšem dôležité charakteristiky jednotlivých objektov, ktoré sú tiež zobrazené v E-R (Entity-relationship) diagrame na obrázku 3.4.

- User - Užívateľ, ktorý má vlastné projekty a niekoľko šablón, ktoré vytvoril. To znamená, že k nim má vzťah 1:M. Každý užívateľ má priradenú určitú rolu. Dôležitými atribútmi sú email a heslo, ktoré slúžia na identifikáciu a reprezentáciu užívateľa, ostatné atribúty poskytujú predovšetkým štatistické údaje.
- Role - Rola slúži na zabezpečenie oprávnení pre rôzne typy užívateľov. Implicitná hodnota je neprihlásený užívateľ, ďalej sa rozlišuje prihlásený užívateľ a administrátor. Rola obsahuje jediný atribút a tým je názov role.
- Project - Projekt môže mať zároveň niekoľko míľnikov, položiek a zoznamov, čiže so všetkými má vzťah 1:M. Atribútmi projektu sú názov, popis, nastavenie súkromia a zobrazovania v bočnom menu. Ďalej obsahuje pozíciu, aby si mohol užívateľ zoradiť projekty podľa dôležitosti.
- Template - Šablóna je na tom rovnako ako projekt, čo sa týka asociácií tj. má vzťah 1:M k míľnikom, zoznamom a položkám. Avšak obsahuje len atribúty názov a popis.
- Milestone - Míľnik patrí buď do šablóny alebo do projektu. Míľnik má názov, popis a časový údaj do kedy sa má splniť.
- List - Zoznam položiek, ktorý je zobrazovaný ako tab. Má názov a pozíciu (kvôli zmene poradia) a patrí buď do projektu alebo šablóny.
- Task - Najčastejšie sa jedná o úlohy, ale reprezentuje obecné položky ako odkaz, atď. Položky môžu alebo nemusia patriť do zoznamu, ale vždy patria buď do projektu alebo šablóny. Položka má názov, popis, percento dokončenia a opäť pozíciu, aby sa dali úlohy v zoznamoch presúvať.
- Skill - Niekoľko schopností je možné pridať k položke. Pričom treba definovať hodnotu o ktorú dokončenie položky zlepšuje danú schopnosť.
- WebMail - Slúži na ukladanie zálohy emailov, ktoré môžu užívatelia odoslať zo stránky a tak posielať otázky, názory na stránku apod. Čiže obsahuje atribúty pre subjekt a text správy a email kam sa má odoslať prípadná odpoveď.

- Action - Tento objekt bol použitý pri experimente v časti 5.2 na ukladanie frekvencie operácií. Ukladá sa do neho názov akcie, kontrolér a počet vykonaní.



Obrázek 3.4: Dátová vrstva aplikácie - E-R diagram

Kapitola 4

Realizácia

V tejto kapitole vysvetlím spôsob realizácie s využitím technológií uvedených v 2.1. Na základe týchto technológií predovšetkým frameworku Ruby on Rails sa odráža aj štruktúra aplikácie. Pretože Ruby on Rails je framework, ktorý núti dodržiavať MVC architektúru. Pri popise implementácie sa budem tiež držať MVC architektúry a začnem najnižšou dátovou vrstvou, ktorá je základným stavebným kameňom aplikácie a postupne prejdem až k najvyššej vrstve, ktorá prezentuje užívateľovi výsledok. V jednotlivých vrstvách budú popísané súvisiace zdrojové súbory programu a dôležité knihovny. Všetky použité knihovny sú vymenované v súbore `Gemfile` na priloženom CD A. Ďalšia sekcia predstaví testmi riadené programovanie, ktoré súvisí s postupom pri realizácií a zároveň uvedie poslednú sekciu automatické testy.

4.1 Dátová vrstva (model)

Návrh dátovej vrstvy bol predstavený v 3.4 a nato aby sme s týmito objektmi mohli jednoducho manipulovať je potrebné ich vhodne reprezentovať. Ruby on Rails používa ORM 2.1 na mapovanie objektov aplikácie na databázové objekty. To znamená, že dátová vrstva bude reprezentovaná modelmi aplikácie. Ku každému z objektov v E-R diagrame 3.4 existuje v `app/models/` odpovedajúca trieda modelu s rovnakým názvom, vzťahmi, atribútmi a ich dátovými typmi.

Tieto databázové objekty nie len, že sú reprezentované aplikačným kódom, ale sú ním aj vytvorené s pomocou tzv. migrácií. Migrácie predstavujú spôsob ako vykonávať aditívne a reverzibilné zmeny databáze, ktoré sú databázovo nezávislé a opakovateľne spustiteľné napríklad v rôznych prostrediach. Ruby on Rails rozlišuje produkčné, testovacie a vývojové prostredie, ktoré majú odlišné nastavenia a pracujú nad odlišnými databázami. Na obrázku 4.1 je možné vidieť prehľad relevantných zdrojových súborov ako je aktuálne schéma databáze `schema.rb` a skript na inicializáciu `seeds.rb`.

Medzi modelmi aplikácie sa nachádza ešte trieda povolenie `ability.rb`, pretože využívam autorizačnú knihovna `cancan`, ktorá poskytuje funkcionalitu na zápis a overenie oprávnení pre jednotlivé role užívateľov. Autorizačné pravidlá sú zapísané na jednom mieste práve v tejto triede, čo zjednodušuje zmenu a pridávanie oprávnení. Na iných miestach aplikácie, hlavne v pohľadoch sú tieto pravidlá overované napríklad pri prístupe k objektu a na základe toho sa vykonáva príslušná akcia.

Autorizácia sa nezaobíde bez autentikácie tj. overenie, že užívateľ je ten za koho sa vydáva. K tomu využívam knihovnu `devise`, ktorá pozostáva s niekoľkých konfigurovateľných

```
app/                          # zdrojové súbory aplikácie
  |-- models/                 # triedy modelov
db/
  |-- migrate/                # migrácie databázi
  |-- schema.rb               # aktuálne schéma databáze
  |-- seeds.rb                # inicializácia databáze
  |-- development.sqlite3     # databáza vo vývojovom prostredií
  +-- test.sqlite3            # databáza vo testovaciom prostredií
```

Obrázek 4.1: Dátová vrstva - prehľad zdrojových súborov

modulov. V aplikácií využívam moduly na registráciu, prihlásenie a overenie užívateľa na základe zašifrovaného hesla v databázi, pamätanie si prihláseného užívateľa, obnovu hesla a sledovanie údajov o prehlásení ako je IP adresa apod.

Použil som aj ďalšie knihovny k zaisteniu niektorých typických funkcií s potrebou databázových objektov, ktoré si vytvárajú (migráciami) a poskytujú obecné riešenie ako bude popísané. Aby som oddelil časti, ktoré som vytváral a taktiež kvôli nadmernej veľkosti neboli tieto objekty zobrazené v návrhu dátovej vrstvy 3.4 avšak sú v dokumentácií na priloženom CD A sú prístupné aj kompletné schémy dátovej a riadiacej vrstvy.

- `acts as follower` - Knihovna využíva polymorfické asociácie, aby umožnila akémukoľvek modelu sledovať akýkoľvek iný model. Spojenie je pri tom zaznamenané modelom `follow.rb`. To som využil, aby som užívateľom umožnil sledovať projekty a šablóny. Zároveň to poskytuje možnosť jednoducho rozširovať aplikáciu v prípade potreby.

- `acts as commentable` - Akýkoľvek model je možno označiť ako komentovateľný, čo je zaznamenané modelom `comment.rb`. Opäť sa využíva polymorfizmu a tak poskytuje obecnejšie a ľahšie rozšíriteľné riešenie.

- `public activity` - Slúži na sledovanie a ukladanie noviniek v modeli `activity.rb`. Novinky sa vytvárajú pri akciách užívateľa ako je pridanie položky do šablóny, dokončenie úlohy.

- `active record reputation` - Je to reputačná knihovna, ktorá umožňuje definovať pravidlá pridávania reputácie. V aplikácií užívateľ získa reputáciu v prípade, že si niekto skopíruje položku, ktorú vytvoril.

4.2 Riadiacia vrstva (kontrolér)

Užívateľove operácie nad modelom sú riadené kontrolérom a typicky pre každý model existuje asociovaný kontrolér. Avšak v niektorých prípadoch to nie je potrebné a je dokonca výhodnejšie pracovať s viacerými modelmi súčasne napríklad úprava projektu môže zahrňovať zmenu míľniku. Tabuľka 4.1 zobrazuje aký model je s priamo asociovaný s akým kontrolérom. Niektoré z modelov nemajú vlastný kontrolér a pracuje sa s nimi len v rámci iných kontrolérov, čo je tiež ilustrované.

Medzi základné operácie s rôznymi objektami, ktoré užívateľ môže vykonať patrí vytvorenie, zobrazenie, úprava a vymazanie objektu tzv. CRUD (create, read, update, destroy) operácie. K zaisteniu týchto operácií existuje v kontroléroch 7 metód, ktoré sú popísané

Kontrolér (trieda)	Asociovaný model (trieda)	Pracuje aj s modelmi
ProjectsController	Project	List, Milestone
TemplatesController	Template	List, Milestone
UsersController	User	Milestone, Project, Activity
TasksController	Task	Project, Template
WebMailsController	WebMail	
FollowsController	Follow	Akýkoľvek model
CommentsController	Comment	Akýkoľvek model

Tabulka 4.1: Vzťahy medzi kontrolérmi a modelmi

v tabuľke 4.2. Čiže k tomu, aby užívateľ vytvoril nový objekt musí spustiť 2 metódy, najprv metódu `new`, ktorá mu zobrazí formulár na pridanie objektu. Odoslanie formulára tj. poslanie POST požiadavku spustí metódu `create`, ktorá vytvorí nový objekt.

HTTP sloveso	Metóda kontoléru	Popis
GET	index	zobraz zoznam objektov
GET	new	zobraz formulár na pridanie nového objektu
POST	create	vytvor nový objekt
GET	show	zobraz konkrétny objekt
GET	edit	zobraz formulár na editovanie objektu
PATCH / PUT	update	zmeň objekt
DELETE	destroy	vymaž objekt

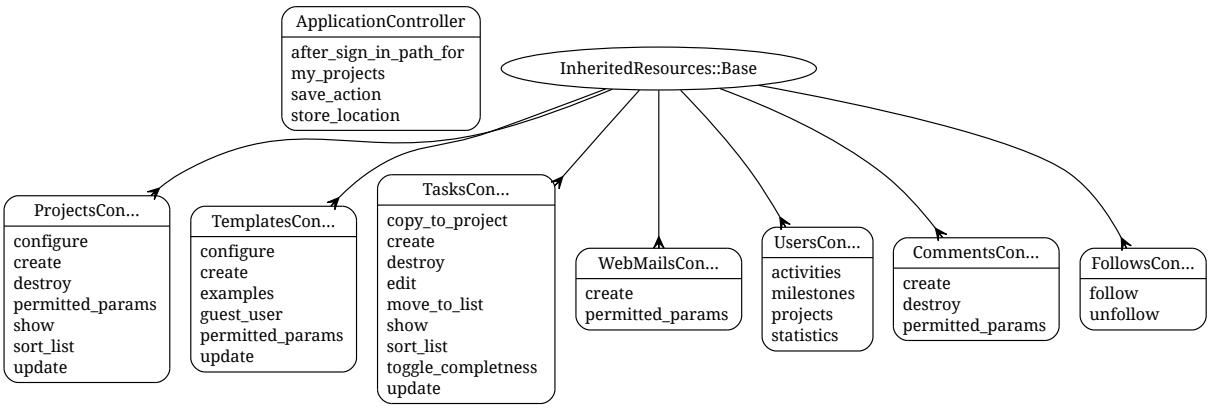
Tabulka 4.2: Popis akcií kontroléru na zaistenie CRUD operácií

Avšak tieto operácie spôsobujú duplikáciu kódu v kontroléroch, pretože ich štruktúra je stále rovnaká - načítaj model, vykonaj CRUD operáciu a zobraz pohľad. Mení sa len model s ktorým sa pracuje. Preto som využil knihovnu `inherited resources` na refaktorizáciu kódu kontrolérov, aby odpovedal konvencií štíhly kontrolér a tučný model. Podľa tejto konvencie by mal kontrolér obsahovať minimum kódu a predstavovať jednoduché rozhranie medzi modelom a pohľadom. Preto akákoľvek logika, ktorá nesúvisí s odpoveďou na spracovávaný požiadavok patrí do modelu. V diagrame tried na obrázku 4.2 je vidieť, že kontroléry aplikácie dedia z triedy knihovny `InheritedResources::Base`. Táto trieda poskytuje doménovo špecifický jazyk na dynamické pridanie CRUD operácií a ich konfiguráciu prepísaním (overriding) v jej potomkoch.

V diagrame tried v kontroléroch aplikácie je vidieť okrem CRUD metód aj iné metódy, ktorých popis je uvedený v tabuľke 4.3 s výnimkou metódy `permitted_params`.

Jedným z častých problémom Ruby on Rails aplikácií je umožnenie tzv. hromadného priradenia, čo znamená že akýkoľvek atribút, ktorý je odoslaný v POST požiadavku sa uloží. To predstavuje závažné bezpečnostné riziko, ak by sa napríklad niekomu podarilo priradiť id administrátora ku svojmu účtu a mohol by napáchať obrovské škody. Preto používam knihovnu `strong parameters` na vymenovanie povolených atribútov, tie sú vypísané v metóde `permitted_params`. Takže to, čo nie je explicitne dovolené je zakázané, pretože opačne je to náchylné k zabudnutiu.

V diagrame je ešte zobrazený `ApplicationController` v ktorom sú definované metódy `after_sign_in_path_for store_location`, ktoré zaistia, že užívateľ je po prihlásení presmerovaný na stránku, ktorú naposledy navštívil. Metóda `my_projects` je spúšťaná pri každom zobrazený stránky na zobrazenie užívateľových projektov v bočnom menu. Metóda



Obrázek 4.2: Riadiacia vrstva - Diagram tried

Kontrolér (trieda)	Metóda kontroléru	Popis
FollowsController	follow	začni sledovať projekt alebo šablónu
	unfollow	prestaň sledovať projekt alebo šablónu
UsersController	projects	zobraz užívateľove projekty
	milestones	zobraz míľniky zo šablón a projektov
	activities	zobraz užívateľove novinky
TasksController	sort_list	zmeň poradie úloh
	copy_to_project	skopíruj úlohu
	toggle_completness	označ úlohu ako dokončenú
ProjectsController	sort_list	zmeň poradie projektov
	configure	pridaj, uprav, vymaž zoznam
TemplatesController	guest_user	vygeneruj šablónu náhodného užívateľa
	configure	pridaj, uprav, vymaž zoznam
	examples	ukážky plánov

Tabulka 4.3: Metódy kontrolérov (okrem CRUD)

`save_action` sa spúšťa po každom zobrazení stránky a zaznamenáva aká akcia a akého kontroléru bola vykonaná, čo sa využilo na zlepšenie efektivity rozhrania 5.2. V zdrojových súboroch sa nachádza ešte kontrolér `Users::RegistrationsController` v ktorom sa konfigurujú nastavenia knihovny `devise` v rámci tejto vrstvy. S kontrolérmi súvisia ešte mailery 4.3, ktorých metódy sú využívané v rámci kontrolérov. V tomto prípade existuje len jedna metóda `contact_us` maileru `UserMailer`, ktorá odošle mail zo stránky.

```
config/routes.rb        # routovacie pravidlá
app/                    # zdrojové súbory aplikácie
  |-- controllers/      # kontroléry
  +-- mailers/          # mailery
```

Obrázek 4.3: Riadiacia vrstva - prehľad zdrojových súborov

```
app/                    # zdrojové súbory aplikácie
  |-- assets/           # súbory, ktoré vyžadujú prekompiláciu
    |-- javascripts/    # javascriptové zdrojové súbory
    +-- stylesheets/    # štýly
  |-- helpers/          # obsahuje pomocné funkcie opakujúce sa v pohľadoch
  |-- inputs/           # textové políčka na výber dátumu a času
  +-- views/            # jednotlivé pohľady rozdelené podľa objektov
    |-- layouts/        # obecné pohľady bez konkrétneho objektu
    |-- projects/       # pohľady súvisiace s projektom
    +-- ...             # pohľady súvisiace s ďalšími objektmi
public/                 # verejne prístupné súbory
  |-- images/           # obrázky
  |-- robots.txt        # informácie pre robotov indexujúcich stránku
  +-- ...               # favicon ikona a chybové hlášky
```

Obrázek 4.4: Vrstva pohľadu - prehľad zdrojových súborov

4.3 Prezentačná vrstva (pohľad)

Ako bolo spomínané táto vrstva sprístupňuje užívateľovi dáta vo webového rozhraní. Na obrázku 4.4 je prehľad súborov, ktoré sa podieľajú na zobrazení. V priečinku `assets/` sa jedná Javascript a SASS zdrojové súbory. Najdôležitejší priečinok je `views/`, ktorý obsahuje pohľady alebo ich časti rozdelené do priečinkov podľa objektu, ktorého sa týkajú. Nachádzajú sa tu rôzne typy súborov:

- súbory s príponou `.html.erb` - sú to HTML dokumenty s vnoreným Ruby kódom. To umožní použiť programové konštrukcie ako sú cykly a podmienky, ktoré sú spracované na strane serveru a klient dostáva html dokument. Súbor s pohľadom je pomenovaný rovnako ako metóda kontroléru, ktorá pohľad zobrazuje, čo je navyše toto je implicitná hodnota. Výnimku predstavuje súbor `app/views/layouts/application.html.erb`, ktorý obsahuje layout aplikácie v rámci ktorého sú zobrazované ostatné pohľady. Obsahuje hlavičku, bočné menu, pätičku a tým pádom nie sú tieto časti niekoľko násobne duplikované.

- súbory s príponou `.js.erb` - Rovnako ako keď kontrolér vráti HTML dokument ako odpoveď bežného požiadavku, tak v prípade ajax požiadavku sa ako odpoveď vykoná javascript v týchto súboroch. Pretože pri ajax požiadavku sú dáta odoslané na server bez toho, aby sa dokument musel znova načítať. Dodržuje sa rovnaká konvencia s pomenovaním akcií, odlišná je len prípona. Prípona .erb opäť dovolí vložiť serverový kód s výsledkom ajax požiadavku.

- súbory začínajúce podtržítkom - Sú to malé časti pohľadu tzv. parciály, ktoré sa opakovane objavujú na viacerých miestach. Parciály je možné do seba ľubovolne vnorovať a tak vytvárať rôznu hierarchiu a dokonca aj rekurzia. Zaistí sa tým znovapoužiteľnosť týchto menších častí. Napríklad parciála `projects/_list` zobrazí zoznam míľnikov, ktorý je zobrazovaný s rôznymi dátami z viacerých odlišných pohľadov a to z detailu projektu, detailu šablóny a zo zoznamu všetkých užívateľových míľnikov. Tabuľka 4.4 zobrazuje význam aj zvyšných parciál.

Adresár pohľadu	Parciála	Popis
comments/	_list	zobrazí zoznam komentárov
	_comment	zobrazí komentár
layouts/	_edit_resource_title	nadpis pri editovaní objektu
	_follows_links	odkazy na začatie alebo zastavenie sledovania
	_messages	flash správy oznamujúce výsledok operácií
	_resource_title	nadpis objektu s ikonami na editovanie, vymazanie
	_resource_errors	výpis chýb objektu
	_configure_form	úprava tabou
tasks/	_tabbed_lists	úlohy v taboch
	_tab_content	obsah tabu
	_task	jedna položka zoznamu s menu

Tabulka 4.4: Popis parciálov

Vkladanie serverového kódu často nesie so sebou vykonávanie podobných činností, ktoré je možné extrahovať do funkcií helperov. V aplikácií sa nachádza helper `SkillsHelper` v ktorom sú pomocné funkcie týkajúce sa schopností ako je zobrazovanie percent k dosiahnutiu ďalšieho levelu. V helperi `ApplicationHelper` sú pomocné funkcie využívané na rôznych miestach aplikácie ako je zobrazenie textu s klikateľnými odkazmi.

Interaktivitu zobrazovaných pohľadov som zvýšil CoffeeScriptovým kódom a knihovnou Jquery, preto uvediem rôzne dynamické prvky, ktoré som zakomponoval do stránky.

- Ťahanie a pustenie - S rôznymi objektami je možno priamo manipulovať a presúvať ich. K tomu bolo potrebné najprv definovať, ktoré objekty je možné ťahať a tiež ich patrične zvýrazniť (ťahací kurzor). Potom pridať rôzne udalosti, ktoré môžu pri ťahaní nastať ako presunutie sa nad a preč z oblasti a samozrejme, čo sa má vykonať v prípade pustenia. Napríklad tab sa zvýrazní keď je nad ním ťahaná položka a prestane byť zvýraznený keď sa položka presunie preč. Pustenie položky na tab vyvolá odoslanie ajax požiadavku na zmenu tabu položky.

- Pamätanie naposledy otvoreného tabu - Aby sa pri zobrazení detailu projektu zobrazil tab s ktorým sa naposledy pracovalo a nie prvý tab (čo je užívateľsky príjemnejšie) bolo potrebné využiť cookies. To sú textové súbory uložené u klienta na ukladanie malého množstva informácií. Ku každej zmene tabu sa zapíše, prípadne aktualizuje hodnota tabu pre príslušný projekt alebo šablónu.

- Klávesové skratky - Využíl som knihovnu `mousetrap-rails` k definovaniu klávesových skratiek, ktoré sú priradené rôznym akciám ako je zmena tabu zobrazenie detailu projektu.

- Zobrazovanie a skrývanie - Aby sme na stránke zobrazili viacej dôležitých informácií a užívateľ, tak niektoré časti stránky sú skryté. Kliknutím si však podrobnosti môže zobraziť, čo využívam napríklad na schovanie formulára na pridávania položky alebo na zobrazenie komentárov. Potrebné je k udalosti kliknutia priradiť príslušnú akciu a v prípade, že chceme aktuálny stav po znovu načítaní stránku, tak je aktuálny stav potrebné uložiť do cookies.

- Editovanie na mieste - K umožneniu editovania na mieste je potrebné pridať príslušné udalosti k čomu som využil knihovnu `best in place`, ktorá umožní zmenu a posiela

ajax požiadavky. Aby boli požiadavky spracované som vytvoril metódy, ktoré ich prijímajú a zasielajú odpovede.

4.4 Testmi riadené programovanie

Táto časť čerpá z [8]. Testmi riadené programovanie je postup pri ktorom akúkoľvek zmenu programu predchádza vytvorenie automatického testu, ktorý deterministický overí správnosť vykonaných zmien. Navyše aj po tom, čo sa podarí korektne zmeniť program sa pokračuje refaktorizáciou tj. zlepšenie kvality kódu.

Tento postup umožňuje odhaliť chybu skôr, čo uľahčuje hľadanie a odstránenie chyby, pretože tým ako sa program rozrastá sa odstránenie chýb stáva náročnejšie. Taktiež sa dosahuje väčšie pokrytie zdrojového kódu blížiace sa 100% percentám, čo je jediný objektívny ukazateľ kvality testovania. Ďalšou výhodou je, že testy je možné opakovane spúšťať, čo pomáha odhaliť chyby, ktoré vznikajú pri ďalšom vývoji.

Avšak aj testmi riadené programovanie má svoje nevýhody ako napríklad, že počiatočné vytvorenie testov je časovo náročné a rovnako ako pri programovaní aj pri vytváraní testov vznikajú chyby, ale existujú opatrenia ako im predchádzať. Avšak niekedy sú chyby zapríčinené rozdielmi medzi testovacom prostredí a vývojového alebo produkčného prostredia. Testmi riadené programovanie tiež vyžaduje osvojiť ďalšie programové nástroje ako aj iný spôsob myslenia. Niekedy je však nutné začať bez testu predovšetkým ak nemáme predstavu, čo a ako treba otestovať. V tom prípade je vhodné testy dopísať na záver.

Pri realizácií som postupoval podľa testmi riadeného programovania konkrétne podľa cyklu červená, zelená a refaktorizácia.

- Začínal som napísaním automatického testu, ktorý som okamžite spustil, aby zlyhal (červená). Keďže aj test je možné považovať za kód (navyše neotestovaný), tak v týmto opatrením sa odhalí chyba ak test úspešne prejde. Avšak samotná červená neznamená, že test neobsahuje žiadnu chybu. Test mohol zlyhať, ale nekontrolovať správne to, čo bolo zamýšľané.

- Následne som naprogramoval minimálne množstvo kódu, aby test úspešne prešiel. Pričom som používal práve vytvorené automatické testy.

- Testovanie však nie je zárukou kvality kódu a samotné testovanie nezlepšuje kvalitu kódu. Preto mojím posledným krokom je refaktorizácia pri ktorej som upravil práve naprogramovaný kód, aby som zlepšil čitateľnosť, odstránil duplikáciu apod. Opakovaním spustením testu som overil, či som pri refaktorizácií nezaniesol do programu chybu.

4.5 Automatické testy

Testovanie zodpovedá štruktúre aplikácie Ruby on Rails ako je vidieť na obrázku 4.5. Na testovanie som použil nástroj `Rspec`, čo je doménovo špecifický jazyk na testmi riadené programovanie. Knihovnou `factory girl` vytváram testovacie dáta, ktorá predstavujú minimálne množstvo validných dát a tie sú upravované podľa potrieb špecifického testu.

Pre každý zdrojový súbor modelu v aplikácií existuje korešpondujúci súbor s testami. Testy sa označujú ako unit testy, pretože overujú správnosť najmenších častí aplikácie, ktorými sú metódy modelu. Tie sú ďalej využívané vyššími vrstvami ako kontrolér a pri

```
spec/                       # priečinok s automatickými testmi
|-- controllers             # testy kontrolérov
|-- coverage                # pokrytie aplikácie a jednotlivých súborov
|-- factories               # testovacie dáta
|-- features                # integračné testy
|-- helpers                 # testy helperov
|-- mailers                 # testy maileru
|-- metrics                 # výsledky analýzy kódu
|-- models                  # testy modelov
|-- routing                 # testy routovacích pravidiel
|-- spec_helper.rb          # nastavenia testov a pomocné funkcie
+-- views                   # testy pohľadov
```

Obrázek 4.5: Automatické testy - prehľad zdrojových súborov

zobrazovaní, práve preto bola týmto testom venovaná najväčšia pozornosť. Rovnako aj pre každý kontrolér, mailer, helper existuje korešpondujúci súbor s testami, ktorý tiež testuje ich metódy. Avšak charakter a úlohy týchto metód sa vždy líšia.

Testy pohľadov, helperov a routovacích pravidiel som vytváral len v obmedzenej miere, pretože tieto časti sú tiež pokryté v rámci integračných testov. Integračné testy kontrolujú interakciu skupiny kontrolérov, pričom ide o testovanie čiernej skrinky pri ktorej nemáme znalosť o častiach alebo štruktúre aplikácie a testujeme len celkové správanie. To znamená, že funkciami knihovni `Capybara` simulujem akcie užívateľa vo webovom prehliadači.

Ďalšia knihovna, ktorú som použil je `rails_best_practices`, ktorá analyzuje kód za účelom odhalenia typických chýb a poskytnutia doporučení na zlepšenie kvality kódu. Na základe výstupu som upravil zdrojový kód aplikácie. Aj napriek tomu, že sa jedná o najlepšie praktiky, tak to neznamená, že sú najlepšie pre konkrétnu situáciu v aplikácií. Preto som najprv zvážil, či je doporučenie skutočne vhodné a niektoré doporučenia vynechal.

Ďalšou nepochybne dôležitou metrikou je pokrytie zdrojové kódu testami, čo je dôležitý ukazateľ kvality testovania. Pokrytie serverových častí som zmeral nástrojom `simplecov` a výsledok pre jednotlivé serverové komponenty ako aj celkový výsledok zobrazený v tabuľke 4.5.

Komponenty	Pokrytie zdrojového kódu testami
Modely	92,86%
Kontroléry	92,86%
Helpery	87,5%
Mailery	100%
Spolu	91,52%

Tabulka 4.5: Pokrytie serverového zdrojového kódu testami

Kapitola 5

Experimenty a vyhodnotenie

Aj keď sa jedná o záverečnú kapitolu, tak testovanie UI užívateľmi bolo vykonávané už v priebehu vývoja a zameriaval som sa na rôzne aspekty rozhrania. V tejto kapitole popíšem jednotlivé spôsoby testovania UI a ako ovplyvnili vývoj rozhrania.

5.1 Myslenie nahlas

Obecný postup tejto technike bol spomenutý v 2.3, teraz popíšem konkrétne údaje a výsledky, ktoré sa pri tomto testovaní zistili. Pri testovaní som postupoval podľa testovacieho protokolu (prevzatý z [2]), ktorý obsahuje predstavenie techniky užívateľovi a popis úloh, ktoré má vykonať. Prvou úlohou bol vytvoriť a naplánovať projekt. Ďalšími úlohami bolo zistenie informácií zo šablóny a v prípade druhej iterácie aj sledovanie šablóny a kopírovanie položky.

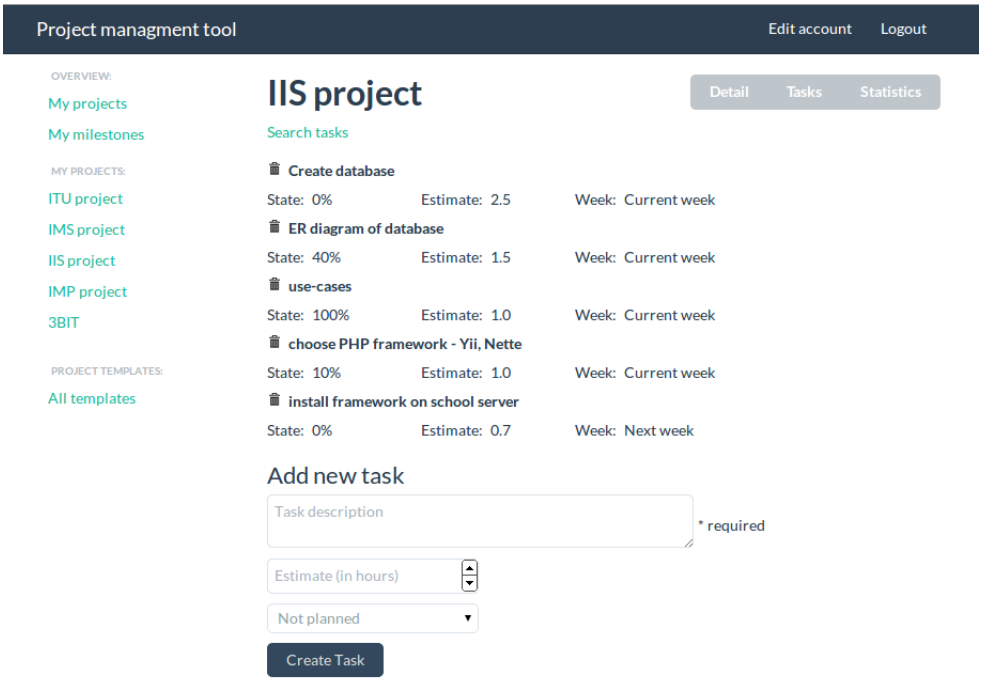
„Najlepšie výsledky prichádzajú z testovania UI s nie viac ako 5 užívateľmi. Väčší počet užívateľov objavuje rovnaké chyby a preto je lepšie najprv tieto chyby odstrániť a testovanie zopakovať.“ [5]

Preto som zvolil 5 užívateľov a vykonal som teda 2 iterácie tohoto testovania, ktoré oddelene popíšem. Všetci testovaní užívatelia boli študenti vysokej školy, ktorí sa považovali za počítačovo skúsených.

1. Počas prvej iterácie som testoval verziu aplikácie na obrázku 5.1, ktorý zobrazuje zoznam úloh (sekcia „Tasks“ v pravom hornom rohu), ktoré je možné naplánovať do nasledujúcich týždňov. Jednotlivé úlohy sú odhadované na základe čoho sa v sekcií „Statistics“ zobrazuje progres vo forme spaľovacieho grafu 2.1 pre jednotlivé týždne. V sekcií detail sú základné informácie a míľniky projektu.

   Pri testovaní sa objavilo mnoho chýb v rozhraní, väčšina z nich však bola očakávaná, pretože sa aplikácia nachádzala v skoršej fázi vývoja. Avšak zaradenie užívateľa už v tejto fázi odhalilo nečakané problémy a bolo veľmi hodnotné pre ďalšie smerovanie pri vývoji rozhrania.

   Najzávažnejšie problémy, ktoré sa objavili sa týkali zrozumiteľnosti a použiteľnosti. Užívatelia mali problém pochopiť štatistiky bez mojej pomoci (pritom obsahovali nápovedu) a navyše to neodpovedalo spôsobu akým pracujú. Buď požadovali rôzne nastavenia alebo vôbec neprejavili záujem meranie progresu používať. Požadovali zadávanie voľných dní alebo plánovanie po iných časových úsekoch a na dlhšiu dobu.



Obrázek 5.1: Verzia projektu pri prvej iterácií testovania

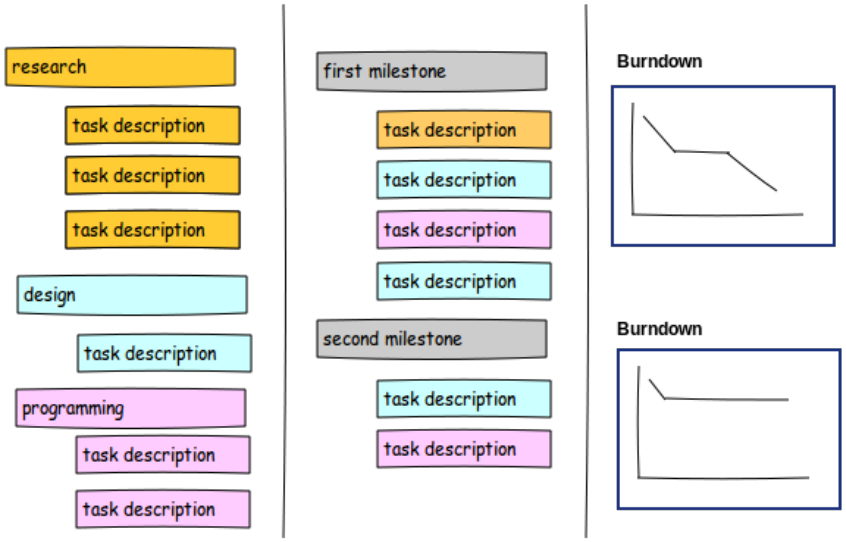
Taktiež chýbala možnosť mať úlohy rozdelené do kategórií. Do tejto verzie som pridal len vyhľadávanie a potvrdili sa moje očakávanie, že vyhľadávanie nestačí.

Avšak testovanie UI nie o nachádzaní, ale o ich odstránení. Na ich odstránenie som vytvoril 2 návrhy: prvý návrh 5.2 poskytoval rozširujúce funkcie a nastavenie dokonca aj niektoré, ktoré neboli požadované a druhý návrh 5.3 naopak zjednodušoval už existujúce riešenie a pridával minimálne množstvo potrebnej funkcionality.

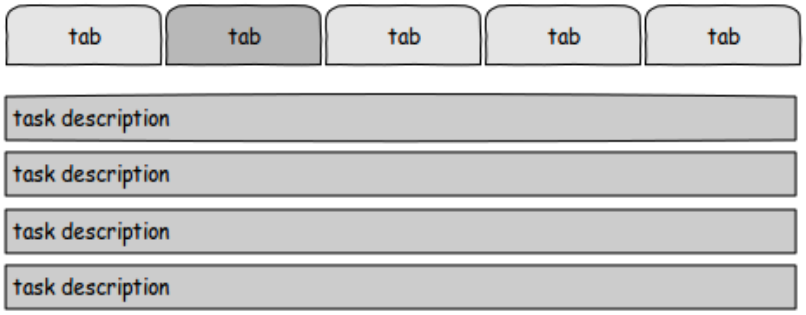
Uvážil som však, že použitie odhadovania a merania progresu spaľovacími diagramy nie je vhodné, pretože študenti sa ľubovolne prepínajú medzi projektami. To predstavuje problém, pretože progres stále „beží“ a tým sa stáva neaktuálny. Navyše pridávanie nastavení by ešte viac zkomplikovalo rozhranie a pravdepodobne by to ani nepomohlo zvýšiť záujem. Preto som sa rozhodol použiť druhý návrh s ktorým sa užívatelia často stretávajú a tým pádom by nemali vzniknúť problémy s pochopením.

2. Druhá iterácia sa vykonávala na verzií aplikácií, ktorá bola inšpirovaná Kanban systémom 5.3 a obsahovala tiež implementované operácie so šablónami. Pri tomto testovaní sa už neobjavili žiadne závažné problémy. Práve naopak bolo vidieť veľké zlepšenie pochopenia aplikácie rovnako ako aj v záujem používať aplikáciu. Výsledkom testovania bolo objavenie niekoľko menších špecifických chýb rozhrania.

   - Chýbalo zadávanie dátumu a času do kedy je úlohu treba vykonať. Toto bola funkcia, ktorú hľadali hneď 3 užívatelia, čo aplikácia neumožňovala. Avšak dvaja z nich použili namiesto toho míľniky, číže len jeden ostal nespokojný. K vyriešeniu problému stačilo pridať stĺpec s dátumom a časom do kedy sa má vykonať.
   - Užívatelia si mýlili taby s míľniky alebo nevedeli k čomu ich použiť. Pridal som teda príklady a nápovedy a taktiež som zmenil počiatočné nastavenie projektu,



Obrázek 5.2: Návrh UI inšpirovaný Scrum metodológiou



Obrázek 5.3: Návrh UI inšpirovaný Kanban systémom

aby užívateľ nezačínal od nuly a upravil si ich až keď sa s aplikáciou lepšie zoznámi.

- Užívatelia skúšali rovnako ako úlohy prehadzovať ťahaním aj taby, čo však nebolo podporované. Každopádne si všimli, že sa to dá vykonať na odlišnej stránke. Pre zaistenie konzistencie som však túto funkcionalitu pridal.
- Pri zobrazení menu poslednej položky sa stalo, že sa zobrazila skrolovacia lišta. Užívateľ si to nevšimol a ostal zmätený z toho, že nevidí všetky položky. Nastavil som preto, aby sa zobrazovalo menu bez skrolovacej lišty nad ďalšími prvkami stránky.
- Ukázalo sa, že mnohí užívatelia skúšali používať klávesové skratky a rýchlejšie ovládať rozhranie. K tomu som nastavil klávesové skratky, viac o tom je popísané v ďalšej časti.

5.2 Efektivita rozhrania

Podľa pravidla 80/20 by malo platiť, že 80% času užívatelia vykonávajú 20% akcií. Preto, aby bolo užívateľské rozhranie efektívne je potrebné sa venovať hlavne najčastejším akciám. To znamená identifikovať ich a prispôsobiť rozhranie, aby tieto akcie boli jednoduchšie a rýchlejšie. Pri prvotnom návrhu však nie sú k dispozícií údaje o frekvencií akcií a preto mi nezostávalo nič iné ako ich odhadnúť. Avšak odhad nemusí odpovedať skutočnosti preto som sa rozhodol zmerať frekvenciu používaných akcií.

Názov akcie	Počet vykonaní	Frekvencia (%)
Prehoď úlohu	479	15,28
Zobraz projekt	475	15,16
Zobraz moje projekty	319	10,18
Zobraz šablóny	276	8,81
Uprav úlohu	209	6,67
Zmeň úlohu	204	6,51
Zobraz šablónu	173	5,52
Editovať úlohu	121	3,86
Uprav taby	89	2,84
Zobraz profil	84	2,68
Označ úlohu ako hotovú	78	2,49

Tabulka 5.1: Frekvencia vykonávaných akcií

Po dobu 30 dní sa zaznamenávali sa do databáze akcie celkovo 24 užívateľov. Zaznamenané výsledky odpovedajú pravidlu a skutočne 23,8% zo všetkých akcií, ktoré sú zobrazené v tabuľke 5.1 tvorí 80% z celkového počtu vykonaných akcií užívateľov. Na základe týchto údajov som upravil rozhranie ako napríklad zmenil poradie položiek menu a priradil im odpovedajúce klávesové skratky a taktiež som urýchlil vykonávanie niektorých operácií s použitým ajaxu.

5.3 Užívateľské názory

„Užívatelia hodnotia produkty primárne podľa 3 dimenzií, ktorými sú použiteľnosť, spokojnosť a jednoduchosť používania.“ [3]

Pri tomto experimente sa aplikácia predstavila 5 ľudom, aby ju používali a po uplynutí 2 týždňov vyplnili dotazník, ktorý bol zameraný na práve tieto tri dimenzie. Užívatelia boli dotazovaní, aby odpovedali na nasledujúce otázky v 10 bodovej stupnici od vôbec nesúhlasí až až po úplne súhlasí.

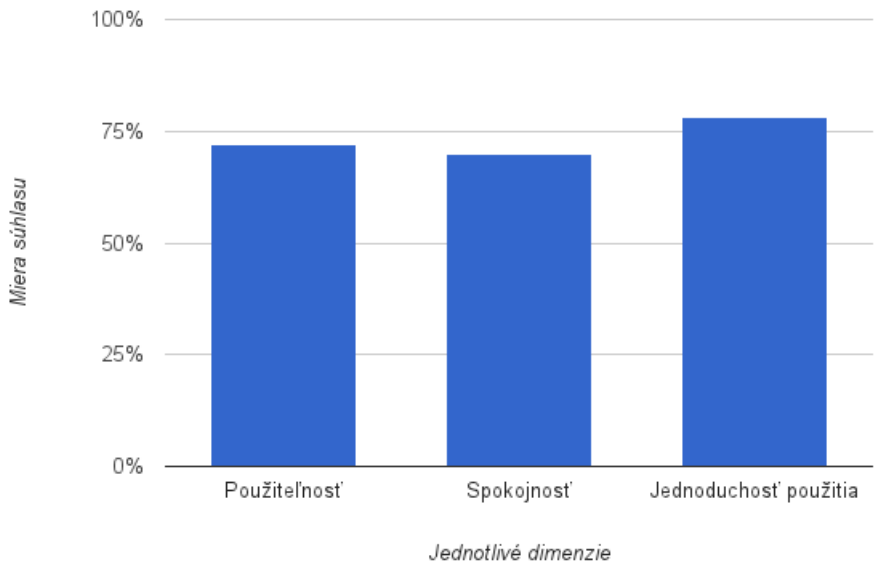
- Ako nápomocné je podľa Vás používanie webovej stránky?
- Ako jednoduché je aplikáciu pochopiť a používať?
- Aká je Vaša celková spokojnosť s tým, čo webová stránka poskytuje?

Výsledky sú zobrazené na obrázku 5.4 a ukazujú, že miera súhlasu (čiže najlepšie je 100%) s použiteľnosťou je v priemere 72%. O málinko horšie je na tom spokojnosť s 70% a ako najlepšie bola hodnotená jednoduchosť použitia s priemerom 78%, čo môžem zhodnotiť ako dobré výsledky.

Užívatelia boli taktiež podporený vyjadriť svoj názor, či už v dotazníku alebo cez formulár na stránke. Názory, ktoré užívatelia vyjadrili:

- „Líbi se mi to. Takové jednoduché.“
- „Páči sa mi, že môžem úlohy ťahať a presúvať. Meranie schopnosti je naozaj motivujúce avšak je trochu únavné zadávať to tam.“
- „Nejsem moc na hry a ani herní prvky mě nelákají, ale jinak to má vše, co potřebuji.“
- „Niektoré veci by som upravil, ale používať sa to určite dá.“
- „Užitočné mi príde hlavne odpočítavanie míľnikov.“
- „Zkusil jsem a zjistil jsem, že to není pro mě. “

S interpretáciou užívateľských názorov to nie je jednoduché a aj presné čísla získané dotazníkom predstavujú len hrubý odhad. Pretože kritéria, čo vlastne znamená ideálnych 100% by užívateľ ťažko vedel definovať a neodpovedalo by to ideálu iného užívateľa, čiže výsledky sú ovplyvnené rôznymi faktormi. Užívatelia mohli mať napríklad privysoké očakávania a následne byť sklamaný keď sa nenaplnili alebo práve naopak niektorí podobnú aplikáciu používali po prvý krát a zrazu sa cítili, že tým získali projekty pod kontrolou. Ale tieto výsledky predsa len niečo znamenajú, pretože užívatelia sú naklonení skôr k pozitívnemu hodnoteniu a ak by aplikácia obsahovali výrazné nedostatky, tak by sa to viditeľne prejavilo. Ak by som to mal zhrnúť, aplikácia je použiteľná a užívatelia sú celkom spokojný, ale môže to byť lepšie.



Obrázek 5.4: Užívateľské názory - výsledky dotazníku

Kapitola 6

Záver

Cieľom práce bolo vytvoriť webovú aplikáciu pre tvorbu a správu cieľov projektov, ktorá umožňuje aj získavanie, organizáciu a zdieľanie informácií a to vo forme užívateľského rozhrania, ktoré sa jednoducho a efektívne používa. K tomu som naštudoval moderné webové technológie, zásady a postupy tvorby a testovania užívateľského rozhrania, princípy agilného riadenia projektov, moderný trend gamifikácie a súvisiacu psychológia (techniky stanovania a dosahovania cieľov a modely motivácie). Na základe týchto poznatkov som vytvoril návrh a implementoval základ aplikácie, ktorý som podrobil skorému testovaniu na užívateľoch (myslenie nahlas) pri ktorom sa ukázalo riešenie ako nepostačujúce a preto som navrhol a realizoval nové riešenie. Ďalšia iterácia testovania priniesla lepšie výsledky a objavilo sa len niekoľko drobných problémov, ktoré som vyriešil. Preto som sa zameral na efektívnosť rozhrania a identifikoval som najčastejšie operácie vykonávané v rozhraní, ktoré som zjednodušil a priradil im klávesové skratky. Na záver testovania UI som zisťoval užívateľské názory, ktoré tiež priniesli uspokojivé výsledky. Tým pádom môžem konštatovať, že sa úspešne podarilo splniť to, čo si táto práca predsavzala.

Dôležitá otázka, ktorá ostala zatiaľ nezodpovedaná je: akú hodnotu prináša vytvorená aplikácia a či bolo vôbec potrebné vytvárať ďalšiu aplikáciu keď už existujú iné riešenia? Na zodpovedanie si dovolím použiť analógiu s oblečením, aby som vyjadril k tomu môj subjektívny názor. Je potrebné, aby sa navrhovali nové druhy oblečenia keď už na trhu existuje obrovské množstvo rôznych oblečení ku všetkým príležitostiam? Nie, netreba. Avšak ľudia z rôznych dôvodov chcú nové druhy oblečenia a preto sa stále nové oblečenie navrhuje a vyrába. A rovnako aj pri riadení projektov vznikajú ďalšie riešenia, pretože ľudia majú potrebu hľadať niečo nové, čo im „sadne“ presne na mieru a v čom sa budú cítiť „in“. Pretože aj riadenie projektov je vec osobnej preferencie, čo sa potvrdilo aj pri experimentoch keď testovaní užívatelia často spomínali, že by si niečo upravili podľa svojich predstáv. Čiže aj táto aplikácia prichádza s iným prevedením (iné rozhranie, dizajn, ovládanie, atď.) a prispieva k rozšíreniu pestrosti a rôznorodosti existujúcich riešení.

Ďalšou hodnotou tejto aplikácie je, že sa pri nej užívatelia podvedome učia dobré techniky a postupy ako je stanovanie cieľu technikou mentálneho kontrastovania na začiatku projektu. A to by mohol byť aj smer pri ďalšom rozvoji aplikácie tj. spraviť učenie explicitný cieľ. Jednou formou ako to uskutočniť je pridanie interaktívneho kurzu, ktorý by postupne zoznamoval s rôznymi informáciami súvisiacimi s riadením projektu s dôrazom na praktické použitie priamo v aplikácií. Pretože aplikácia je len nástroj a ako s každým nástrojom platí, že závisí na užívateľových schopnostiach ako dobre ho vie používať.

Literatura

[1] Kniberg, H.: *Lean from the Trenches*. O'Reilly Media, 2011, ISBN 978-1-934356-85-2.

[2] Krug, S.: *Don't Make Me Think*. New Riders Publishing, 2006, ISBN 0-321-34475-8.

[3] Lund, A. M.: Measuring Usability with the USE Questionnaire [online]. `http://www.stcsig.org/usability/newsletter/0110_measuring_with_use.html`, 2004 [cit. 2014-05-02].

[4] Mathis, L.: *Designed for Use*. The Pragmatic Programmers LLC., 2011, ISBN-13 978-1-93435-675-3.

[5] Nielsen, J.: Why You Only Need to Test with 5 Users [online]. `http://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/`, 2000-03-19 [cit. 2014-04-27].

[6] Oettingen, G.; Gollwitzer, P. M.: Strategies of setting and implementing goals [online]. `http://www.psych.nyu.edu/gollwitzer/OettingenGollwitzer.pdf`, 2010 [cit. 2014-05-12].

[7] Preece, J.: *Human-Computer Interaction*. Addison-Wesley, 1994, ISBN 0-201-62769-8.

[8] Rappin, N.: *Rails Test Prescriptions*. The Pragmatic Programmers LLC., 2010, ISBN-13 978-1-934356-64-7.

[9] Ruby, S.; Thomas, D.; Hansson, D. H.: *Agile Web Development with Rails*. The Pragmatic Programmers LLC., 2010, ISBN-13 978-1-934356-54-8.

[10] Scott, B.; Neil, T.: *Designing Web Interfaces*. O'Reilly Media, 2009, ISBN 978-0-596-51625-3.

[11] Tidwell, J.: *Designing Interfaces*. O'Reilly Media, 2010, ISBN 978-1-4493-7972-8.

[12] Werbach, K.: *For the Win: How Game Thinking Can Revolutionize Your Business*. Wharton Digital Press, 2012, ISBN 978-1613630235.

[13] Wu, M.: Gamification 101: The Psychology of Motivation [online]. `https://community.lithium.com/t5/Science-of-Social-blog/Gamification-101-The-Psychology-of-Motivation/ba-p/21864`, 2011-03-01 [cit. 2014-02-14].

Příloha A

Obsah CD

Na priloženom CD sa nachádzajú tieto priečinky a súbory:

```
data/              # dáta z experimentov
  |-- Actions frequency.xlsx               # experiment 5.2
  |-- Thinking aloud - testing protocol.txt # experiment 5.1
  |-- Thinking aloud - video sample.wmv     # experiment 5.1
  +-- USE questionnaire.xlsx               # experiment 5.3
documents/         # video, plagát a technická správa
srcApp/            # zdrojové súbory aplikácie
  |-- doc/         # dokumentace aplikácie - schémy
  |-- Gemfile      # použité knihovny (gemy) - stiahnu sa pri inštalácií
  |-- README.rdoc  # inštrukcie k inštalácií
  +-- ...          # ďalšie súbory aplikácie popísané v realizácií
scrTs/             # zdrojové súbory technickej správy
sketches/          # ďalšie náčrty aplikácie
```

Obrázek A.1: Popis obsahu CD