VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

PODPORA MANAGEMENTU HARMONOGRAMU V PROJEKTECH S AGILNÍM PŘÍSTUPEM VÝVOJE PRODUKTU

SUPPORT OF PROJECT SCHEDULE MANAGEMENT WITH AGILE PRODUCT DEVELOPMENT

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. ANDREJ BLIŽŇÁK**
AUTHOR

VEDOUCÍ PRÁCE — **doc. RNDr. JITKA KRESLÍKOVÁ, CSc.**
SUPERVISOR

BRNO 2019

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2018/2019

Zadání diplomové práce

Student: **Bližňák Andrej, Bc.**
Program: Informační technologie Obor: Management a informační technologie
Název: **Podpora managementu harmonogramu v agilním vývoji produktu**
Support of Schedule Management in Agile Product Development
Kategorie: Softwarové inženýrství

Zadání:

1. Seznamte se se znalostními oblastmi managementu projektů dle aktuálního standardu PMI. Zaměřte se na procesy znalostní oblasti Management harmonogramu v rámci projektu.
2. Seznamte se s principy a metodami agilního řízení projektů. V jednotlivých znalostních oblastech identifikujte vhodné formy uplatnění agilního přístupu. Zaměřte se na jejich možné kombinace pro řízení času v projektech.
3. Analyzujte prostředí vývoje softwarových produktů v reálné firmě a specifikujte požadavky na systém pro podporu použití agilního přístupu managementu harmonogramu v projektech. Uvažujte případně i specifika malých firem. Systém navrhněte.
4. Zvolte vhodné vývojové prostředí a navržený systém implementujte.
5. Na vzorku dat, vybraném po dohodě s vedoucí, ověřte funkčnost vytvořeného systému.
6. Zhodnoťte dosažené výsledky a diskutujte možnosti jeho dalšího rozšíření.

Literatura:

- *A Guide To The Project Management Body Of Knowledge:* Sixth Edition, Project Management Institute, 2017. ISBN 978-1-62825-184-5.
- *Agile Practice Guide*: global standard Project Management Institute, 2017. ISBN 978-1-62825-199-9.

Při obhajobě semestrální části projektu je požadováno:

- Splnění bodů 1 až 3 zadání.

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Kreslíková Jitka, doc. RNDr., CSc.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 22. května 2019
Datum schválení: 30. října 2018

Abstrakt

Táto práca sa zaoberá riešením problematiky z oblasti riadenia projektov a konkrétnejšie z oblasti riadenia harmonogramu s prvkami agilných techník. Následne je skúmané prostredie malých firiem, na základe čoho vznikol návrh systému. Tento systém je potom implementovaný a testovaný. Na záver sú diskutované možnosti ďalšieho rozšírenia.

Abstract

This thesis deals with the solution from the project management especially with the focus on the schedule management with attributes of agile techniques. Subsequently, the small business enviroment is researched, based on which the system was designed. At the end, the further possibilities of further expansion are discussed.

Kľúčové slová

PMI, PMBOK, projektové riadenia, riadenie harmonogramu, agilné techniky, Firebase, React, softvérové inžinierstvo

Keywords

PMI, PMBOK, project management, schedule management, agile techniques, Firebase, Reack, software engineering

Citácia

BLIŽŇÁK, Andrej. *Podpora managementu harmonogramu v projektech s agilním přístupem vývoje produktu.* Brno, 2019. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce doc. RNDr. Jitka Kreslíková, CSc.

Podpora managementu harmonogramu v projektech s agilním přístupem vývoje produktu

Prehlásenie

Prehlasujem, že som túto semestrálnu prácu vypracoval samostatne pod vedením docentky Jitky Kreslíkovej. Uviedol som všetky literárne pramene a publikácie z ktorých som čerpal.

.....
Andrej Bližňák
19. mája 2019

Poďakovanie

Moje poďakovanie bude smerované viacerým. V prvom rade chcem menovite poďakovať pani docentke Jitke Kreslíkovej za jej prístup a odbornú pomoc. Moje poďakovanie patrí aj všetkým ľuďom v okolí, ktorí nejakým spôsobom prispeli k tomu, aby táto práca bola dokončená s čo najmenšou psychickou ujmou na mojom zdraví. Je to hlavne moja najbližšia rodina, priateľka, spolubývajúci, kamaráti a samozrejme všetci ľudia čo venovali svoj čas prípadnej spätnej väzbe k diplomovej práci.

Obsah

Kapitola 1

Úvod

Táto práca sa zaoberá znalostnými oblasťami podľa štandardu *PMI*, konkrétne riadením harmonogramu a jeho prepojením na agilné metódy pre agilné riadenie. Tieto znalosti sú potrebné pre návrh riešenia podporného nástroja v skúmanom prostredí.

Prvým krokom, v ktorý, bolo potrebné spraviť je štúdium štandardov podľa *PMI*, ktoré je potom popísaný v kapitole 2. Následne na to je venovaná kapitola 3 konkrétnej oblasti manažmentu harmonogramu a ich procesom. Tu sú už naznačené väzby na rôzne iné oblasti projektového riadenia.

Popis a následný prienik štandardov podľa *PMI* s agilnými metódami je spracovaný v kapitole 4, kde sú spomenuté aj niektoré konkrétne príklady agilných metodík. Následne bolo potrebné analyzovať aj konkrétne prostredie firmy, ktoré bolo analyzované v kapitole 5, kde je popísané aj všeobecné prostredie začínajúcich firiem v oblasti informačných technológií.

Toto štúdium rôznych oblastí projektového riadenia je inšpiráciou pre návrh prototypu v kapitole 6. V rámci neho sú vykonané aj ďalšie činnosti, ako vytvorenie prípadov použitia, návrh architektúry a i. Po návrhu je možné venovať sa samotnej implementácií systému, ktorá je zachytená v kapitole 7.

Po implementácii nasleduje užívateľské testovanie, ktoré je popísané v kapitole 8. Na výsledky testov nadväzuje už samotné zhodnotenie výsledkov implementovanej aplikácie v kapitole 9.

Cieľ tejto práce by sa dal rozdeliť na dve časti a to teoretický a praktický, kde teoretickým cieľom je získanie vedomostí zo znalostných oblastí podľa štandardu *PMI* spolu s agilnými metódami. Praktickým cieľom práce je potom návrh a implementácia systému, ktorý vychádza zo znalostí získaných v teoretickej časti.

Kapitola 2

Znalostné oblasti podľa štandardu PMI

Pred samotným stručným súhrnom štandardu je vhodné stručne zhrnúť, čo sa vlastne myslí *projektovým riadením*.

Tento termín označuje proces, ktorý koordinuje zložky činností pri realizácii projektu. Nemusí sa jednať len o komerčné projekty, ale tieto techniky sa dajú aplikovať aj na oblasti súkromných projektov.

V súčasnosti existuje niekoľko štandardov pre riadenie projektu ako napr. *IPMA*[1], *Prince2*[2] alebo spomínané *PMI*.

Project Management Institute (skr. *PMI*) je medzinárodná nezisková organizácia zaoberajúca sa vývojom štandardov, výskumom a inými činnosťami v oblasti projektového manažmentu.

Táto práca pojednáva o štandardoch vychádzajúcich z najnovšieho vydania smerníc, pravidiel a charakteristík, hlavne pre projektový manažment. Táto kapitola bude teda vychádzať primárne z najnovšieho kmeňového diela organizácie *PMI* a to konkrétne *A guide to the Project Management Body of Knowledge* (skr. *PMBOK guide*) alebo teda vo voľnom preklade *Smernice hlavných znalostných oblastí projektového manažmentu*. Konkrétne sa jedná o šieste vydanie z roku 2017[3].

2.1 Riadenie integrácie projektu

Táto časť projektového manažmentu zahrňuje procesy a aktivity, ktoré majú za úlohu identifikovať, definovať, kombinovať, zjednotiť ale aj koordinovať rôzne procesy a činnosti projektového manažmentu [2].

To zahŕňa rozhodovania v rámci:

- alokácie zdrojov,
- vyvažovania konkurenčných požiadaviek,
- skúmania alternatívnych prístupov,
- prispôsobovania procesov k splneniu cieľov projektu,

[1] https://www.ipma.world/
[2] http://www.prince2.cz/
[3] https://www.pmi.org/pmbok-guide-standards

- riadenia vzájomných závislostí medzi *hlavnými vedomosťami projektového manažmentu.*

Samotné procesy potom v rámci riadenia integrácie projektu sú:

- vypracovanie charty projektu,
- vypracovanie plánu riadenia projektu,
- riadenie projektovej práce,
- riadenie vedomostí o projekte,
- monitorovanie a kontrola práce projektu,
- vykonanie integrovanej kontroly zmien,
- uzavretie projektu, fázy alebo kontraktu.

Viaceré výstupy z týchto procesov sú vstupmi pre ďalšie procesy.

2.2 Riadenie rozsahu projektu

Táto časť projektového riadenia zahŕňa procesy, ktoré majú za úlohu vytýčiť všetku potrebnú prácu k úspešnému dokončeniu projektu [2].

Náležité procesy sú:

- riadenie rozsahu plánu,
- zozbieranie požiadaviek,
- definícia rozsahu,
- vytvorenie WBS[4],
- validácia rozsahu,
- monitorovanie rozsahu.

V kontexte projektu sa termín *rozsah* vyskytuje v dvoch podobách. Jedna z nich je *rozsah produktu*, ktorý vyjadruje vlastnosti a funkcie daného produktu, služby alebo všeobecne výsledku.

V druhom prípade sa jedná o *rozsah projektu*, ktorý špecifikuje prácu potrebnú pre dodanie produktu, služby alebo všeobecne výsledku [2].

2.3 Riadenie harmonogramu projektu

Táto časť projektového riadenia zahŕňa procesy potrebné dokončenie projektu v stanovenom termíne.

Jednotlivé procesy riadenia harmonogramu projektu sú:

- plánovanie riadenia harmonogramu,

[4] z anglického *Work Breakdown Structure* – často prekladané ako hierarchická štruktúra práce

- definovanie činností,
- radenie činností,
- odhad trvania činností,
- vytvorenie harmonogramu,
- sledovanie harmonogramu.

Riadeniu harmonogramu projektu je venovaná celá kapitola 3.

2.4 Riadenie nákladov projektu

Táto časť projektového riadenie sa zaoberá plánovaním, odhadovaním, rozpočítavaním, financovaním, riadením financovania a kontrolou nákladov tak, aby mohol byť projekt dokončený so schváleným rozpočtom [2].

Riadenie nákladov projektu zahrňuje procesy:

- plán riadenia nákladov,
- odhadovanie cien,
- určenie rozpočtu,
- monitorovanie nákladov.

Najmú v rámci menších projektov sú často prvé z troch menovaných procesov združené do jedného procesu.

2.5 Riadenie kvality projektu

Časť projektového riadenia kvality projektu zahrňuje procesy pre začlenenie politík kvality organizácie v súvislosti s plánovaním, riadením, monitorovaním projektu a produktovými požiadavkami v záujme požiadaviek zúčastnených strán [2].

Procesy riadenia kvality projektu sú:

- plánovanie kvality riadenia,
- riadenie kvality,
- kontrola kvality.

Jednotlivé procesy sa môžu líšiť naprieč priemyselnými odvetviami.

2.6 Riadenie zdrojov projektu

Táto časť projektového riadenia zahŕňa procesy na identifikáciu, získanie a riadenie zdrojov potrebných pre úspešné dokončenie projektu. Jednotlivé procesy napomáhajú zaistiť dostupnosť zdrojov v čase a mieste [2].

Takéto procesy sú:

- plánovanie riadenia zdrojov,

- odhad zdrojov činností,
- získanie zdrojov,
- tvorba tímu,
- riadenie tímu,
- kontrola zdrojov.

V praxi tieto procesy vzájomne spolupracujú.

2.7 Riadenie komunikácie projektu

Manažment komunikácie projektu zahŕňa procesy, ktoré zaisťujú aby výmena informácií v rámci projektu prebiehala medzi všetkými zúčastnenými stranami. Medzi procesy patria:

- plánovanie riadenia komunikácie,
- riadenie komunikácie,
- sledovanie komunikácie.

Riadenie komunikácie je rozdelené do dvoch častí, prvá časť slúži pre vytvorenie stratégie efektívnej komunikácie medzi zúčastnenými stranami a druhá je určená na vykonávanie samotných činností potrebných pre implementáciu komunikačných stratégií [2].

2.8 Riadenie rizík projektu

Časť projektového riadenia zahŕňa procesy súvisiace s plánovaním rizík, identifikáciou, analýzou ako aj návrhom (konkrétnych) opatrení. Cieľom spomínaných procesov je zvýšenie následkov možných príležitostí (alebo pozitívnych rizík) ako aj zníženie následkov negatívnych rizík [2].

Konkrétne procesy potom sú:

- plánovanie riadenia rizík,
- identifikácia rizík,
- vykonávanie kvalitatívnej analýzy rizík,
- vykonávanie kvantitatívnej analýzy rizík,
- plánovanie opatrení,
- implementácia opatrení,
- kontrola rizík.

Jednotlivé procesy sú v praxi často úzko previazané.

2.9 Riadenie obstarávania projektu

Táto časť projektového riadenia špecifikuje procesy na získavanie potrebných zdrojov, produktov, služieb atď. z prostredia mimo projektového tímu. Táto oblasť zahrňuje riadenie a kontrolu procesov pre vytvorenie požiadavok ku kontraktom, zmluvám, objednávkam atď. [2]

Procesy sú:

- plánovanie riadenia obstarávania,
- vykonávanie obstarávania,
- kontrola obstarávania.

V praxi sú často tieto procesy komplexne združené.

2.10 Riadenie zainteresovaných strán projektu

Projektové riadenie zainteresovaných strán má za úlohu identifikáciu ľudí, skupín a organizácií, ktoré môžu ovplyvňovať alebo môžu byť ovplyvnené projektom.

Procesy sú:

- identifikácia zúčastnených strán,
- plánovanie zainteresovania zúčastnených strán,
- riadenie zainteresovania zúčastnených strán,
- kontrola zainteresovania zúčastnených strán.

Úlohou jednotlivých procesov je podpora práce tímu, analýza očakávaní zúčastnených strán, určovanie akou mierou budú jednotlivé strany projektom ovplyvnené, ako aj vývoj stratégií na podporu efektívneho zapojenia zúčastnených strán k rozhodovaniu, plánovaniu a výkonu práce na projekte [2].

Kapitola 3

Procesy manažmentu harmonogramu

Manažment harmonogramu je jeden z desiatich spomínaných oblastí projektového manažmentu stručne popísaných v kapitole 2.

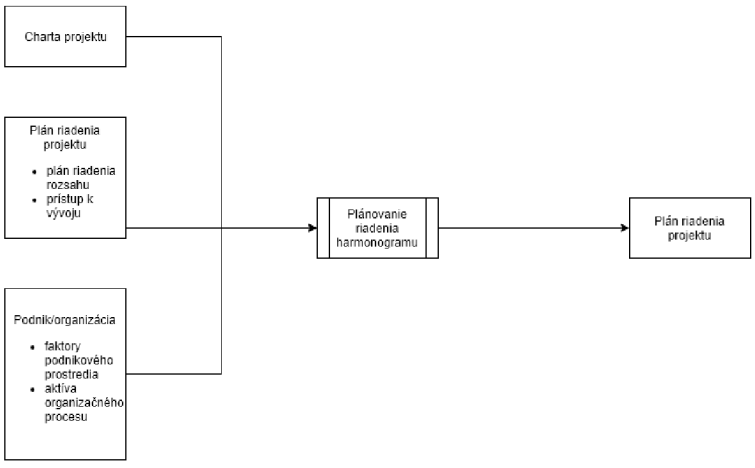
Manažment harmonogramu projektu v predchádzajúcej verzii *PMBOKu* nefiguroval pod týmto názvom. V piatej verzii z roku 2013 je časť venovaná plánovaniu aktivít pomenovaná ako *časový manažment projektu*.

V popise procesov je následne jediný rozdiel a to ten, že je združený *odhad trvania činností* a *odhad zdrojov pre činnosti* len do *odhadu trvania činností* v novom vydaní.

Táto sa zaoberá samotným rozdelením tejto oblasti *PMI* na procesy s ich vstupmi a výstupmi, ktoré majú súvis s inými oblasťami.

3.1 Plánovanie riadenia harmonogramu

Tento proces má za úlohu nastaviť pravidlá, procedúry a dokumentáciu pre vývoj, riadenie, vykonávanie a kontrolu plánu projektu.



Obr. 3.1: Zjednodušený diagram vstupov a výstupov plánovania harmonogramu projektu [inšpirované [2]]

Na obrázku 3.1 je zachytené plánovanie riadenia harmonogramu a súvislosti vstupov, z ktorých sú pomocou nástrojov a techník tvorené výstupy.

Vstupy

Jedným zo vstupov je tzv. *charta projektu* (zo sekcie 2.1 o riadení integrácie projektu), ktorá definuje naplánované ciele, ovplyvňujúce plánovanie riadenia harmonogramu.

Ďalším zo vstupov je plán riadenia projektu, ktorý zahrňuje (ale nie je obmedzený na) *plán riadenia rozsahu* (zo sekcie 2.2 o riadení rozsahu projektu) a z *plán prístupu k vývoju* (výstupom z *vypracovania plánu riadenia projektu* zo sekcie 2.1).

Ďalším vstupom sú *faktory prostredia*, v tomto prípade faktory podnikového prostredia, ktoré zahŕňajú:

- organizačnú kultúru a štruktúru,
- zdroje – z pohľadu tímov, zručností a fyzickej dostupnosti zdrojov,
- plánovací softvér,
- databázy, napr. údaje štandardizovaného odhadovania.

Posledným vstupom sú tzv. *aktíva organizačného procesu*. Tie zahŕňajú:

- informácie z minulosti a ponaučenia,
- existujúce formálne a neformálne plány vývoja, riadenia atď.

Toto sú vstupy pre následné nástroje a techniky.

Nástroje a techniky

Plánovanie riadenia harmonogramu má jednotlivé nástroje a techniky.

Odborný úsudok

Jedná sa o odborné skúsenosti či už jednotlivcov alebo skupín na predchádzajúcich projektoch s podobným charakterom. Rovnako sa môže jednať o predchádzajúce školenia v danej oblasti.

Dátová analýza

Dátová analýza zahŕňa rôzne druhy analýz. Často sa jedná o výber vhodnej metodiky k vytvoreniu harmonogramu. Môže sa tiež jednať o určenie aký detailný má byť harmonogram, ako často kontrolovať priebeh atď.

Schôdze

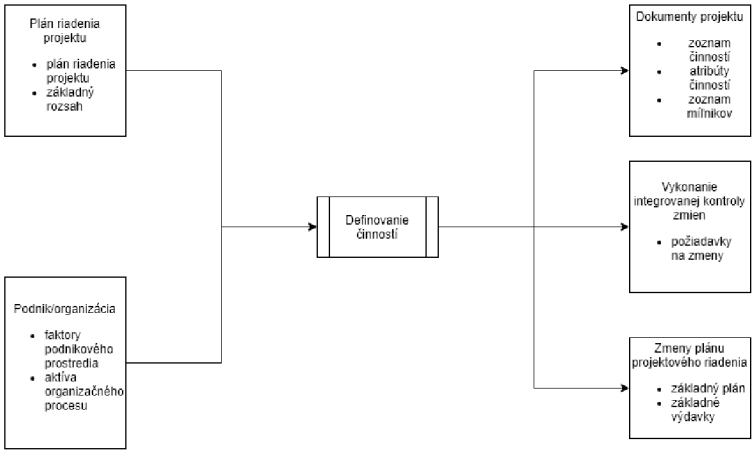
Projektové tímy môžu zvolávať schôdze, na ktorých sa zúčastňujú rôzni účastníci projektu – manažér, vybraní členovia projektu, vybrané zúčastnené strany a ktokoľvek, kto má zodpovednosť plánovania.

Výstupy

Výstupom je *plán riadenia harmonogramu*, ktorý môže ustanoviť napr.: vývoj modelu projektu, životný cyklus vydávania softvéru, úroveň presnosti odhadu, merné jednotky atď.

3.2 Definovanie činností

Ako už samotný názov napovedá, jedná sa o proces identifikovania a dokumentovania činností potrebných k dodaniu potrebných výstupov k projektu.



Obr. 3.2: Zjednodušený diagram vstupov a výstupov definovania činností [inšpirované [2]]

Obrázok 3.2 znázorňuje popisovaný proces v závislosti na vstupoch a jednotlivé výstupy daného procesu.

Vstupy

Jedným zo vstupov je tzv. *plán riadenia projektu* (zo sekcie 2.1 o projektovej integrácii). Jeho súčasťou môžu byť:

- plán riadenia projektu (z prechádzajúceho procesu plánovania riadenia harmonogram 3.1),
- základný rozsah (z riadenia rozsahu projektu 2.2).

Ďalšími vstupmi sú:

- faktory podnikového prostredia ako sú organizačná kultúra a štruktúra, informačný systém pre riadenie projektov atď.,
- aktíva procesu organizácie ktoré zahŕňajú skúsenosti z podobných projektov z minulosti, štandardizované procesy, existujúce pravidlá pre plánovanie, atď.

Tieto vstupy sú následne využívané nástrojmi a technikami.

Nástroje a techniky

Rovnako ako bolo uvedené v sekcii 3.1 k nástrojom a technikám v tejto časti patria: **odborný úsudok**, **schôdze** a ďalšie, zatiaľ nepopísané nástroje a techniky.

Dekompozícia

Dekompozíciou sa rozumie technika rozdeľovania atribútov projektu na menšie časti, ktoré sú z hľadiska riadenia lepšie zvládnuteľné.

„Rolling wave planning“

Technika iteratívneho plánovania, pri ktorej je práca na najbližšie obdobie naplánovaná podrobnejšie ako práca vo vzdialenejšej budúcnosti.

Jedná sa o progresívnu metódu vyhodnocovania použiteľného k vytvoreniu častí práce alebo plánovania vydávaných verzií. Môže byť aplikovaná vodopádovým alebo agilným prístupom.

Výstupy

Výstupmi tohto procesu sú:

- Zoznam činností – pri projektoch s využitím techniky „rolling wave planning“ sa jedná o pravidelne aktualizovaný zoznam činností.
- Atribúty činností – ním sa rozumie unikátny identifikátor činnosti, identifikátor *WBS* a názov alebo označenie činnosti. Tieto atribúty môžu byť doplňované o rôzne iné užitočné atribúty napr: predchodca, nasledovník, trvanie činnosti atď.
- Zoznam míľnikov – míľnikmi sa rozumejú významné body v projekte s nulovou dĺžkou trvania.
- Požiadavky na zmeny – požiadavky, ktoré vyplynuli z progresívneho vyhodnocovania výstupov a môžu mať za následok návrh na zmenu pôvodných základov projektu.
- Zmeny plánu projektového riadenia – akákoľvek zmena plánu riadenia projektu by mala viesť k požiadavke na zmenu: základných požiadaviek, základných výdavkov atď.

Výstupy sú vstupmi pre ďalšie procesy.

3.3 Radenie činností

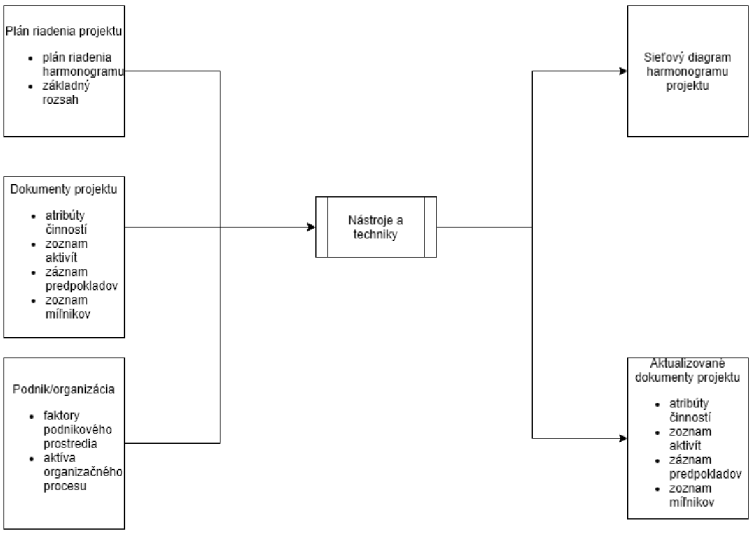
Proces, ktorý má za úlohu identifikovať a zaznamenať vzťahy medzi jednotlivými činnosťami v rámci projektu nazývame „radenie činností“.

Obrázok 3.3 znázorňuje vstupy potrebné pre proces radenia pomocou nástrojov a techník, z ktorých vznikajú znázornené výstupy.

Vstupy

Jedným zo vstupov je **plán riadenia projektu**, popísaný ako aj vstup pre definovanie činností zo sekcie 3.2. Ďalším vstupom je **plán riadenia harmonogramu** zo sekcie 3.1.

Ostatnými vstupmi sú faktory podnikového prostredie ako napr.: organizačná kultúra a štruktúra, informačný systém pre riadenie projektov atď.



Obr. 3.3: Zjednodušený diagram vstupov a výstupov radenia činností [inšpirované [2]]

Dokumenty projektu

Posledným vstupom sú doteraz nespomenuté *dokumenty projektu* a tie zahŕňajú nasledovné:

- atribúty činností,
- zoznam aktivít,
- záznam predpokladov - predpoklady k radeniu aktivít, vzťahom medzi aktivitami atď.,
- zoznam míľnikov.

Tieto vstupy opäť vchádzajú do nižšie spomenutých nástrojov a techník.

Nástroje a techniky

Z doposiaľ známych techník sa jedná o využívanie *informačného systému pre riadenie projektu*.

Diagram priorít

Jedna zo základných techník je tvorba *diagramu priorít*. Takýto diagram má za úlohu znázorniť logické vzťahy medzi jednotlivými aktivitami. Tieto väzby kombinujú následnosť začiatkov a koncov aktivít v rôznych kombináciách napr. koniec jednej aktivity nadväzuje na začiatok druhej, začiatok jednej nadväzuje na začiatok druhej atp.

Určenie závislostí a integrácia

Závislosti môžu byť určené nasledujúcimi možnosťami: povinné alebo diskrétne a interné alebo externé.

Povinné závislosti sú striktne vymedzené či už zákonom alebo zmluvou. Diskrétne sú založené často na skúsenostiach alebo iných zvykoch. Externé závislosti určujú vzťah medzi

činnosťami mimo prácu na projekte (napr. závislosť na dodávanom hardvéri) a interné sú opakom.

„Vedenie“ a „zaostávanie“

Jedná sa o určenie časového oneskorenia medzi rôznymi procesmi. Existujú tu dva druhy väzieb a to: oneskorenie aktivity po konci inej a druhý variant je oneskorenie aktivity po začiatku inej.

Výstupy

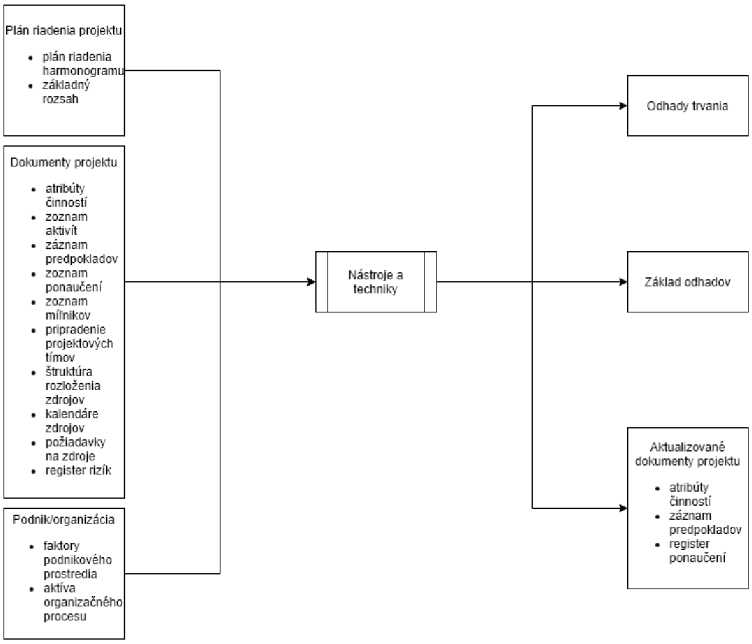
Jedným z výstupov sú aktualizované **dokumenty projektu**.

Sieťový diagram harmonogramu projektu

Sieťový diagram harmonogramu projektu predstavuje grafickú reprezentáciu logických vzťahov - často nazývaný aj grafická reprezentácia medzi naplánovanými činnosťami v rámci projektu.

3.4 Odhad trvania činností

Jedná sa o proces určovania počtu časových jednotiek potrebných na dokončenie jednotlivých činností.



Obr. 3.4: Zjednodušený diagram vstupov a výstupov odhadu trvania činností [inšpirované [2]]

Obrázok 3.4 graficky znázorňuje jednotlivé vstupy, ktoré sú pomocou nástrojov a techník premieňané na výstupy.

Vstupy

Jedným zo vstupov je opäť **plán riadenia projektu**, popísaný ako aj vstup pre definovanie činností zo sekcie 3.2. Ďalším zo vstupov sú faktory podnikového prostredia ako napr.: organizačná kultúra a štruktúra, informačný systém pre riadenie projektov atď.

Dokumenty k projektu

Dokumenty k projektu obsahujú:

- atribúty činností,
- zoznam činností,
- záznam predpokladov,
- register ponaučení – ponaučenie z predchádzajúcich fáz projektu, slúžiace ako ponaučenie pre nasledujúce projekty,
- zoznam míľnikov,
- priradenie projektového tímu,
- štruktúru rozloženia zdrojov,
- kalendár zdrojov – kalendár ovplyvňujúci trvanie činností v závislosti na obsadenosti zdrojov,
- požiadavky na zdroje,
- register rizík – vyplývajúci z analýzy rizík.

Tieto vstupy opäť vchádzajú do nižšie spomenutých nástrojov a techník.

Nástroje a techniky

Sekcia využíva okrem nových nástrojov a techník aj *odborný úsudok* ,ktorý už bol uvedený v predchádzajúcich sekciách.

Analogické odhadovanie

Jedná sa o techniku odhadovania dĺžky trvania, ako aj ceny jednotlivých činností na základe historických dát. Ide o najúspornejšiu techniku z hľadiska potrebného času a financií, ale táto technika je menej presná.

Parametrické odhadovanie

Parametrické odhadovanie je technikou odhadovania ceny a času trvania činností na základe algoritmov podložených konkrétnymi historickými dátami z vykonávania podobných činností. Presnosť danej techniky záleží na dátach, ktorými sú výpočty podložené.

Trojčíselný odhad

Ako už napovedá názov, jedná sa o vylepšenie metódy odhadu na základe jedného čísla, ale v tomto prípade sa jedná o odhad na základe až troch čísel.

$$tE = (t0 + tM + tP)/3 \tag{3.1}$$

Vyššie spomenutý vzorec 3.1 vyjadruje výpočet takejto hodnoty, kde *tE* je výsledná hodnota, *t0* optimistický časový odhad trvania činnosti, *tM* najpravdepodobnejšia dĺžka trvania činnosti a *tP* je pesimistický odhad trvania činnosti.

Odhad zdola hore

Odhad na základe *WBS*, kde sa ohodnocujú časti na nižších úrovniach rozkladu a následne sú agregované do vyšších úrovní a zároveň je určená doba trvania činností.

Dátová analýza

Dátová analýza môže používať následné techniky:

- Analýza alternatív – analýza porovnávania rôznej úrovne schopností a zručností s využitím rôznych ďalších rozhodnutí, týkajúcich sa zdrojov (nákup, prenájom, vytvorenie) s ohľadom na cenu.
- Analýza rezerv – analýza používaná na určenie núdzových a riadiacich rezerv potrebných pre projekt.

Rozhodovanie

V rozhodovaní sa často jedná o využitie širokej škály hlasovacích nástrojov.

Schôdze

Súčasťou práce projektových tímov je aj organizácia pracovných schôdzí, ktorých hlavným výstupom je časový odhad trvania jednotlivých častí projektu. Táto technika sa často využíva napr. pri agilnom vývoji.

Výstupy

Vykonaním techník na vstupoch je možné dostať následne popísané výstupy.

Odhady trvania

K jednotlivým činnostiam, fázam alebo k celému projektu je určený časový úsek, za ktorý je konkrétna časť vykonaná. Odhady sú často dopĺňané aj o možné natiahnutie tejto doby či už s konkrétnymi číslami alebo s percentuálnym vyjadrením.

Základ odhadov

Tieto výstupy obsahujú spektrum dokumentov, ktoré sú podkladom pre vykonané odhady. Sú to napr.:

- dokumentácia základov odhadov – ako boli odhady vykonané,

- dokumentácia predpokladov,
- dokumentácia obmedzení atď.

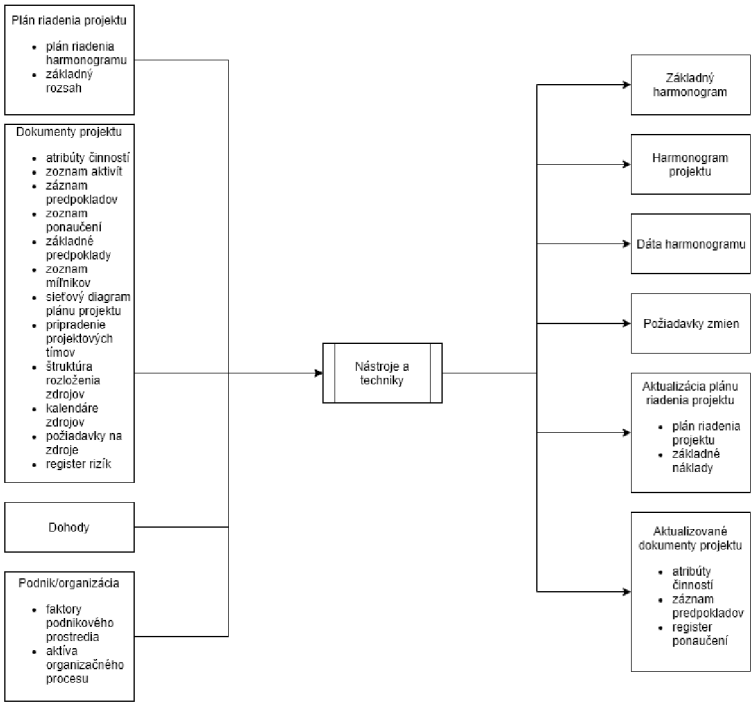
Tieto dokumenty a výstupy sú vstupmi pre ďalšie procesy v rámci štandardu *PMI*.

Aktualizácia dokumentov projektu

Jedným z výstupom môže byť opäť aktualizácia projektových dokumentov ktorej súčasťou sú aj atribúty činností, záznamy predpokladov, alebo register predpokladov.

3.5 Vytvorenie harmonogramu

Vytvorenie harmonogramu je proces analýzy poradia činností, doby trvania, požiadaviek na zdroje, obmedzenia plánu a i. Cieľom je vyhotovenie modelu na vykonávanie a kontrolu týchto činností.



Obr. 3.5: Zjednodušený diagram vstupov a výstupov vytvorenia harmonogramu. [inšpirované [2]]

Na obrázku 3.5 je diagram znázorňujúci vstupy potrebné pre nástroje a techniky, pomocou ktorých je možné dosiahnutie požadovaných výstupov.

Vstupy

Vstupy sú veľmi podobné ako v predchádzajúcich procesoch. Je to teda **plán riadenia projektu**. K **projektovým dokumentom** sa pridali *základné odhady* a *sieťový diagram harmonogramu projektu*.

Dohody

Vstupy sa tu zároveň rozširujú aj o *Dohody*, konkrétne dohody medzi dodávateľmi ktoré sú potrebné na dokončenie projektu.

Nástroje a techniky

K premene vstupov na výstupy existuje niekoľko techník a nástrojov. Okrem nových je tu opäť využitá technika *vedenia a zaostávania* a technika *informačných systémov pre riadenie projektu*.

Sieťová analýza harmonogramu

Jedná sa o techniku, ktorá využíva iné techniky (*metóda kritickej cesty*, *techniku optimalizácie zdrojov*, modelovacie techniky a i.) [8]. Sieťová analýza harmonogramu využíva iteratívny prístup, ktorý je aplikovaný dovtedy, kým nie je zostavený použiteľný model.

Metóda kritickej cesty

Cieľom tejto metódy je zostavenie harmonogramu činností na projekte. Tzv. *kritická cesta* je určená ako najdlhšia cesta medzi závislými činnosťami. Určuje najdlhší čas potrebný na dokončenie činnosti [10].

Optimalizácia zdrojov

Technika *optimalizácie zdrojov* sa používa k úprave začiatkov a koncov činností projektu tak, aby dostupné zdroje boli rovné alebo menšie ako požiadavky jednotlivých aktivít.

Dátová analýza

Môžu byť použité následné techniky:

- Analýza scenárov „čo ak“ – venuje sa prípadom, ktoré môžu nastať počas priebehu činností a snažia sa pripraviť scenáre tak, aby nevznikli nečakané situácie[1].
- Simulácia – používajú sa simulačné modely, ktoré obsahujú riziká projektov a kombináciu neistoty zdrojov k výpočtu dopadu na projekt.

V rámci dátovej analýzy môžu byť použité aj iné, vyššie nespomenuté nástroje a techniky.

Kompresia harmonogramu

Jedná sa o techniku zrýchľovania alebo skracovania naplánovaného harmonogramu bez redukovania rozsahu projektu. Kompresiu harmonogramu je možné dosiahnuť navýšením zdrojov pre jednotlivé aktivity, alebo za pomoci paralelizácie určitých činností, čím sa však zvyšuje riziko.

Plánovanie agilného vydávania

Táto technika poskytuje stručnú časovú os vydávania na základe produktovej mapy. Agilné plánovanie tak isto určuje počet iterácií (alebo šprintov) potrebných pre vydanie verzie a necháva na tímoch, koľko času je potrebné na ich dokončenie.

[1] https://project-management-knowledge.com/definitions/w/what-if-scenario-analysis/

Výstupy

Výstupy obsahujú časti spomenuté v predchádzajúcich sekciách ako sú *základný harmonogram*, *dokumenty projektu*, *plán projektového riadenia* a pribudli *požiadavky na zmeny*.

Harmonogram projektu

K výstupom pribudli plány harmonogramu, ktoré typicky tvoria:

- Stĺpcový diagram – inak nazývané aj *Ganttove diagramy*, ktoré reprezentujú naplánované činnosti na vertikálnych osách a horizontálna osa predstavuje časovú osu.
- Diagram míľnikov – jedná sa o variantu stĺpcového diagramu, s tým, že na vertikálnej ose sú míľniky a horizontálna osa často predstavuje väčšiu časovú jednotku.
- Sieťový diagram harmonogramu projektu – jedná sa o diagram, ktorý okrem činností zachytáva aj závislosti medzi jednotlivými činnosťami na časovej osi.

Harmonogram projektu môže mať v niektorých prípadoch len tabuľkovú formu.

Dáta harmonogramu

Dáta harmonogramu sú informácie popisujúce a riadiace harmonogram projektu. Tieto dáta zahrňujú aspoň plánované míľniky, činnosti, atribúty činností a zdokumentované predpoklady a obmedzenia.

Projektové kalendáre

Kalendáre zobrazujú pracovné smeny dostupné na vykonávanie činností, ktoré boli naplánované. Často je mernou jednotkou pracovný deň alebo jeho časti.

3.6 Sledovanie harmonogramu

Ako napovedá názov, jedná sa o proces kontroly aktuálneho stavu projektu vzhľadom na harmonogram.

Obrázok 3.6 zobrazuje jednotlivé vstupy, ktoré sú pomocou nástrojov a techník menené na výstupy.

Vstupy

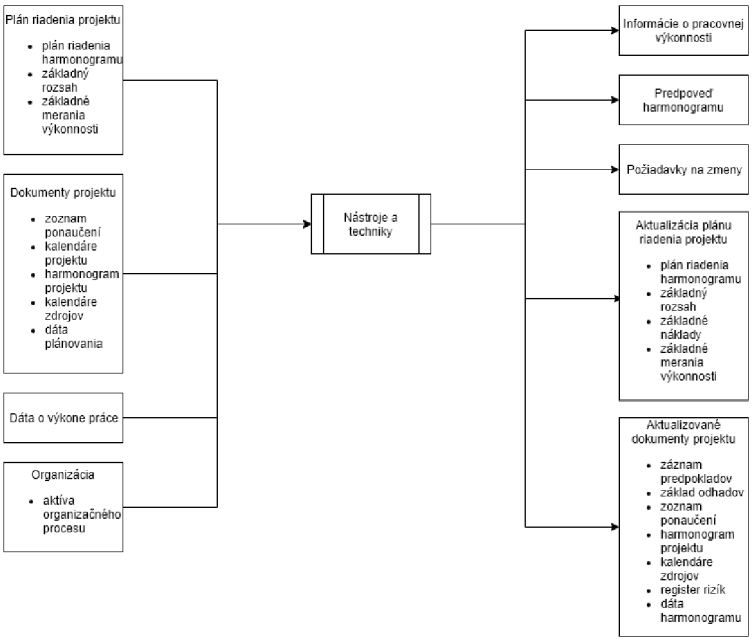
Základnými vstupmi v tejto sekcií sú: *plán riadenia projektu*, *základné merania výkonnosti*, *dokumenty projektu*, *dáta plánovania*, *aktíva organizačných procesov*.

Dáta o výkone práce

Dáta zachytávajú stav v akom sa projekt nachádza, konkrétne ktoré činnosti začali, ktoré prebiehajú a ktoré boli ukončené.

Nástroje a techniky

K sledovaniu harmonogramu projektu sú využívané rôzne techniky. Z už spomínaných je to *metóda kritickej cesty*, *informačné systémy pre riadenie projektov*, *optimalizácia zdrojov*, *kompresia harmonogramu* a technika *„vedenia“ a „zaostávania“*.



Obr. 3.6: Zjednodušený diagram vstupov a výstupov kontroly harmonogramu. [inšpirované [2]]

Dátová analýza

K dátovej analýze je možné použiť viacero techník a nástrojov, ktorými môže byť:

- Analýza dosiahnutej hodnoty – jedná sa o analýzu stavu projektu z pohľadu nákladov a času[2].
- Iteratívny graf „burndown“ – zobrazuje koľko zostáva práce v závislosti na čase. Ukážka na obrázku 3.7.

Je možné použiť aj iné, vyššie nespomenuté nástroje dátovej analýzy.

Výstupy

Výstupy sa zhodujú s výstupmi opísanými v predchádzajúcich sekciách, ako sú *požiadavky na zmeny*, *aktualizácie plánu riadenia projektu* a *aktualizácie dokumentov projektu*.

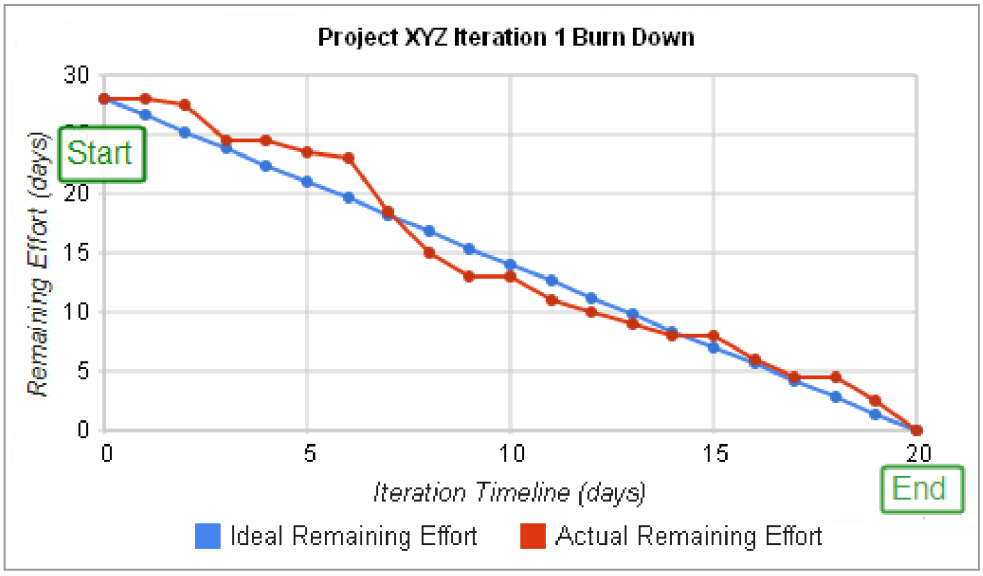
Informácie o pracovnej výkonnosti

Tento výstup má za úlohu určiť, ako prebiehajú činnosti vzhľadom na základný plán.

Predpoveď harmonogramu

Jedná sa o aktualizácie harmonogramu, ktoré vznikajú na základe informácií dostupných v čase vytvárania aktualizácie. Predpokladom pre tieto informácie sú *informácie o výkonnosti*.

[2] https://www.nasa.gov/evm



Obr. 3.7: Ukážka grafu typu *burndown*. Modrá čiara zobrazuje ideálny priebeh, červená aktuálny stav úloh [zdroj: [23]]

Kapitola 4

Princípy a metódy agilného riadenia

Agilné riadenie projektu je taký spôsob riadenia projektu, ktorý sa vyznačuje pripravenosťou a schopnosťou rýchlej reakcie a má vynaliezavý a prispôsobivý charakter [6].

V porovnaní s tradičným vývojom softvéru sa agilný vývoj zameriava hlavne na komplexné systémy a vývoj produktov s dynamickou, nedeterministickou a nelineárnou charakteristikou. Presné odhady, nemenné plány a odhady nie sú často možné na začiatku vývoja a je pravdepodobné, že ich istota je nízka [14].

Z tohto dôvodu si kladú agilné metódy za cieľ pracovať v krátkych cykloch a byť pripravený sa rýchlo prispôsobiť na základe vyhodnotení a spätných väzieb.

Konkrétne prístupy sú [1]:

- krátke cykly spätnej väzby,
- pravidelné prispôsobovanie procesov,
- reprioritizácia,
- pravidelné aktualizovanie plánov,
- časté dodávky.

Nasledujú charakteristiky rôznych druhov životných cyklov.

Charakteristika prediktívnych životných cyklov

Prediktívne životné cykly fungujú na princípe vysokej istoty požiadaviek, stabilnom tíme a nízkom risku. Na dosiahnutie takéhoto prístupu je potrebný potrebný plán toho, čo má byť doručené a ako. Takéto projekty uspejú, ak sa podarí čo najviac eliminovať možné zmeny [1].

Cieľom je správa výdavkov projektu.

Charakteristika iteratívneho životného cyklu

Iteratívne životné cykly vylepšujú produkt postupnými prototypmi overovania koncepcie. Každý nový prototyp prináša spätnú väzbu od zúčastnených strán a z dát tímu [1].

Projekty získavajú z tohto druhu životného cyklu projektu, ak je dosiahnutá vysoká komplexita, ak sa vyskytujú časté zmeny alebo ak je rozsah predmetom rôznych zúčastnených strán. Cieľom je správnosť výsledného riešenia.

Charakteristika inkrementálneho životného cyklu

Inkrementálny životný cyklus optimalizuje prácu na dodanie hodnoty sponzorom alebo zákazníkovi častejšie ako len jeden krát. Prvá dodávka produktu je čo najrýchlejšia.

Príkladom takéhoto prístupu je dodávanie produktu po jednotlivých vylepšeniach alebo po dokončení častí práce [1].

Cieľom teda je rýchlosť dodania produktu po častiach.

Charakteristika agilného životného cyklu

Charakteristika agilného vývoja už bola spomenutá na začiatku. Pre zhrnutie sa ale jedná o prístup k vývoju s dynamickými požiadavkami, v pravidelných dodávkach malých častí so spätnou väzbou od zákazníka.

4.1 Hybridné životné cykly

V reálnej praxi nie je nutné, aby sa používal striktne jeden životný cyklus - často sa používajú kombinácie vyššie spomínaných životných cyklov. Tieto kombinácie môžu byť v rôznych podobách, v závislosti na charaktere konkrétneho projektu.

Najčastejšie kombinácie tvorí kombinácia prediktívneho a agilného prístupu. Tá sa môže vyskytovať vo viacerých variantách.

Agilný vývoj s prediktívnym uvedením na trh

Tento prístup sa vyznačuje fázou vývoja, ktorá nemá striktne dané požiadavky - preto sa využíva agilný prístup. Po nej nasleduje prediktívna časť uvedenia na trh.

Typickým príkladom je vývoj technologickej novinky nasledovaný nasadením na trh a školením pre tisícky užívateľov [1].

Simultánna kombinácia agilného a prediktívneho prístupu

Takýto prístup kombinuje opäť prediktívny a agilný prístup počas celého trvania projektu, kde sú napr. krátke iterácie, denné stretnutia a retrospektívy, ale ostatné aspekty zostávajú prediktívne ako napr. predbežné odhady, priradenie práce a sledovanie postupu [1].

Takýto prístup by bolo nesprávne nazývať vyložene agilným alebo predikatívnym, nakoľko sa využívajú prístupy z oboch druhov životných cyklov. Preto je tento prístup skôr radený k hybridným.

Prediktívny prístup s agilnými komponentami a agilný prístup s prediktívnymi komponentami

Jedná sa o prístupy, kde výrazne dominuje jeden alebo druhý prístup, ale niektoré časti projektu sú agilné alebo opačne prediktívne.

4.2 Metódy agilného riadenia

Existuje niekoľko metód na podporu agilného riadenia. V tejto sekcii je uvedený popis niektorých momentálne populárnych metód agilného riadenia.

4.2.1 Lean metódy vývoja softvéru

Metódy vývoja softvéru nazývané *lean* vznikli adaptáciou od *Toyouta Production System.*

Princípy *lean* môžu byť zhrnuté v siedmych bodoch [16]:

- Eliminácia odpadu – odstránenie všetkého, čo neprináša žiadnu hodnotu.
- Zlepšenie učenia – učiť sa už v priebehu vykonávania činností projektu a upravovať procesy podľa spätnej väzby.
- Rozhodovanie čo najneskôr - rozhodovanie by malo byť podporené čo najviac istými skutočnosťami, preto je potrebné rozhodovať čo najneskôr s najväčším množstvom vedomostí.
- Dodávanie čo najskôr – produkt je potrebné dodať čo najskôr najmä v súvislosti so spätnou väzbou od koncového používateľa.
- Posilňovanie tímu – tím bude posilňovať, pokiaľ sa vezmú do úvahy názory všetkých členov a zároveň bude aj zodpovednosť rozdelená medzi všetkých členov.
- Stavanie integrity – vo vývoji softvéru nejde len o samotný finálny produkt. Je potrebné dbať na celkový dojem a kvalitu dodávaného systému.
- Videnie celku – je potrebné vidieť produkt ako celok so všetkými jeho časťami, z ktorých sa skladá. Čím väčší systém, tým viac úsilia je potrebné venovať organizácií práce.

4.2.2 Scrum

Ďalší agilný prístup je nazývaný *scrum.*

Je určený pre tímy o veľkosti 3–9 členov, ktorý si delia prácu na činnosti. Tieto činnosti majú byť finalizované v časovom rámci nazývanom *šprint.* Tie majú dĺžku najviac jeden mesiac a najčastejšie sa jedná o rámec dvoch týždňov.

Postup činností je kontrolovaný v denných *stand-up* stretnutiach nazývanými *denný scrum* [19].

4.2.3 Kanban

Metóda *Kanban* sa zameriava na poskytovanie vizuálneho systému pre rozhodovanie.

Samotná metóda má pôvod v *lean* metódach a zvyčajne sa používa v kombinácií s inými agilnými metódami.

Vizualizácia tejto metódy spočíva v používaní tzv. *tabúľ Kanban.* Tie obsahujú stĺpce, ktoré vyznačujú v akom stave sa daná úloha nachádza. Zvyčajný smer posúvania úloh je zprava doľava.

4.3 Prepojenie PMI a agilného riadenia

V jednotlivých znalostných oblastiach rozobraných v kapitole 2 a následne aj v konkrétne popísanej oblasti riadenia harmonogramu projektu v kapitole 3 je možnosť uplatnenia agilných metód pre agilné prostredia - nejedná sa teda len o striktne nastavené procesy, ako by sa mohlo zdať.

Hneď v úvode pri „riadení integrácie“ ktorá bola popísaná v sekcií 2.1 je potrebné zvážiť možností agilného prostredia. Kľúčovou úlohou manažéra je v tomto prípade kreovať čo najlepšie tímové prostredia pre prípadné zmeny v projekte ako aj samotné reakcie tímu na takéto zmeny. Pri zostavovaní tímu uprednostňujeme členov, ktorí disponujú širokým spektrom znalostí [2]. Následne je možné v tejto oblasti riadenia zvoliť konkrétnu agilnú metódu riadenia vývojového procesu.

V oblasti riadenia rozsahu zo sekcie 2.2 – je možné opäť využiť agilný prístup.

Zvyčajne sa tak deje, ako už bolo priblížené v popise agilných metód, hlavne ak je sponzor alebo zákazník priebežne oboznamovaný s výsledkami v iteráciách. Ten je aktívne zapojený do procesu a formou spätnej väzby zadáva nové požiadavky, ktoré sa zohľadňujú v ďalších analýzach rozsahu.

Z tohto dôvodu sa v agilnom prostredí nevenuje toľko času ustanoveniu rozsahu. Čas a úsilie je venované čo najrýchlejšiemu zdokonaľovaniu výsledku a včasnej špecifikácii skutočných požiadaviek zákazníka [2].

Oblasť riadenia harmonogramu, ktorej bola venovaná samostatná kapitola 3.1, sa často využíva agilný prístup pri iteratívnom plánovaní so zoznamom požiadaviek[1]. Požiadavky sú zhlukované do *user stories* a sú ohodnotené podľa priority na základe čoho sú radené v harmonograme.

Tento prístup je možné využiť aj vo väčších vývojových skupinách, kde je viac tímov a jednotlivé tímy môžu samostatne pracovať na vylepšeniach [2]. Rovnako sa tento prístup využíva aj v rámci niektorých hybridných metód, kde môžu agilné metódy doplňovať predikatívne.

V prípade riadenia harmonogramu je možné agilný prístup aplikovať aj pri tvorbe rozhodnutí, kde je na stretnutiach možné použiť napr. metódu tzv. „päsť piatich“[2]. Pomocou tejto metódy je možné rýchlo zistiť, aký názor majú členovia tímu na danú myšlienku podľa toho, koľko prstov ukazuje ich päsť [5].

Konkrétne názory sa vzápätí rozdiskutujú, prípadne sa priamo vezmú do úvahy [2].

Pre často používaný agilný prístup je typické využívanie spomínaných šprintov, ktoré sú diskutované na dennej báze a ich progres je pravidelne hodnotený. Prípadné nezrovnalosti je tak možné odstrániť čo najskôr.

To je v priamej interakcii s agilným plánovaním vydávania produktu, kde po každej iterácii, ako bolo viac krát spomenuté, je od sponzora či zákazníka zozbieraná spätná väzba, ktorá sa zapracováva do požiadaviek na ďalší vývoj. V zmysle plánovania harmonogramu je často určený len hrubý časový rámec a počet iterácií, ktoré majú byť priebežne dodávané. Takýmto prístupom je možné prinášať požadovanú hodnotu pre zákazníka už počas vývoja produktu, keďže ako bolo spomenuté, iterácie väčšinou so sebou prinášajú vylepšenia funkcionality produktu požadované zákazníkom.

Ďalšou oblasťou súvisiacou s agilným prístupom je riadenie ceny projektu, stručne popísaného v sekcii 2.4. Takýto prístup sa využíva pri projektoch s vysokou neistotou vstupných požiadaviek, kde sa namiesto detailného ocenenia ceny projektu určí len vysoko-úrovňový

[1] backlogom

[2] z angl. *fist of five* alebo *fist to five*

odhad cien prác. Detailnejší odhad je možné vykonať pred jednotlivými iteráciami. Vo veľkej miere sa jedná o projekty so striktným rozpočtom, a tak je na úkor financií koordinovaný rozsah a harmonogram projektu [2].

V prípade oblasti riadenia kvality zo sekcie 2.5 sa jedná o proces kontroly kvality nie až na konci celého vývojového procesu, ale kontrola kvality je vykonávaná priebežne, zvyčajne na konci každej iterácie. Takýmto spôsobom je možná skorá špecifikácia nedostatkov čím sa značne znižuje cena korekcie odhalených nedostatkov. V agilných tímoch sú za kvalitu zodpovední všetci členovia tímu počas celého vývojového cyklu.

V oblasti riadenia zdrojov zo sekcie 2.6 sa z agilného pohľadu prihliada najmä na riadenie jednotlivých členov tímu. Tieto tímy sú často samostatnou organizačnou jednotkou, kde neexistuje akákoľvek hierarchia. Tu je potrebné brať do úvahy všeobecné zameranie členov tímu, spomenuté aj v odseku s riadením integrácie projektu.

Podstatnou mierou s touto oblasťou súvisí oblasť organizácie komunikácie zo sekcie 2.7. Dôraz sa kladie najmä na pravidelné a rýchle stretnutia najmä členov tímu, ktorí často medzi sebou zdieľajú znalosti a skúsenosti.

Do procesu komunikácie je tiež potrebné zahrnúť aj ostatné zúčastnené strany ako napr. zákazník, s ktorým je v priebehu iterácií pravidelne preberaný ďalší postup, aktuálny výsledok atď.

V prípade oblasti riadenia rizík zo sekcie 2.8 sa jedná o prístup, kde sa analyzujú riziká medzi iteráciami. Riziká je potrebné preberať spolu so všetkými členami tímu, keďže sa jedná o prostredie so zdieľanými vedomosťami a schopnosťami.

V oblasti riadenia obstarávania zo sekcie 2.9 ide v agilnom prostredí o prístup, kde je samotný kupujúci zahrnutý do agilného tímu, čím sa na neho prenáša aj časť rizík. Týmto sa dá dosiahnuť aj prípadná väčšia ochota k zmenám na projekte zo strany kupujúceho[2].

Pre poslednú oblasť riadenia zainteresovaných strán zo sekcie 2.10 platí hlavne doteraz popisovaný prístup, kde sú všetky zúčastnené strany súčasťou agilného tímu, v ktorom neexistuje vyslovená autorita a všetky strany majú záujem na úspešnej spolupráci a dokončení projektu.

Kapitola 5

Analýza firemného prostredia

Táto kapitola sa zaoberá skúmanou firmou a jej prípadnými problémami z pohľadu projektového manažmentu.

5.1 Popis firmy

Skúmaná softvérová firma bola založená v roku 2010 ako *startup*. V súčasnosti má spoločnosť niečo vyše 30 zamestnancov vo viacerých oddeleniach.

Štruktúru firmy tvoria jednotlivé oddelenia: oddelenie záruky kvality a nasadenia produktov, oddelenie obchodu, zákaznícka podpora a oddelenie *backoffice*.

Firma sa špecializuje na dodávanie softvérových riešení pre záchranárske a mierové zložky v súvislosti s ich lokalizáciou. Z tohto prostredia pochádza aj niekoľko kľúčových zákazníkov. Finálnymi výstupmi firmy pre konečných zákazníkov sú často kombinované softvérovo-hardvérové riešenia nasadzované priamo u zákazníka.

Pracovníci na jednotlivých oddeleniach sú riadený prostredníctvom vedúcich oddelenia. Značná časť spolupráce na vývoji prebieha medzi oddelením obchodu a oddelením vývoja a zárukou kvality. Značný podiel práce projektových manažérov, v rámci jednotlivých produktov, preberajú v súčastnosti obchodní zástupcovia.

Charakteristika projektov je rozmanitá, niektoré projekty, hlavne z oblasti verejnej sféry majú jasne špecifikované zadanie už na samotnom začiatku s jasne daným rozpočtom. V takomto prípade je možné aplikovať jednu z predikatívnych metód vývoja, avšak nie vždy je to možné.

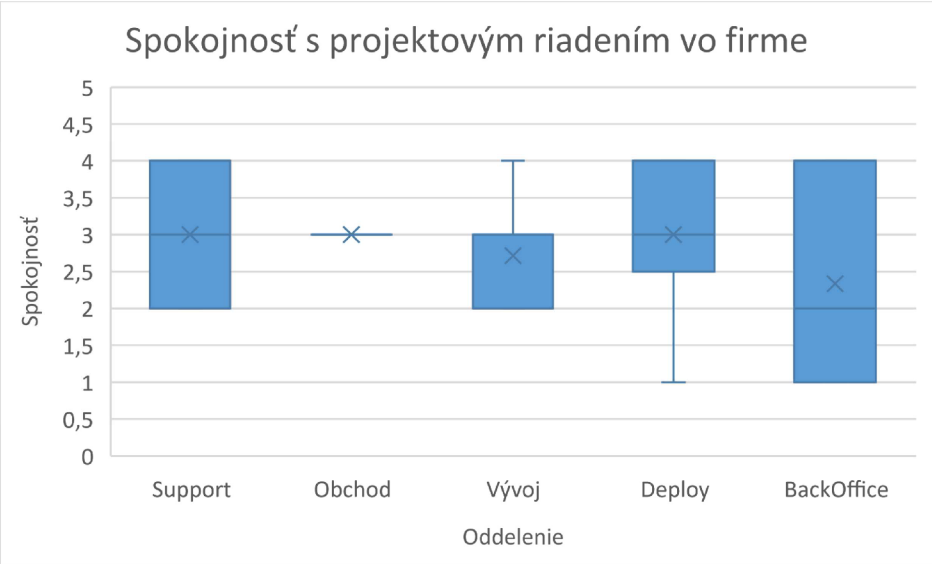
V súčasnosti sa však v tejto spoločnosti realizujú aj projekty komerčnejšieho charakteru, kde zákazník na samotnom začiatku nepozná presnú podobu požadovaného výsledku. V takomto prípade sa často jedná o agilný prístup s prvkami predikatívneho. V súčasnosti sú používané nástroje najmä *Redmine*[1] s rôznymi rozšíreniami a nástroj *Sharepoint*.

Prieskum spokojnosti

V rámci získania spätnej väzby bol v septembri roku 2018 vykonaný prieskum medzi všetkými oddeleniami, ktorého sa zúčastnilo 24 zamestnancov.

Na obrázku 5.1 je možné vidieť škatuľový graf znázorňujúci spokojnosť zamestnancov zozbieranú počas prieskumu. Zadávané hodnoty boli od 1 do 5, kde 1 znázorňuje najnižšiu spokojnosť a 5 najvyššiu spokojnosť so súčasným stavom projektového riadenia vo firme.

[1] https://www.redmine.org/



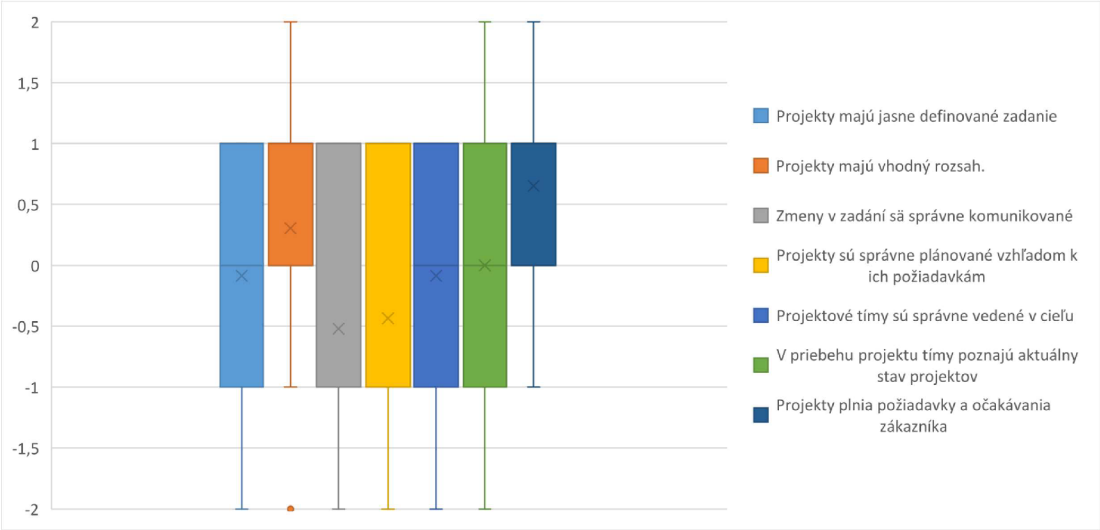
Obr. 5.1: Škatuľový graf zobrazujúci spokojnosť jednotlivých oddelení s projektovým riadením vo firme [vlastná tvorba]

Zobrazené hodnoty sú rozdelené podľa spomínaných oddelení, a tak je možné vyvodiť závery pre jednotlivé oddelenia. Priemerná hodnota je znázornená pomocou krížiku v jednotlivých škatuľkách. Najnižšiu mieru spokojnosti vykazuje backoffice oddelenie, čo je prekvapivý výsledok, nakoľko priamo nekooperuje s projektovým riadením.

Druhá najnižšia priemerná hodnota je potom pri vývojovom oddelení, čo je hodnotené ako problém. Všeobecne pri všetkých oddeleniach neprekračujú priemerné hodnoty priemernú spokojnosť, ktorá má hodnotu 3.

Tieto relatívne nízke čísla mali za následok následný prieskum konkrétnych oblastí projektového manažmentu. Časť výsledkov je možné vidieť na grafe z obrázku 5.2, ktorý zobrazuje výsledky otázok zameraných na jednotlivé oblasti projektového riadenia. 0 je v grafe neutrálna hodnota a hodnoty do 2 vyjadrujú spokojnosť a naopak, záporné označujú mieru nespokojnosti.

Tu je možné pozorovať v priemerných hodnotách (značených krížikom) najväčšiu nespokojnosť v oblasti riadenia zmien zadania a s podobnou nespokojnosťou je tu zaznačená miera nespokojnosti s riadením plánovania harmonogramu – v tomto kontexte vzhľadom k zadaniu.



Obr. 5.2: Graf zobrazujúci zobrazujúci spokojnosť so zameraním na oblasti projektového manažmentu [vlastná tvorba]

5.2 Charakteristika startupového prostredia

V praxi sa často používa definícia *startupu* podľa *Steava Blanka*, ktorý prišiel s nasledovnou definíciou pojmu *startup*. Jeho definícia sa dá rozdeliť na následné časti [24]:

- **Dočasný stav organizácie** – *Startup* nezostáva *startupom*. Buď takáto spoločnosť zanikne, alebo uspeje v uspokojení zákazníka a tí sú ochotní za riešenie zaplatiť.
- **Hľadanie** – Poslaním *startupu* je preskúmavať, skúšať a validovať nenaplnené požiadavky. Táto definícia potvrdzuje konečný životný cyklus *startupu*.
- **Opakovateľný a škálovateľný biznis model** – Všetky *startupy* sú založené na predpokladoch, s cieľom iterovania dokiaľ tieto predpoklady nie sú zvalidované. Ak sa podarilo dokázať, že biznis model môže fungovať a že *startup* je udržateľný nejedná sa už o *startup*.

Je potrebné si popísať aj špecifiká startupového prostredia, keďže aj popisovaná firma vznikla ako startup a niektoré „choroby“ startupu pravdepodobne pretrvávajú dodnes. Tieto špecifiká vznikli na základe vlastného pozorovania.

Startup je teda začínajúca firma, zväčša sa však takto označujú hlavne spoločnosti z oblasti technológií. V týchto prípadoch sa väčšinou jedná o jednotlivcov alebo malé skupiny ľudí, ktoré sa snažia vlastným produktom ponúknuť riešenie pre istú časť trhu.

Takýto jednotlivec alebo teda malá skupina ľudí pripomína svojou charakteristikou agilné tímy hneď v niekoľkých bodoch:

- jedná sa o skupinu jednotlivcov (zvyčajne 1 až max 10),
- neexistuje striktná hierarchia medzi členmi tímu,
- nutné zdieľanie vedomostí a znalostí,
- jednotlivci majú široký záber pôsobnosti – nemajú len úzku špecializáciu v rámci tímu,
- striktný postup nie je často špecifikovaný, keďže sa takýto tím snaží o doteraz neexistujúce riešenie nejakého problému,
- tím musí byť pripravený na zmeny v priebehu vývoja, keďže často vstupuje do prostredia či už mentor alebo investor, ktorý ovplyvňuje finálny produkt,
- startupy sa prvotných fázach sústredia na dodanie tzv. *MVP*[2], ktorý je možné vnímať ako prvú iteráciu pre získanie spätnej väzby napríklad od potenciálnych zákazníkov alebo investora.

Ďalšie charakteristiky mimo súvis s agilnými tímami sú:

- častá neskúsenosť so zložitými metódami manažmentu,
- pracovný čas členov tímu často nepredstavuje plný pracovný úväzok – členovia tímu môžu mať inú pracovnú činnosť ako zdroj príjmov, môžu to byť študenti atd.

Aj na základe týchto špecifických čŕt *startupového* prostredia je možné zvoliť lepší prístup k návrhu samotného prototypu.

[2] z angl. „Minimum Viable Product“ teda minimálny životaschopný produkt

Kapitola 6

Návrh prototypu

Predchádzajúca sekcia 5.2 o charakteristike *startupového* prostredia predstavila hneď niekoľko charakteristických čŕt, ktoré by mal daný prototyp spĺňať. Táto kapitola ich bližšie špecifikuje a doplní o ďalšie potrebné informácie.

Na daný prototyp je potrebné sa pozerať z pohľadu návrhu *produktu s najmenšou možnou funkcionalitou*[1]. Takýto návrh spočíva v stratégii, pri ktorej sa identifikuje len minimálna potrebná funkcionalita alebo vlastnosti produktu, ktoré budú hodnotné pre koncových užívateľov, ktorí budú produkt používať – ale nič viac [15].

Najprv je vhodné pozrieť sa na už existujúce riešenia alebo konkrétne aplikácie, ktoré ponúkajú približne podobnú ako očakávanú funkcionalitu a sú určené na podobný cieľ.

6.1 Analýza existujúcich riešení

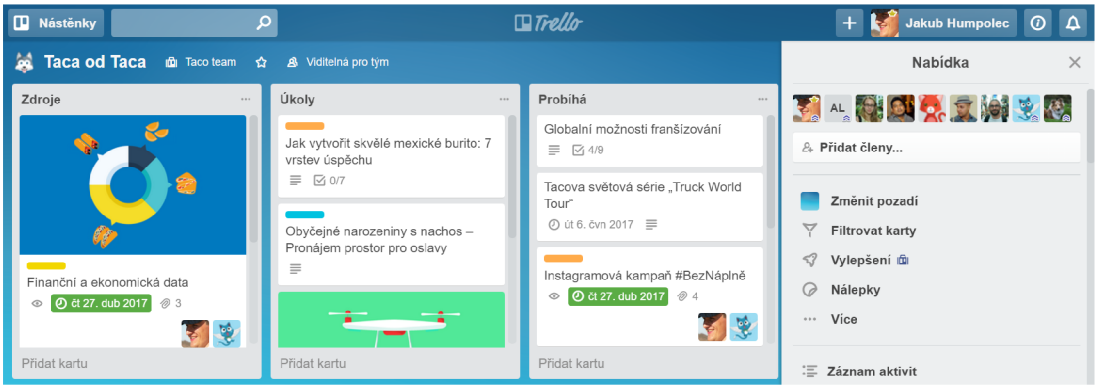
Táto sekcia sa zaoberá pohľadom na niektoré z najznámejších nástrojov z oblasti agilného riadenia projektov, často práve už spomínaného *Kanbanu*.

Trello

Prvým a asi najznámejším používaným nástrojom je nástroj *Trello*[2]. Ten v samotnej základnej podobe ponúka funkcie, ako *vytváranie tímov*, *vytváranie projektov* a základné úkony k vytvoreniu organizácie. V rámci samotnej práce, je potom možné vytvárať *nástenky*, ktoré už slúžia ako *Kanban nástenky* – teda je možné do stĺpcov (stavov) pridávať kartičky (úlohy). K týmto kartičkám je možné pridať rôzne parametre ako napr. popis, termín plánovaného dokončenia, priradených pracovníkov, komentáre a rôzne iné informácie. Ukážka popisovaného prostredia je na obrázku 6.1.

Trello ponúka množstvo druhov rozširovacích súčastí produktu, ako sú napr. prepojenie na komunikačné nástroje, verzovacie systémy, nástroje na podporu predaja a mnohé iné. V základnej (bezplatnej) verzii je však možné k nástenke pridať len jedno vybrané rozšírenie. Prihlasovanie sa vykonáva cez registráciu priamo na portály, pomocou *Google* účtu alebo pomocou prihlasovania cez organizáciu.

[1] z angl. MVP – *minimum viable product*
[2] https://trello.com

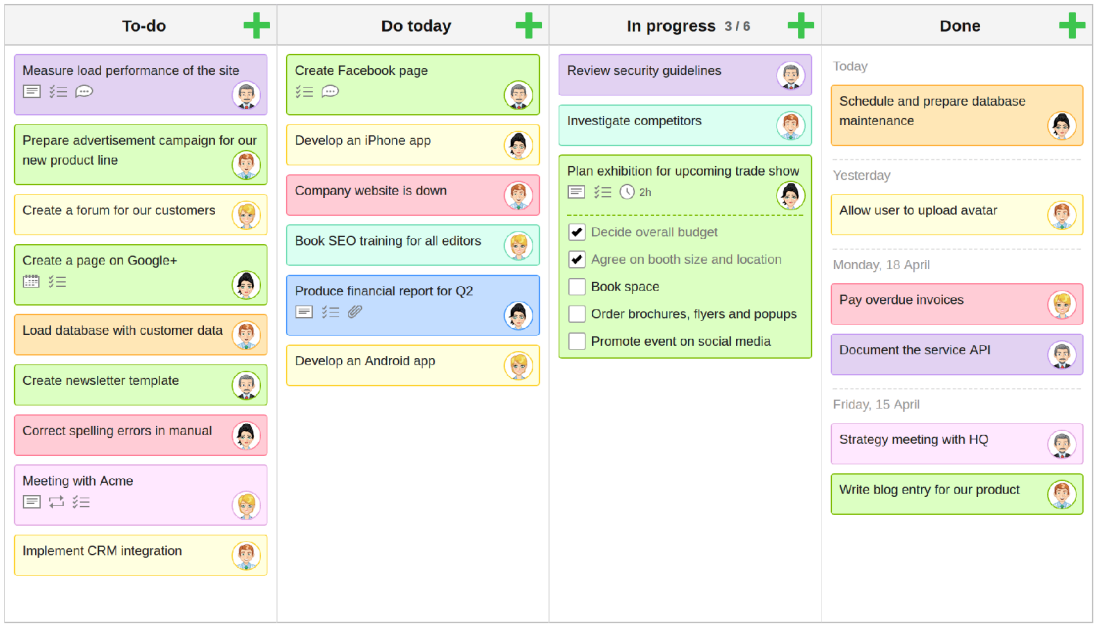


Obr. 6.1: Ukážka nástenky v nástroji *Trello* s úlohami (kartičkami) v rôznych stavoch (stĺpcoch)[4]

KanbanFlow

Ďalším nástrojom, sústrediacim sa na *Kanban* nástenky je nástroj *KanbanFlow*. Ten ponúka podobné možnosti ako *Trello* v rámci práce s tímami a nástenkami. Naviac pridáva napr. obmedzenie jednotlivých stavov (stĺpcov) nástenky len na určitý počet úloh v danom stave (kvôli lepšej sústredenosti na úlohy). Rovnako ponúka možnosť tvorby vzájomnej závislosti jednotlivých úloh.

Tento nástroj sa sústredí aj na časové sledovanie plnenia úloh v rámci priebežných štatistík, tiež zahrňuje aj analytické nástroje ako *burndonw* grafy, kumulatívne grafy a i. Širšie možnosti spomínaných funkcií sú v ponuke opäť len pri platenej verzii.



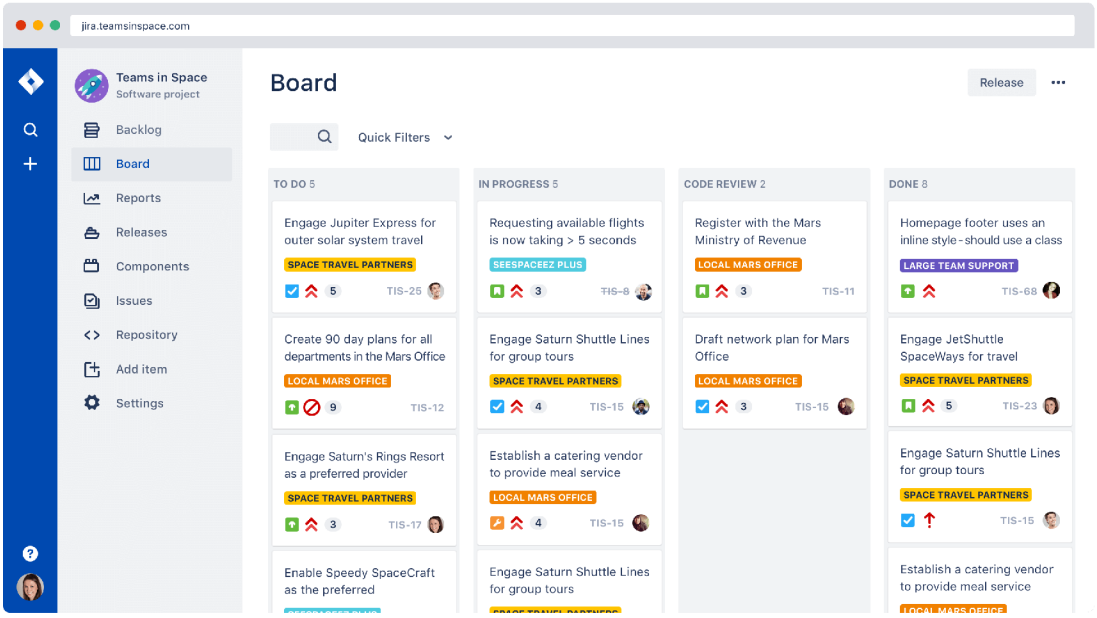
Obr. 6.2: Ukážka nástenky v nástroji *KanbanFlow* s úlohami (kartičkami) v rôznych stavoch (stĺpcoch)[6]

[4] https://trello.com/guide/create-a-board
[4] https://kanbanflow.com/press

JIRA Software

JIRA ponúka hneď niekoľko agilných prístupov pre užívateľov a to konkrétne *Scrum* nástenky alebo *Kanban* nástenky (ukážka na obrázku 6.3). Tak isto ponúka rôzne analytické nástroje vyhodnocované v reálnom čase. V rámci systému je často používaná aj funkcia prepojenia nahlásených problémov priamo na kód v repozitároch. *JIRA* opäť poskytuje možnosť napojenia na mnoho ďalších služieb a rôzne ďalšie rozšírenia. Tie však často nie sú v základnom bezplatnom balíčku služieb.

Mnoho *IT* firiem dnes používa práve tento softvér pre projektové riadenie, avšak začína sa vyskytovať až u malých až stredných firiem, ale pre mikro firmy je tento softvér často zbytočne komplikovaný.



Obr. 6.3: Ukážka nástenky v nástroji *JIRA Software* s úlohami (kartičkami) v rôznych stavoch (stĺpcoch)[8]

Ostatné

Existuje niekoľko ďalších možných alternatív k vyššie spomínaným nástrojom používaných v prostredí s agilným vývojom. Často sa v nich kombinujú jednotlivé vlastnosti vyššie spomenutých nástrojov s rôznymi obmenami, možnosťami doplnkov atď. Vyššie spomenuté nástroje by tak mali poskytnúť dostatočný prehľad, čo nástroje pre podporu riadenia obsahujú a dávajú priestor pre špecifikáciu požiadaviek vlastného riešenia aj v kombinácií s požiadavkami skúmanej spoločnosti zo sekcie 5.1.

6.2 Špecifikácia požiadaviek

Po predchádzajúcich analýzach skúmanej firmy zo sekcie 5.1, ale aj všeobecne prostredia začínajúcich firiem z oblasti technológií, je vhodné ako základ brať do úvahy hlavne agilné

[8] https://www.atlassian.com/software/jira

metódy. Rovnako je vhodné brať do úvahy poznatky získané analýzou existujúcich riešení. Z nich je možné získať informácie o najpoužívanejších spôsoboch v oblasti nástrojov, ktoré slúžia na riadenie projektov vo všeobecnosti.

Základná podoba

Z hľadiska návrhu nástroja aplikácie odporúčame brať ako základ spomínanú vizuálnu metódu *Kanban*. Malo by sa teda jednať o nástroj podporujúci *úlohy* v podobe *kartičiek* na akejsi *tabuli* so stĺpcami, ktorý bude znázorňovať stav, v akom sa daná *úloha* nachádza. V súčastnosti patrí k najznámejším nástrojom s danou charakteristikou aplikácia *Trello*[9].

Pre optimálne využitie a napasovanie na konkrétneho užívateľa je vhodné aplikáciu rozšíriť o špecifiká skúmaného prostredia a ďalšie požadované vlastnosti. V rámci skúmanej firmy bola vyšpecifikovaná požiadavka na umiestnenie dát - a to konkrétne, aby mala firma **plný prístup k uloženým dátam** a prípadnú možnosť manipulovať s nimi podľa vlastných predstáv.

Požiadavky na funkčnosť

Medzi stĺpcami tabuľky bude môcť užívateľ, v zmysle *Kanbanu*, presúvať úlohy. Tie budú mať stav, ktorý bude znamenať v akej fáze sa daná úloha nachádza. Budú musieť mať tak isto priradeného vlastníka úlohy a predpokladaný časový interval potrebný pre dokončenie tejto úlohy.

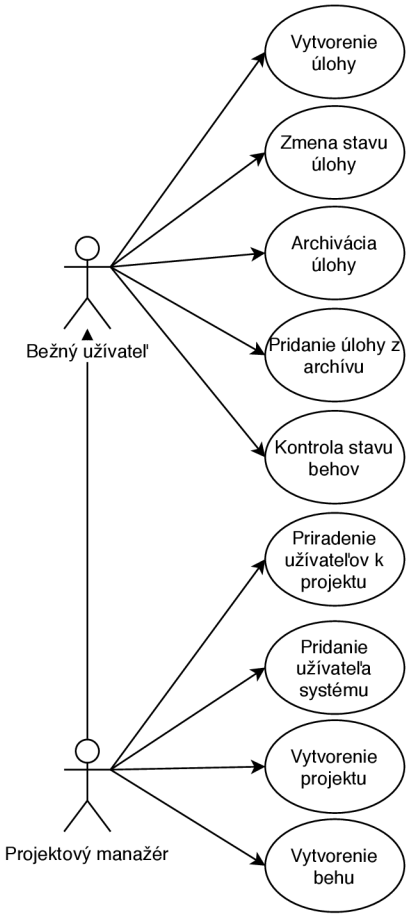
Úlohy by mali byť rozdelené podľa priradených členov tímu aby bolo možné jednoducho určiť vlastníka úlohy. Tak isto budú úlohy delené podľa iterácií, v ktorých majú byť vykonané a úlohy budú mať poradie pre vykonanie.

Na jednotlivé iterácie bude určený celkový čas, v ktorom je predpokladané dokončenie tejto iterácie, podľa čoho bude prototyp zobrazovať, či sa daná úloha stíha alebo nestíha vykonať v danej iterácií.

Prípady použitia

Jednotlivé požiadavky boli spracované a bol zostrojený nasledovný názorný diagram prípadov použitia 6.4, ktorý zobrazuje jednotlivé role s činnosťami, ktoré by mal pre danú rolu systém podporovať.

[9] https://www.trello.com



Obr. 6.4: Diagram zobrazujúci prípady použitia systému. [vlastná tvorba]

Role zobrazené v spomínanom diagrame a činnosti s podrobnejším popisom sú:

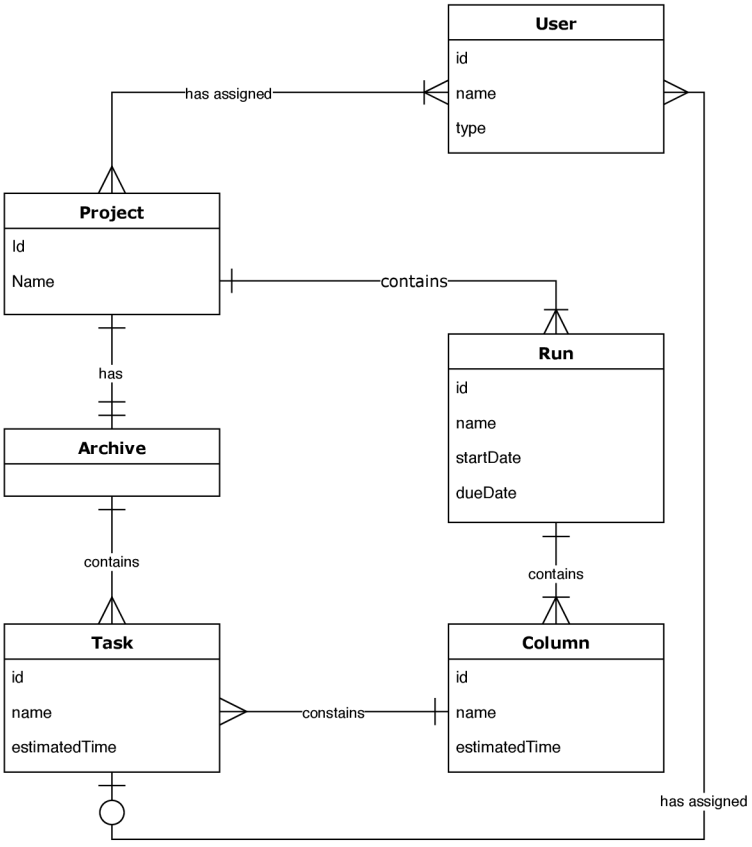
- **Bežný užívateľ** – užívateľ, ktorého úlohy v rámci systému budú hlavne základné činnosti súvisiace s činnosťami v rámci behov:
  - **Vytvorenie úlohy** – vytvorenie úlohy pre stav (stĺpec) behu (Kanbanu).
  - **Zmena stavu úlohy** – presunutie úlohy do iného stavu (stĺpca) behu (Kanbanu).
  - **Archivácia úlohy** – presunutie úlohy do tzv. archívu, ktorý je spoločný pre daný projekt, čiže archív môže slúžiť pre presun úloh medzi behmi projektu.
  - **Pridanie úlohy z archívu** – presunutie úlohy z archívu do stavu behu projektu.
  - **Kontrola stavu behu** – kontrolovať stav behu je možné pomocou grafu typu *burn-down* alebo pomocou grafu, ktorý proporčne zobrazuje úlohy behu v jednotlivých stavoch (stĺpcoch) behu (Kanbanu).
- **Projektový manažér** – užívateľ vyššieho stupňa, ktorého činnosť bude pozostávať z činností *bežného užívateľa*, plus bude rozšírený o činnosti súvisiace s riadením projektov v rámci organizácie:
  - **Pridanie užívateľa** – v rámci systému bude môcť projektový manažér pridávať užívateľov všetkých druhov (*bežný užívateľ* aj *projektový manažér*).

– **Priradenie užívateľov k projektu** – ku každému projektu bude možné pridávať užívateľov samostatne, bez využitia tejto možnosti má prístup k projektu len *projektový manažér*, ktorý daný projekt vytvoril.
– **Vytvorenie projektu** – do systému bude môcť užívateľ tohto typu pridávať nové projekty.
– **Vytvorenie behu** – do projektov bude môcť užívateľ pridávať behy pre ďalšie fázy projektov, ktoré budú mať nejaké trvanie a bude možné pre ne sledovať ich stav.

Z diagramu prípadov použitia sú už na prvý pohľad viditeľné niektoré entity a vzťahy medzi nimi.

Diagram vzťahu entít

Pomocou definície entitne-vzťahového diagramu [3] a z analýzy prípadov použitia je možné následne zostrojiť diagram vzťahov entít, ako je zobrazený na nasledovnom obrázku 6.5.



Obr. 6.5: Diagram vzťahu entít. [vlastná tvorba]

Hlavné entity systému teda sú: *projekt*, *archív*, *úloha*, *užívateľ* (jeho typ je určený pomocou atribútu), *beh* a *stĺpec*.

Z tohto návrhu bude neskôr vychádzať samotný návrh databázy ktorý bude prebiehať v rámci kapitoly implementácie 7.

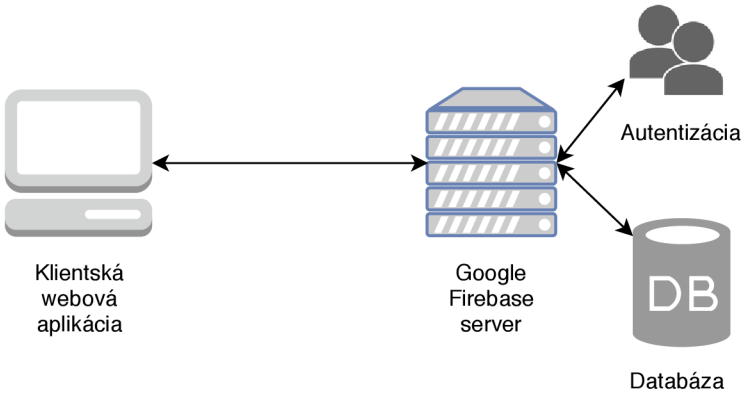
Požiadavky na prostredie

Vhodná platforma pre prototyp aplikácie bude webová aplikácia, keďže nebude nutné hľadieť na operačný systém a aplikácia bude ľahko prenosná medzi operačnými systémami.

Ako referenčné podporované prostredie je vybraný prehliadač *Google Chrome* vo verzii *73.0.+*. Minimálne rozlíšenie zobrazovaného okna bolo stanovené na *1280 x 720* pixlov.

6.3 Návrh architektúry

Na základe vyššie popísaných požiadaviek na systém bolo možné navrhnúť požadovanú architektúru. Pre implementáciu je zvolená tzv. bez-serverová architektúra, ktorá dokáže fungovať bez komunikačného prostredníka, teda webová aplikácia komunikuje priamo s časťami serveru, ktoré potrebuje [9].



Obr. 6.6: Príklad navrhnutej obrazovky - konkrétne obrazovka zobrazujúca úlohy v behu – formou Kanbanu. [vlastná tvorba]

Ako zobrazuje aj obrázok 6.6, architektúra bude teda pozostávať z dvoch hlavných častí a to *klientskej webovej aplikácie*, ktorá bude komunikovať s druhou z hlavných častí, ktorou je server *Google Firebase*. *Google Firebase* je server, ktorý poskytuje rôzne služby podporujúce vývoj aplikácií vo všeobecnosti. Pre účely tejto práce však budú využité hlavne služby *autentifikácie užívateľov* a služby tzv. *NoSQL databázy*. Využitie týchto služieb bude bližšie popísané v kapitole 7.

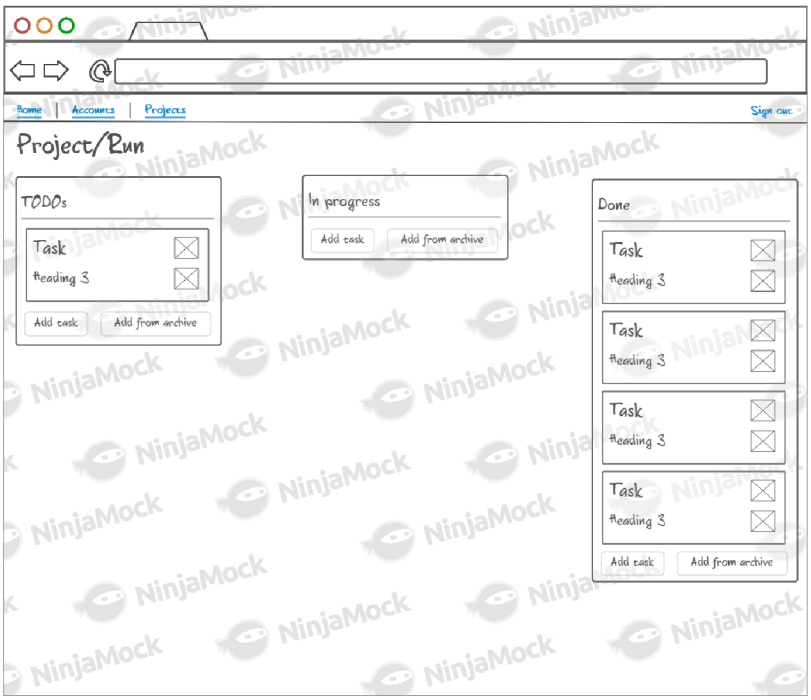
6.4 Návrh užívateľského rozhrania

Z predchádzajúcej sekcie 6.2, v ktorej boli špecifikované formálne požiadavky na systém je možné preklenúť návrh do návrhu užívateľského rozhrania. Ten bude vytvorený pomocou tzv. *drôteného modelu*.

Vytváranie týchto modelov patrí k významným činnostiam tvorby webového dizajnu. Ide o formu zobrazenia základného rozloženia prvkov užívateľského rozhrania – často sa používa analógia so stavebnými plánmi. Takéto návrhy nie je nutné vytvárať pre každú stránku samostatne, ale často sa vytvárajú len modely unikátneho rozostavenia prvkov užívateľského rozhrania [11].

Z toho vyplýva aj návrh tejto aplikácie v podobe *drôtených modelov*. To je dosiahnuté pomocou nástroja *Ninjamock*[10]. V ňom, sú v zmysle *drôtených modelov* zobrazené hlavné grafické prvky jednotlivých obrazoviek potrebných pre následnú implementáciu [4]. V tomto nástroji sú načrtnuté jednotlivé obrazovky pre hlavné časti systému, ktoré sú následné:

- **Prihlasovacia obrazovka** - táto obrazovka bude slúžiť k autentifikácii užívateľov systému a následný vstup do systému.
- **Obrazovka užívateľov** – na tejto obrazovke bude možné hlavne pridávať nových užívateľov do systému.
- **Obrazovka projektov** – táto obrazovka bude slúžiť k výberu alebo založeniu nového projektu. V zozname projektov sa zobrazia užívateľovi priradené projekty a každý projekt bude mať možnosť nejakej interakcie, či už výber samotného projektu alebo napr. priradenie užívateľov k projektu.
- **Obrazovka behov projektu** – táto obrazovka bude slúžiť k výberu behu v rámci projektu, v rámci ktorých bude môcť vykonať akcie súvisiace s behmi projektu. Takýmito akciami sa rozumie napr. kontrola stavu behu projektu, ktoré budú viesť k zobrazeniu pomocných grafov.
- **Obrazovka behu projektu** – táto obrazovka bude slúžiť ako obrazovka pre prácu s jednotlivými behmi projektu. V rámci týchto činností bude možné napríklad: meniť stav úloh v rámci behu, archivovať úlohy, pridávať úlohy z archívu alebo vytvárať úlohy. Ukážka návrhu tejto obrazovky je aj na obrázku 6.7.



Obr. 6.7: Príklad navrhnutej obrazovky - konkrétne obrazovka zobrazujúca úlohy v behu – formou Kanbanu. [vlastná tvorba]

[10] https://ninjamock.com

Všetky grafické návrhy v obrázkovej podobe sú umiestnené v prílohe A.

6.5 Spätná väzba k návrhu užívateľského rozhrania

Grafický návrh aplikácie bol vytvorený v nástroji *Ninjamock*, ktorý umožňuje vytvoriť klikateľný model a zdielať takýto model s užívateľmi.

Pomocou *drôteného modelu* boli užívatelia oboznámený s návrhom prototypu systému a získali tak predstavu o myšlienke navrhovaného systému. Z dotazníku (uvedeného s odpoveďami v prílohe B) vyplynuli rôzne predpoklady pre ďalšie verzie systému.

Od respondentov, ktorí boli z rôznych pracovných prostredí a rôznych pracovných pozícií vzišlo široké spektrum odpovedí na pokladané otázky.

6.5.1 Vzorka užívateľov

Užívatelia pre testovanie *drôteného modelu* boli vybraný zo širokého okruhu firiem s rôznym zameraním a na rôznych pracovných pozíciach. Konkrétne išlo o pozície *prevádzkového riaditeľa*, *výkonného riaditeľa startupu*, *vývojára backendu*, *testera užívateľských rozhraní*, *projektového manažéra* a *Java vývojára*. Jednalo sa o ľudí pôsobiacich v rôznych firmách, rôznych veľkostí, teda s predpokladom iných pracovných návykov a zvyklostí v rámci firemného prostredia.

6.5.2 Funkcionalita pre ďalšie verzie

Prvou časťou dotazníka bola otázka na požadovanú funkcionalitu pre ďalšie verzie navrhovaného systému. Z dotazníkov vyplynuli hlavne tieto požiadavky:

- stĺpce (stavy úloh) – viac druhov stĺpcov, vlastné názvy stĺpcov,
- užívateľské role – viacero druhov užívateľských rolí – v rámci organizácie napr. riaditeľ alebo technický riaditeľ s prístupom do všetkých projektov organizácie s prípadnými štatistikami,
- možnosti rozdelenia projektov podľa zamerania oddelenia,
- možnosť zmeny farebnej témy prostredia (konkrétne často vyhľadávaná tmavá téma).

V rámci spätnej väzby sa v odpovediach z dotazníku vyskytli aj logické požiadavky, no mimo zámer projektu:

> *„Nějaký nástroj pro získávání zpětné vazby k projektu od zákazníka“*

V odpovediach sa vyskytla aj odpoveď na tzv. *otvorené API*[11], čo by znamenalo sprístupnenie dát alebo akcii pre ďalších vývojárov. To by mohlo mať funkcionalitu napr. presúvanie úloh v rámci behu, pridávanie úloh do behov s naviazaním na externý zdroj (napr. repozitáre) atď. Tým je možné sa premostiť na ďalšiu kategóriu požiadaviek uvedených v nasledujúcej sekcii.

6.5.3 Integrácia s inými službami

V otázkach o integrácii – ale nie len tam – užívatelia spomínali možnosť integrácie s rôznymi druhmi služieb. Integráciu požadovanú užívateľmi je možné rozdeliť na niekoľko kategórii.

[11] skratka pre programové rozhranie z angl. *Application Programming Interface*

Organizéry

V rámci integrácie s organizérmi alebo teda organizačným softvérom, sa môže jednať o rôzne podporované služby rôzneho charakteru. Môže sa jednať napr. o export do kalendára (často je používaný *Google Kalendár*), nástroja *Toggle*[12] a pod.

DevOps nástroje

Samotný význam tzv. *DevOps* je prístup k vývoju softvéru s dôrazom na komunikáciu, spoluprácu a integráciu medzi vývojárom a odborníkmi v IT [17]. Môžu to byť nástroje ako *Azure DevOps*[13] alebo napr. *Jira Software*[14].

Verzovacie systémy

V prípade verzovacích systémov existuje široká škála rôznych poskytovateľov a podôb týchto systémov, medzi najznámejšie patria napr. *Bitbucket*[15], *GitLab*[16] a rôzne iné.

6.5.4 Rozšírenie podľa spätnej väzby

Na základe spätnej väzby od užívateľov bola rozšírená navrhovaná verzia systému o tieto prípady:

- Editácia existujúcich projektov – pre prípad zmeny informácií o vytvorenom projekte bude možné editovať vytvorený projekt. Taktiež boli pridané polia *dátumu začiatku* a pole *konca projektu* pre prípadné Ďalšie štatistiky v budúcnosti. Tento prípad použitia sa týka užívateľa typu *projektový manažér*.
- Editácia existujúceho behu projektu – podobne ako vyššie spomenutá editácia projektov, editácia behu projektu bude mať za cieľ zmeniť informácie o danom behu po jeho vytvorení.
- Editácia existujúcich úloh – rovnako ako v predchádzajúcich prípadoch sa jedná o upravenie už existujúcich úloh v rámci systému.
- Možnosť výberu stĺpca (stavu) behu (nástenky) projektu – pre každý beh projektu bude možné v rámci filozofie *Kanbanu* vyberať z viacerých stĺpcov pre presúvanie úloh. Ponuka bola rozšírená na *Backlog*[17], *Potrebná analýza*, *K práci*, *Prebieha práca*, *Potrebné testy*, *V testoch* a posledný stav je stav *Hotové*. Stĺpce sú presne v tomto poradí a z týchto stavov sú stavy *K práci* a *Hotové* zvolené vždy. Spomínané stavy boli zvolené opäť podľa filozofie *Kanbanu*, kde by úlohy mali meniť stav hlavne zľava doprava a jednotlivé názvy stĺpcov (stavov) boli vybrané na základe skúseností s prácou na vývoji produktov.

Tieto prípady použitia rozširujú užívateľa typu *projektový manažér*.

[12] https://toggl.com/
[13] https://azure.microsoft.com/en-us/services/devops/
[14] https://www.atlassian.com/software/jira
[15] https://bitbucket.org
[16] https://gitlab.com
[17] v preklade *resty*

Kapitola 7

Implementácia

Táto kapitola je venovaná implementácii samotnej webovej aplikácie. Sekcia 6.3 popisovala aplikáciu z hľadiska architektúry, na základe čoho je možné selektovať vhodné prostriedky implementácie.

Na základe návrhu užívateľského prostredia (sekcia 6.4) bude možné vyberať vhodné knižnice pri vytváraní užívateľského rozhrania.

7.1 Klientská aplikácia

Jedna z častí implementácie, ako vyplýva aj z návrhu architektúry v kapitole 6.3, je práve klientská aplikácia.

7.1.1 React

Podstatnou súčasťou implementácie bol výber vhodného aplikačného rámca nad jazykom *Javascript*. Použitý je aplikačný rámec *React*. Ten samotný je knižnicou *Javascriptu* pre vytváranie primárne užívateľských rozhraní, vyvíjaný spoločnosťou *Facebook*. Tá je orientovaná na tzv. *DOM(z angl. Document Object Model)* a koncentruje sa na vytváranie samotných prvkov užívateľského rozhrania [7].

Vytvára základ aj pre samotnú súborovú štruktúru, kde je možné zdrojové kódy hierarchicky rozdeliť podľa už navrhovaných obrazoviek samotnej aplikácie, ako boli ukázané v sekcii 6.4, ale aj v rámci nich je možné rozdeliť obrazovky na väčšie celky – komponenty.

7.1.2 Typescript

Programovací jazyk *Typescript*, vyvíjaný spoločnosťou *Microsoft*, je nadstavbou programovacieho jazyka *JavaScript* a prináša so sebou niekoľko výhod.

Medzi základné výhody patrí skoré odchytávanie vytváraných chýb v kóde. V praxi to znamená, že vo väčšine editorov (napr. *Visual Studio Code* alebo *WebStorm*) sú chyby zvýraznené hneď po ich napísaní do kódu [18].

Pre využitie do budúcna a prípadné rozširovanie podpory aplikácie aj pre iné prehliadače je *Typescript* dobrou voľbou z dôvodu podpory *Javascriptových* funkcií naprieč prehliadačmi. To znamená, že pri transpilácii kódu do *JavaScriptu* prebieha zápis funkcií spôsobom, aký podporuje daný prehliadač [18].

Nakoľko je jazyk staticky typovaný je taktiež dosiahnuté zvýšenie produktivity vývojára a týmto je tak isto možné odbúrať možné chyby pri programovaní [18].

7.1.3 CXS

V súvislosti s rozdelením zdrojového kódu na komponenty súvisí aj rozdelenie štýlovacích súborov podľa komponent užívateľského rozhrania. Na tento účel je použitá knižnica *CXS*[1]. Tá okrem iného ponúka aj znovupoužívanie opakovaných štýlov, zníženie počtu súborových závislostí alebo zvýšenie výkonu.

7.1.4 Material UI

Ďalšou podpornou knižnicou je knižnica *Material UI*[2]. Ako napovedá názov, jedná sa o knižnicu obsahujúcu prvky tzv. *Material Designu*.

Ten priniesla v roku 2014 spoločnosť *Google*, ktorý sa odvtedy presadil ako systematický súbor dizajnovej filozofie. Tá by sa dala predstaviť ako akýsi „smart“ papier. Rovnako ako papier, sa v tomto dizajne uplatňujú princípy povrchu, tieňov ako pri skutočnom papieri. Ten bol rozšírený ešte o rôzne animácie, zmeny tvarov a rôzne iné vylepšenia [13].

Z prvkov *Material Designu* boli použité hlavne:

- Navigačná lišta – hlavný ovládací prvok, pomocou ktorého je možná navigácia v rámci aplikácie.
- Papierové kartičky – táto komponenta bola využitá hlavne ako odlíšenie prvkov grafického rozhrania pre vypĺňané formuláre.
- Dialógové okná – dialógové okná boli použité väčšinou v kombinácií s predchádzajúcim prvkom *papierových kartičiek* a to pre zadávanie nových údajov v rôznych situáciách.
- Zoznamy – dizajn zoznamov bol použitý najmä, ako aj vyplýva z návrhu, v prípadoch zobrazovaných zoznamov projektov alebo samotných behov projektov.
- Tlačidlá – na tlačidlá v rámci celej aplikácie bola použitá podpora práve z knižnice *Material UI*.

Pri implementácii boli použité aj iné ďalšie grafické prvky z popisovanej knižnice alebo aspoň jej časti v rámci zachovania dizajnového konceptu.

Obrázok 7.1 zobrazuje použitie *papierovej kartičky* s prvkami pre zadávanie textu, výber dátumu a pre výber položiek zo zoznamu. Jedná sa o prípad pridávania behu projektu.

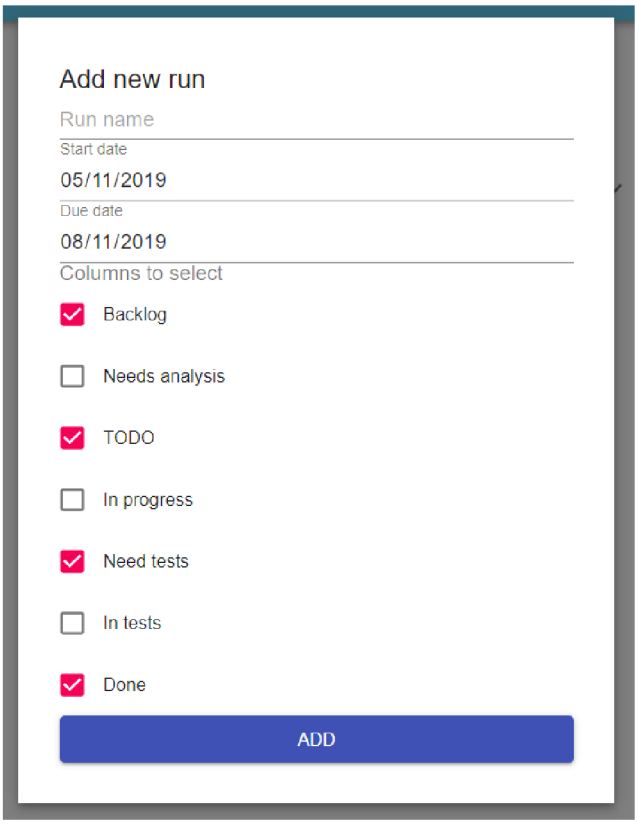
7.1.5 Recharts

Pre vykresľovanie grafov je v aplikácii použitá knižnica *Recharts*[3], ktorá umožňuje vykresliť relatívne širokú paletu rôznych druhov grafov. Naviac tieto grafy je možné dosť obšírne meniť podľa vlastných požiadaviek, čo môžu byť napr. zmeny mierok osi alebo rôzne doplnkové ukazovatele (ako je pomocná vertikálna čiara pre graf typu „burn-down“ atp.) [12].

Konkrétne bola táto knižnica použitá v dvoch prípadoch a to:

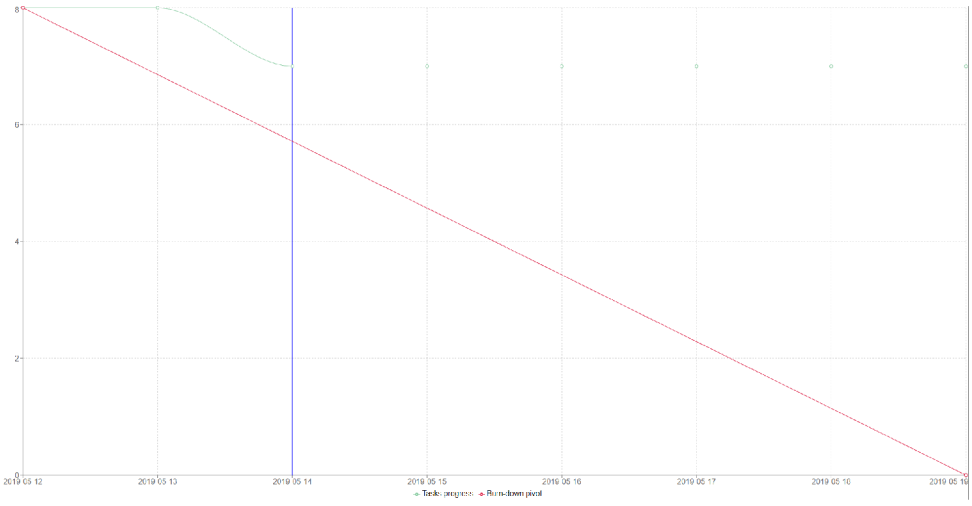
- graf *burn-down* – Potrebné dáta boli vypočítané na základe počtu úloh presunutých do konečného stavu v daný deň rozmedzia trvania behu projektu. Ukážka takéhoto grafu je na obrázku 7.2.

[1] http://jxnblk.com/react-cxs/
[2] https://material-ui.com/
[3] http://recharts.org



Obr. 7.1: Príklad použitia grafických elementov [vlastná tvorba]

- koláčový graf – Dáta potrebné pre vykresľovanie tohto grafu boli získané jednoduchým sčítaním odhadovaných časov úloh v jednotlivých stavoch behu projektu.



Obr. 7.2: Príklad použitia knižnice [vlastná tvorba]

7.1.6 Ostatné

V rámci implementácie navrhnutého systému boli použité aj ďalšie knižnice, ktoré dopomohli dosiahnuť požadovaný výsledok. Napríklad bola použitá knižnica *react-beautiful-dnd*[4] ako základ pre prácu s úlohami v rámci stĺpcov *Kanban* tabule v rámci behu projektu alebo podporná knižnica *moment*[5] pre uľahčenie práce s dátumami, s ktorými bolo nutné pracovať hlavne v prípade výpočtu dát pre vykresľovanie grafov.

7.2 Firebase

Oproti tradičným riešenia *backendu*, toto *cloudové* riešenie prináša zo sebou mnoho návrhových a architektonických noviniek. *Firebase* funguje ako *backend ako služba (BaaS)*[6], čo je prínosné hlavne z dôvodu odbúrania zbytočne zdĺhavého nastavovania. Takže je možné lepšie sa sústrediť na samotnú aplikáciu [25].

To znamená, že na pridanie služieb *Firebase* do projektu stačí vo webovej konzole *Firebase* vytvoriť projekt (ak už nie je vytvorený). Ďalej je potrebné zvoliť zadanie webovej aplikácie na pridanie a následne by mal byť poskytnutý krátky konfiguračný kód pre konfiguračný súbor webu. Tento kód nesie informácie pre verifikáciu, databázu, prípadne ďalšie služby *Firebase* [22]. Uvedená možnosť jednoduchej konfigurácie pomocou pár riadkov kódu môže slúžiť v neskorších prípadoch, ak by sa aplikáciu rozhodlo používať viacero subjektov s tým, že každý chce dáta aplikácie pod svojím účtom – aplikácia nemusí byť verejne prístupná.

7.2.1 Autentifikácia

Tak ako mnoho webových aplikácii je potrebné k systému pridať možnosť autentifikácie kvôli ochrane obsahu pred neželaným zneužitím. V rámci služieb *Firebase* sú možnosti autentifikácie dosť široké. Dostupné sú možnosti prihlásenia prihlásenie pomocou e-mailu a hesla. Tak isto je možné využiť na prihlásenie aj služby tretích strán ako sú napr. *Facebook*, *Google*, *Twitter* alebo *GitHub*. Novou možnosťou je prihlásenie sa aj pomocou telefónneho čísla [21].

V rámci implementovanej aplikácie je autentifikácia možná len pomocou e-mailu a hesla, ale do budúcna bude možné v prípade potreby relatívne jednoducho pridať ďalšie možnosti prihlasovania do webovej aplikácie.

7.2.2 Databáza

Firebase tiež v rámci svojich služieb ponúka vlastnú databázu. Konkrétne sa jedná o tzv. *reálno-časovú NoSQL* databázu, ktorá vie synchronizovať svoj stav s každým napojeným klientom. *Firebase* databáza, na rozdiel od klasických (požiadavka-odpoveď) databáz, používa synchronizačný mechanizmus, ktorý synchronizuje dáta v rámci milisekúnd na všetkých pripojených zariadeniach [20].

Databáza tiež poskytuje možnosť úpravy prístupových práv, ktorými je možné zabezpečiť dáta. Tieto práva je možné nastaviť pre rôzne úrovne dát v databázy. Potrebné nastavenia je možné vykonať cez webové rozhranie služieb *Firebase* [20].

[4] https://github.com/atlassian/react-beautiful-dnd
[5] https://www.npmjs.com/package/react-moment
[6] z angl. Backend-as-a-service

Tak ako je možné vidieť aj na ukážke 7.1, databázu je možné si predstaviť ako JSON súbor, v ktorom sa nachádzajú rôzne *uzly* s *pod-uzlami* obsahujúce dáta atp. Práca s databázou následne prebieha na základe referencií na uzly a pod-uzly. Spomínaný koncept databázy sa potom následne prenáša do niektorých myšlienok v rámci návrhu databázy v nasledujúcej sekcii.

Návrh databázy

Ako je uvádzané v prechádzajúcom odseku, databázu bolo potrebné rozdeliť na uzly. Tieto uzly bolo potrebné vytvoriť s ohľadom jednak na náročnosť na objem dát v databázy, ale aj s ohľadom na neskorší dátový prenos na základe dotazovania sa na referenciu databázy. Výsledný návrh je síce menej priestorovo úsporný, ale dátový prenos je optimálny vzhľadom na potreby webovej aplikácie.

Databáza sa skladá z nasledujúcich hlavných uzlov:

- `archive` – uzol obsahujúci kľúče, ktoré zodpovedajú identifikátorom projektov, pod ktorými sa nachádza zoznam úloh v archíve daného projektu.
- `projects` – uzol obsahujúci kľúče identifikátorov projektu, pod ktorými sa nachádzajú objekty so základnými informáciami o projektoch.
- `runs` – uzol obsahujúci kľúče projektov, ktoré majú pod sebou obsiahnuté kľúče k behom projektu. Tie následne obsahujú informácie o jednotlivých behoch projektov.
- `tasks` – uzol, ktorý obsahuje kľúče projektov, ktoré majú pod sebou kľúče k behom daného projektu. Tie následne obsahujú identifikátory stĺpcov (stavov) daného behu, ktoré už následne obsahujú zoznam objektov úloh, ktoré k nim patria.
- `users` – uzol obsahujúci identifikátory užívateľov ako kľúče, pod ktorými sa nachádzajú objekty jednotlivých používateľov.

Názorný príklad formátu uzlov databázy je uvedená v ukážke 7.1.

```
{
    "archive" :{
        "projectId" :
            [TASK],
        ...
    },
    "projects" :{
        "projectId" :{
            ...
        },...
    },
    "runs" :{
        "projectId" :{
            "runId": {
                "columns": [runId],
                "dueDate": Date,
                "startDate": Date
            },...
        },...
    },
    "tasks" :{
        "projectId" :{
            "runId": {
                "columnId": [TASK],
                ...
            },...
        },...
    },
    "users" :{
        "userId" :{
            "accountType": Type,
            "email": Email,
            "projects": [PROJECT_ID],
            "username" :Name
        },...
}
```

Ukážka 7.1: Jednoduchá ukážka databázy a jej hlavných uzlov s poduzlami.

Následná práca s takouto databázov môže vyzerať ako na ukážke 7.2, kde je možné vidieť referenciu na úlohy k behu projektu, kde sú parametre `projectId` ako identifikátor projektu a `runId` ako identifikátor behu projektu.

```
export function getRunTasks(projectId: string, runId: string) {
    return db.ref(`tasks/${projectId}/${runId}`).once("value")
}
```

Výpis 7.2: Ukážka referencie na čítanie z databázy.

Na ďalšej ukážke 7.3 je možné vidieť prácu pri zápise do databázy. Konkrétne sa jedná o zápis objektu behu projektu pod uzol `runs`, kde `projectId` opäť zodpovedá identifiká-

toru projektu a `runId` zodpovedá identifikátoru behu projektu. `run` je potom objekt behu projektu.

```
export function doCreateRunForProject(projectId: string, runId: string, run: IRun) {
    db.ref(`runs/${projectId}/${runId}`).set(run)
}
```

Výpis 7.3: Ukážka práce pri zápise do databázi.

Kapitola 8

Testovanie a zhodnotenie prvých výsledkov

Po implementácii tzv. *produktu s najmenšou možnou funkcionalitou* je potrebné dať systém otestovať užívateľom, ktorí môžu byť potenciálni používatelia v budúcnosti. Po otestovaní aktuálnej funkcionality je tak isto vhodné zozbierať spätnú väzbu od užívateľov na vývoj ďalších verzii aplikácie.

8.1 Užívateľské testovanie

V rámci užívateľského testovania bolo vytvorené testovacie prostredie (popísané v sekcii 8.1.1), v rámci ktorého testovací užívatelia testovali implementovanú aplikáciu za pomoci testovacích prípadov (popísaných v sekcii 8.1.2).

8.1.1 Testovacie prostredie

Aplikácia bola testovaná v lokálnom prostredí s inicializovanou databázou a aspoň jedným existujúcim užívateľom typu *projektový manažér* v rámci databázy. Ďalšie aspekty prostredia boli v súlade s definovanými požiadavkami z kapitoly 6.2 o špecifikácií požiadaviek na prostredie.

8.1.2 Testovacie prípady

Testovacím užívateľom boli k testovaniu dodané testovacie prípady, podľa ktorých mali daný systém otestovať a overiť tak jeho funkčnosť.

V testovacom protokole uvedenom v tabuľke 8.1 sú uvedené testy, ktoré majú za úlohu otestovať funkčnosť činností všetkých druhov užívateľov týkajúcu sa užívateľských účtov.

Scenár	Podmienky	Očakávaný výsledok	Stav
Registrácia užívateľa	Užívateľ môže byť prihlásený aj odhlásený	Užívateľ vytvorí užívateľa ale len jedného typu – *bežný užívateľ*	OK
Prihlásenie užívateľa	Inicializovaný užívateľ	Po zadaní korektných údajov je užívateľovi sprístupnená funkcionalita systému	OK
Obnovenie hesla užívateľa	Užívateľ môže byť prihlásený aj odhlásený	Po zadaní e-mailovej adresy bude užívateľovi odoslaný na e-mail obnovovací formulár	OK
Zmena hesla účtu	Užívateľ prihlásený v systéme	Po zadaní aktuálneho hesla môže užívateľ zvoliť svoje nové heslo	OK

Tabuľka 8.1: Užívateľské testy – užívateľský účet (všetci užívatelia)

V ďalšej tabuľke 8.2 sú uvedené testovacie prípady týkajúce sa všetkých užívateľov. V tomto prípade sa jedná o testy spojene s činnosťou práce s projektami a behmi projektov.

Scenár	Podmienky	Očakávaný výsledok	Stav
Zobrazenie priradených projektov	Užívateľovi je priradený aspoň jeden projekt	Užívateľ má sprístupnený zoznam jemu priradených projektov	OK
Zobrazenie behov projektu	Užívateľovi je priradený aspoň jeden projekt a v rámci neho existuje aspoň jeden beh	Sprístupnený zoznam behov priradeného projektu	OK
Zobrazenie „burn-down“ grafu behu	Užívateľovi je priradený aspoň jeden projekt a v rámci neho existuje aspoň jeden beh	Užívateľovi sa zobrazí grafická podoba „burn-down“ grafu	OK
Zobrazenie grafu času behu	Užívateľovi je priradený aspoň jeden projekt a v rámci neho existuje aspoň jeden beh	Užívateľovi sa zobrazí grafická podoba grafu behu	OK
Priradenie užívateľov k projektu	Užívateľovi je priradený aspoň jeden projekt a existuje aspoň jeden ďalší užívateľ	Je možné vybrať ďalšieho užívateľa zo zoznamu	OK

Tabuľka 8.2: Užívateľské testy – projekty a behy (všetci užívatelia)

Ďalšia tabuľka 8.3 zobrazuje testovacie prípady pre všetkých užívateľov, ktorých úlohou je otestovať prípady súvisiace s prácou s úlohami v rámci jednotlivých behov projektu.

Scenár	Podmienky	Očakávaný výsledok	Stav
Vytvorenie úlohy do stĺpca	Existujúci projekt a beh projektu	Úloha úspešne pridaná do stavu (stĺpca)	OK
Modifikácie úlohy	Existujúca úloha	Aktualizácia dát úlohy	OK
Archivácia úlohy	Existencia aspoň jednej úlohy	Úloha nie je v stave (stĺpci) behu ale je prístupná len z archívu projektu	OK
Pridanie úlohy z archívu	Existuje aspoň jedna úloha v archíve	-	OK
Premiestnenie úlohy do iného stavu	-	-	OK

Tabuľka 8.3: Užívateľské testy – beh projektu a úlohy (všetci užívatelia)

Predposledná séria testov v tabuľke 8.4 je určená pre rolu *projektový manažér*. Táto sada testov obsahuje naviac aktivity spojené so správou projektov.

Scenár	Podmienky	Očakávaný výsledok	Stav
Vytvorenie projektu	-	Nový projekt v systéme	OK
Editácia informácií projektu	Existuje aspoň jeden projekt priradený užívateľovi	Informácie o projekte sú aktualizované	OK
Pridanie behu projektu	Existujúci projekt	Nový beh v rámci projektu	OK
Editácia informácií behu projektu	Existujúci projekt s aspoň jedným behom	Informácie o behu projektu sú aktualizované	OK
Vytvorenie užívateľa typu *Projektový manažér*	-	Vytvorený nový užívateľ typu *projektový manažér* s možnosťou prihlásenia do systému	OK

Tabuľka 8.4: Užívateľské testy - projekty a behy (užívateľ typu *projektový manažér*)

Posledná séria testov obsahuje prípady očakávaných chýb, ktorých sa užívateľ môže dopustiť pri používaní systému. Zoznam týchto testov je v tabuľke 8.5.

Scenár	**Podmienky**	**Očakávaný výsledok**	**Stav**
Zadanie nesprávnych prihlasovacích informácií	-	Užívateľ nie je prihlásený do systému a je zobrazená hláška	OK
Vytvorenie užívateľa s už registrovaným e-mailom	V systéme existuje užívateľ	Zobrazená chybová hláška a užívateľ nie je znovu registrovaný	OK
Dátum začiatku projektu je väčší ako dátum ukončenia projektu	-	Zobrazená chybová hláška a neuložený požadovaný stav	OK
Dátum začiatku behu projektu je väčší ako dátum ukončenia behu projektu	-	Zobrazená chybová hláška a neuložený požadovaný stav	OK

Tabuľka 8.5: Užívateľské testy – testovanie chybových stavov

Po vykonaní testov aplikácie je možné analyzovať súčasný stav a zhodnotiť ďalšie možnosti rozšírenia aplikácie.

Kapitola 9

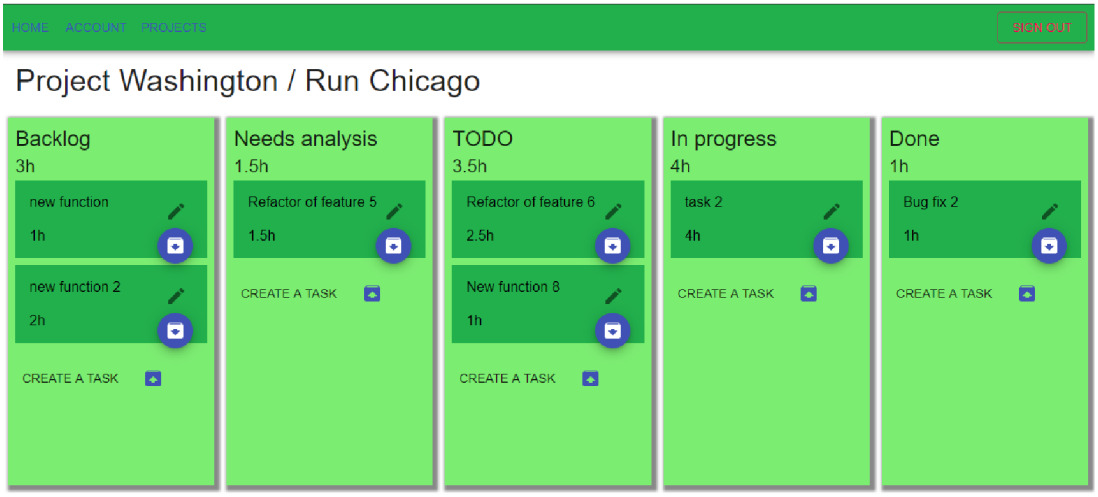
Ďalšie možnosti rozšírenia aplikácie

Táto kapitola sa venuje diskusii aktuálneho stavu aplikácie, ako aj analýzou ďalších možností na rozšírenie aplikácie.

9.1 Súčasný stav aplikácie

Aplikácia samotná, ktorá bola vyvinutá ako tzv. *MVP* na základe skúmaného prostredia je aj podľa reakcii na samotný návrh v použiteľnom stave. Súčasný stav aplikácie vyhovuje základným potrebám agilných tímov, ktoré nemusia byť vyslovene z prostredia *IT*.

Ukážka výslednej aplikácie je na obrázku 9.1, ktorý zobrazuje stránku nástenky behu projektu s jednotlivými stavmi a úlohami.



Obr. 9.1: Ukážka nástenky behu projektu v aplikácii [vlastná tvorba]

Pre prípad použitia aplikácie v praxi sa ponúkajú dve možnosti distribúcie súčasnej aplikácie. Jedna z možností je nasadenie aplikácie do prostredia napr. skúmanej firmy a prispôsobenie konkrétnejším potrebám tej danej firmy. Druhou možnosťou publikovania aplikácie je možnosť publikovania zdrojových súborov pod *open source licenciou*, a v tom prípade by bol prístupný ľubovoľným užívateľom s možnosťami rozšírenia podľa požiadaviek komunity.

9.2 Možnosti rozšírenia

Podoba aplikácie sa v budúcnosti môže vybrať rôznymi smermi. Rovnako aj jej rozšírenia môžu byť rôzneho charakteru. Podobne ako bolo analyzované aj v sekcii zaoberajúcej sa analýzou spätnej väzby k návrhu užívateľského rozhrania 6.5, je možné rozšírenia aplikácie rozdeliť do viacerých kategórií.

Rozšírenie funkcionality

V rámci rozširovania súčasnej funkcionality je možné sa pozrieť na viaceré úrovne v rámci aplikácie.

Všeobecná úroveň

Vo všeobecnosti by sa aplikáciu dalo rozširovať viacerými smermi napr. pridanie viacerých užívateľských rolí s rôznymi právomocami, rozdeľovanie projektov na základe oddelení v rámci organizácie, z čoho vyplýva aj prípadné vytváranie organizačnej hierarchie s vyplývajúcimi právami užívateľov na jednotlivých pozíciách k vykonávaniu rôznych druhov akcii v rámci aplikácie.

Úroveň projektov a behov

V prípade projektov a behov k daným projektom sú možnosti hlavne v oblasti pridávania rôznych atribútov, ktoré by mohli slúžiť k vytváraniu ďalších štatistických dát – zaujímavých pre rôznych užívateľov. Tak isto by sa v tejto súvislosti dalo rozširovať aplikáciu aj v smere podpory ďalších štatistických nástrojov alebo ukazateľov.

Úroveň behu

V rámci samotného behu projektu je možné, tak ako aj v prípade projektov a behov projektu, pridávanie rôznych atribútov hlavne pre úlohy vytvorené v rámci behu alebo pre stavy behu. K príkladom patrí možnosť pridávania ľubovoľných stĺpcov užívateľom k behu projektu alebo rôzne možnosti automatizácie v rámci behu projektu, ako sú napr. notifikácie pre priradených užívateľoch o blížiacom sa termíne atď.

Prepojenia na iné služby

V prípade možností prepojenia s inými službami je možné vychádzať z možností popísaných v sekcii 6.5.3. Vo všeobecnosti sa teda jedná o integráciu so službami ako sú rôzne organizéry, tzv. *DevOps* nástroje, verzovacie nástroje alebo a i.

Ostatné

Medzi ďalšie oblasti rozširovania implementovanej aplikácie patrí napríklad vytvorenie *otvoreného API* pre možnú komunikáciu s aplikáciou, prípadne pre importovanie respektíve exportovanie dát zo systému.

V rámci zabezpečenia nezávislosti dát na iných službách, alternatívne nad možnosťou mať dáta uložené na vlastnom serveri sa ako riešenie ponúka viacero možností, od klasických *SQL databáz* až po príbuznú možnosť tej, ktoré bola implementovaná a to *Realm*[1] databáza.

[1] https://realm.io/products/realm-database/

Aplikáciu by bolo možné rozširovať aj v oblasti týkajúcej sa myšlienky samotného nástroja, ktorej základná myšlienka je podpora riadenia harmonogramu v agilnom prostredí. Tá myšlienka by sa mohla rozšíriť aj na ďalšie oblasti nie len vývoja, ale aj napr. komunikácie so zákazníkmi, zbere dát od zákazníkov atď.

Kapitola 10

Záver

Cieľom tejto diplomovej práce bolo v prvej časti hlavne teoretické oboznámenie sa s časťami projektového manažmentu. Konkrétnejšie sa jednalo o štandardy znalostných oblastí podľa *PMI*. Jednotlivé znalostné oblasti bolo potrebné bližšie spoznať a stručne popísať do teoretickej rešerše.

Konkrétna oblasť, na ktorú je táto práca zameraná, je bližšie rozobraná v časti o riadení harmonogramu projektu a sú popísané prípadné detaily.

Po štúdiu znalostných oblastí bolo potrebné preštudovať agilné metodiky v rámci projektov z oblasti informačný technológií, kde bolo úlohou nájsť aj súvislosti a spracovať možné prepojenia so znalostnými oblasťami podľa *PMI*, ktoré boli preštudované v rámci predchádzajúcich kapitol.

V rámci teoretickej prípravy bola vykonaná analýza reálnej spoločnosti a hľadanie nedostatkov súvisiacich hlavne s problematikou začínajúcich firiem, čomu bola tiež venovaná časť rešerše a analýzy spoločnosti a všeobecne prostredia začínajúcich firiem. Zo získaných znalostí boli následne určené predpoklady a požiadavky pre vývoj prototypu nástroja pre podporu riadenia harmonogramu.

V rámci návrhu prototypu boli prvotne preskúmané a popísané niektoré existujúce riešenia z oblasti agilného riadenia projektov, najmä z oblasti informačných technológií. Zo získaných znalostí, ale aj z predchádzajúcej analýzy či už reálnej spoločnosti alebo všeobecne *startupového prostredia* vznikla písomná podoba základnej predstavy o navrhovanom systéme. Špecifikácia systému bola následne rozšírená o prípady použitia, kde boli definované aj jednotlivé role, ktoré v rámci systému existujú. Na základe tohto popisu a základných prípadov použitia bolo potom možné identifikovať entity, ktoré by v rámci systému mali existovať a identifikovať aj väzby medzi takýmito entitami.

Dôležitým krokom bol návrh architektúry celého systému a rozdelenie na časti, kde bolo určené, že vhodným riešením bude webová aplikácia napojená na služby *Google Firebase*, ktoré poskytujú jednak možnosti autentizácie užívateľov, ale aj spôsob ako ukladať dáta. Po určení základnej predstavy o podobe systému nasledoval návrh grafického rozhrania pomocou tzv. *drôtených modelov*, ktoré boli potom rozposlané viacerým druhom potenciálnych užívateľov na hodnotenie. Prípadné pripomienky a návrhy pre ďalšiu funkcionalitu boli od respondentov zozbierané pomocou dotazníka.

Pred samotnou implementáciou bolo potrebné najprv štúdium technológií, ktoré mali byť pre implementáciu použité a to hlavne *React*, ktorý bol použitý ako hlavný aplikačný rámec pre implementáciu. Spolu s ním boli aplikované aj viaceré knižnice, ktoré dopomohli dosiahnuť požadovanú funkcionalitu. V rámci serverovej časti alebo teda časti použitých služieb *Google Firebase* bolo potrebné primárne zvoliť vhodnú štruktúru databázy, ktorá sa

vymyká konvenčným *SQL* databázam a rovnako bolo vhodné optimalizovať veľkosť dátového prenosu v pomere k veľkosti, ktorú databáza zaberá.

Implementovanú aplikáciu bolo dôležité otestovať, na čo boli zvolené užívateľské testy, ktoré overili funkčnosť jednotlivých prípadov použitia pre rôzne typy užívateľov. V rámci týchto testov bolo vždy nutné definovať podmienky, za ktorých je možné očakávať definovaný výsledok po vykonaní scenára.

Dosiahnuté výsledky boli následne diskutované v rámci predchádzajúcej kapitoly. Záverečným krokom bola analýza možností rozšírenia implementovanej aplikácie, ktoré boli rozčlenené do viacerých úrovní, čo ponúka široké možnosti pre pokračovanie v práci na implementovanom systéme.

Literatúra

[1] *Agile practice guide.* Newtown Square: Project Management Institute, 2017, ISBN 978-1-62825-199-9.

[2] *A guide to the project management body of knowledge.* Newtown Square: Project Management Institute, sixth edition. vydanie, 2017, ISBN 978-1-62825-184-5.

[3] Beynon-Davies, P.: *Database Systems.* Red Globe Press, 2003, ISBN 1403916012.

[4] Brown, D. M.: *Communicating Design: Developing Web Site Documentation for Design and Planning (2nd Edition) (Voices That Matter).* New Riders, 2010, ISBN 0321712463.

[5] Calabrese, J.: Learning with Fist of Five Voting. September 2014, [Online; navštívené 30.12.2018].
URL https://agileforall.com/learning-with-fist-of-five-voting/

[6] Doležal, J. a. k.: *Projektový management – Komplexně, prakticky a podle světových standardů.* Praha: Grada, 2016, ISBN 978-80-247-5620-2.

[7] Fedosejev, A.: *React.js Essentials: A fast-paced guide to designing and building scalable and maintainable web apps with React.js.* Packt Publishing, 2015, ISBN 9781783551620.

[8] Gaur, A.: Schedule Network Analysis Methods. Jún 2018, [Online; navštívené 38.12.2018].
URL https://milestonetask.com/schedule-network-analysis/

[9] Gladwell, B.: Serverless Web Architecture with Firebase. Marec 2016, [Online; navštívené 20.2.2019].
URL https://www.bengladwell.com/serverless-architecture-with-firebase/

[10] James E. Kelley, M. R. W., Jr: Critical-Path Planning and Scheduling. *Procedings of the eastern joint computer conference*, 1959: s. 160–173.

[11] Lopuck, L.: *Web Design for Dummies.* Wiley, 2012, ISBN 1118004906.

[12] McClure, M.: So we redid our charts... (Part II: Graphing, React-ing, and maybe a little crying). Jún 2017, [Online; navštívené 2.4.2019].
URL https://mux.com/blog/so-we-redid-our-charts-part-ii-graphing-react-ing-and-maybe-a-little-crying/

[13] Mew, K.: *Learning Material Design.* Packt Publishing, 2015, ISBN 1785289810.

[14] Mitchell, I.: *Agile Development in Practice*. Tamare House, 2016, ISBN 1908552492.

[15] Moreira, M. E.: *Being Agile: Your Roadmap to Successful Adoption of Agile*. Apress, 2013, ISBN 143025839X.

[16] Poppendieck, M.; Poppendieck, T.: *Lean Software Development: An Agile Toolkit*. Boston, MA: Addison-Wesley, 2003, ISBN 978-0-321-15078-3.

[17] Redbooks, I.: *Using Liberty for Devops, Continuous Delivery, and Deployment*. Vervante, 2015, ISBN 0738441163.

[18] Rippon, C.: *Learn React with TypeScript 3: Beginner's guide to modern React web development with TypeScript 3*. Packt Publishing, 2018, ISBN 1789610257.

[19] Schwaber, K.: *Agile Project Management with Scrum*. Redmond, WA: Microsoft Press, 2004, ISBN 978-0-7356-1993-7.

[20] Singh, H.; Tanna, M.: *Serverless Web Applications with React and Firebase: Develop real-time applications for web and mobile platforms*. Packt Publishing, 2018, ISBN 1788477413.

[21] Smyth, N.: *Firebase Essentials - Android Edition: Second Edition*. CreateSpace Independent Publishing Platform, 2017, ISBN 1979961603.

[22] Stonehem, B.: *Google Android Firebase: Learning the Basics*. CreateSpace, 2016, ISBN 1535004460.

[23] Wenzel, J.: Burn Down Chart Tutorial: Simple Agile Project Tracking". November 2013, [Online; navštívené 29.12.2018].
URL https://web.archive.org/web/20130707200105/http://joel.inpointform.net/software-development/burn-down-charts-tutorial-simple-agile-project-tracking/

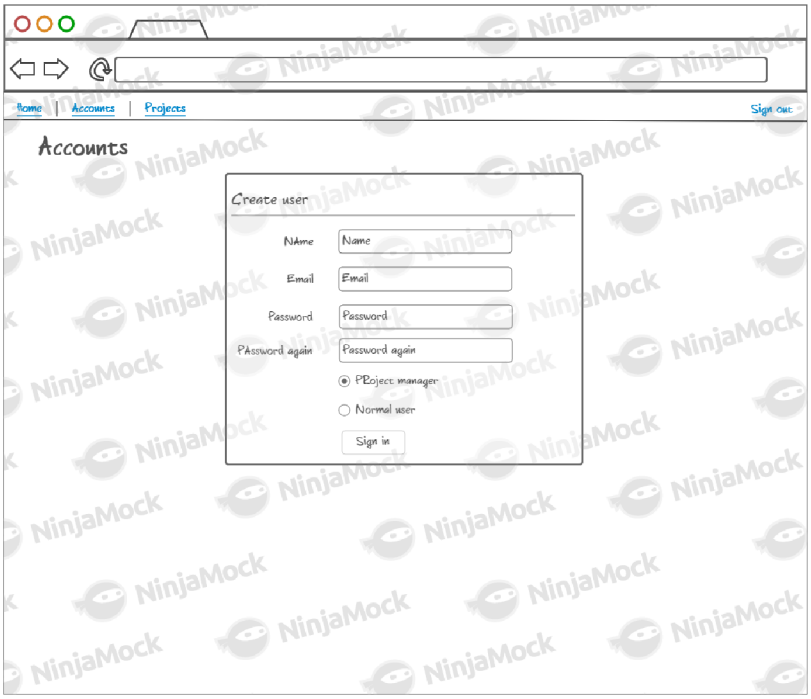
[24] Wise, S.: *Startup Opportunities: Know When to Quit Your Day Job*. Wiley, may 2017, ISBN 9781119378181.

[25] Yahiaoui, H.: *Firebase Cookbook: Over 70 recipes to help you create real-time web and mobile applications with Firebase*. Packt Publishing, 2017, ISBN 1788296338.

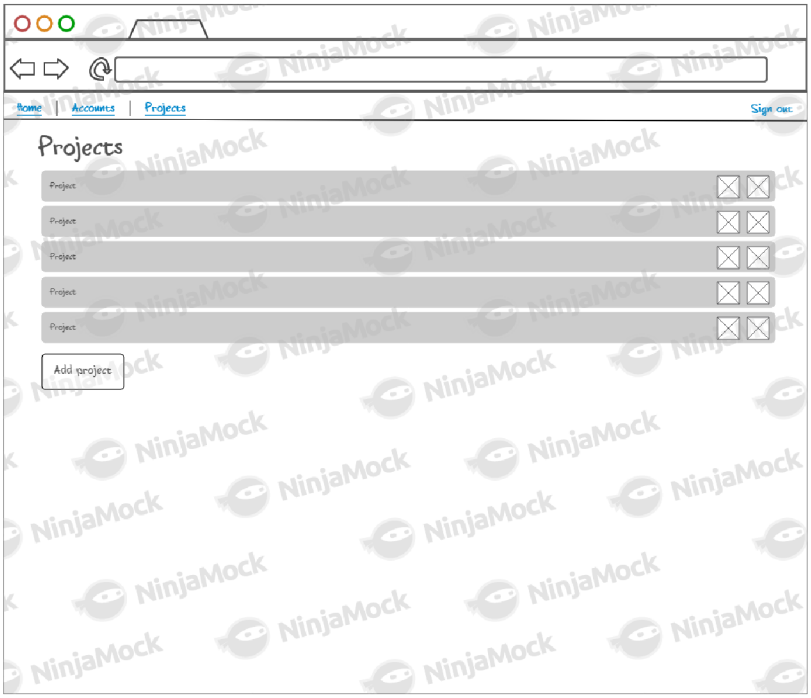
Príloha A

Návrh užívateľského rozhrania

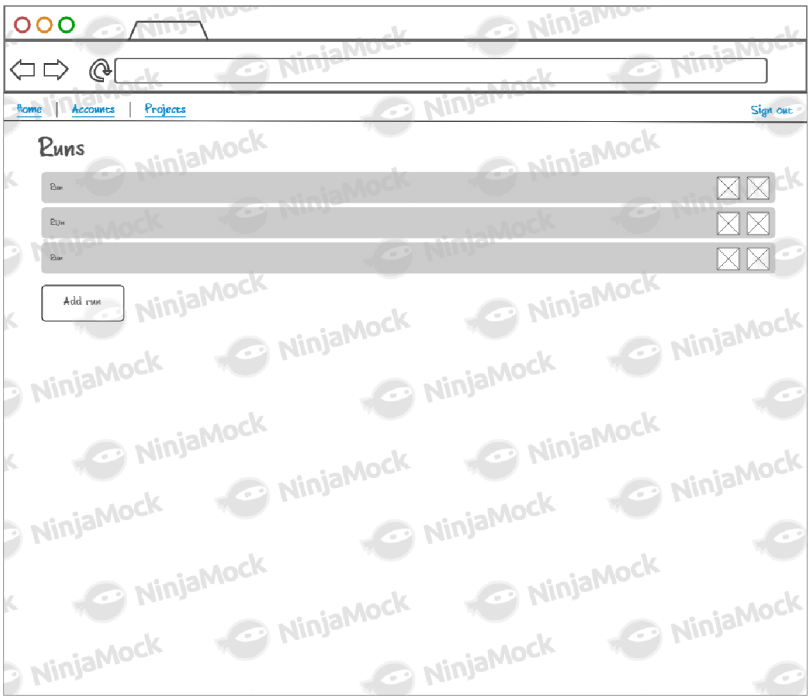
Obr. A.1: Návrh úvodnej - prihlasovacej obrazovky aplikácie [vlastná tvorba]



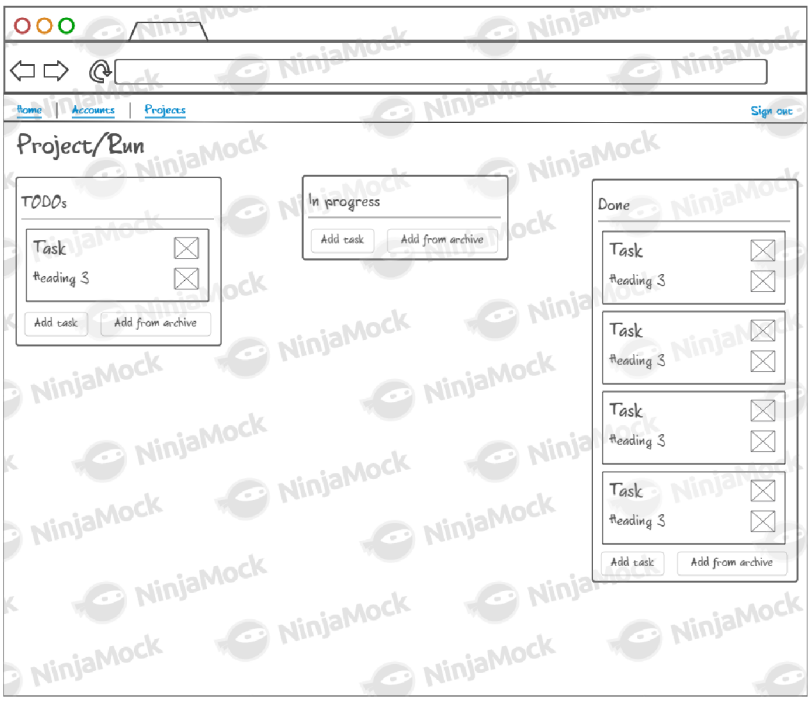
Obr. A.2: Návrh obrazovky správy užívateľov [vlastná tvorba]



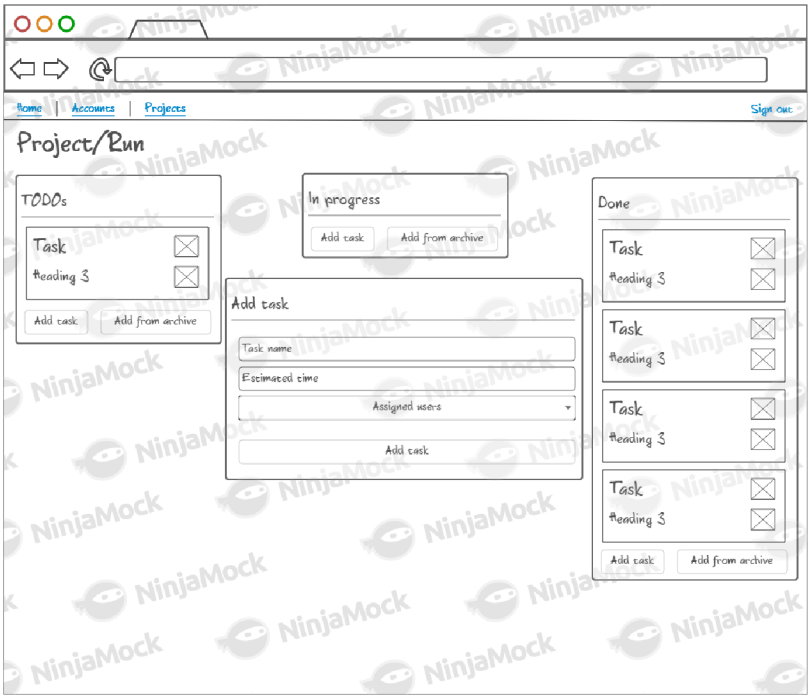
Obr. A.3: Návrh obrazovky zobrazujúcej zoznam všetkých projektov priradených k užívateľovi s možnosťami akcie pomocou tlačidiel [vlastná tvorba]



Obr. A.4: Návrh obrazovky zobrazujúcej zoznam behov projektu s možnosťami akcie pomocou tlačidiel [vlastná tvorba]



Obr. A.5: Návrh obrazovky zobrazujúci tzv. kanban tabuľu s preddefinovanými stĺpcami so základnou podobou úloh pre užívateľov a so základnými akciami pre prácu s úlohami [vlastná tvorba]



Obr. A.6: Podoba dialógového okna pre pridávanie úloh pre daný stav behu projektu a potrebnými vstupnými atribútmi [vlastná tvorba]

Príloha B

Užívateľský dotazník

Ako by si definoval svoju rolu v rámci firmy?

- COO
- CEO (zakladatel startupu)
- Backend Developer
- Tester GUI
- Projektový manažer
- Java dev

Čo by si uvítal v najbližšom vydaní aplikácie? (napr. viac užívateľkých rolí, nejaké informácie o taskoch/runoch/projektoch)

- Určitě více uživatelských rolí/ přidání funkce "rozdělení odvětví", např. tech dev, biz dev,.../ lepší zobrazení timeline tasku a důraz na blížící se časový termín / hlasové nahrávání tasku? (trochu sci-fi, ale dost pomůže při otravném zadávání a vymýšlení ručně, prostě to nadiktuješ a jedním klikem vytvoříš) / přidání funkce důležitosti tasku (pokud něco hoří, zobrazí se notifikace všem i bez tagnutí člena)
- Otvorené API
- V najblizsom vydani by mohlo byt zaujimave pridat rolu "CTO riaditela"spolocnosti, ktory by mal nahlad na statistiky vsetkych projektov a vedel tak efektivne vyhodnocovat situaciu v spolocnosti.
- Automatické úpravy bucketu dle informací z externích služeb, read-only uživatelská role
- pridat vice sloupcu, nazvy dle uzivatele

S akou službou by sa mala aplikácia ideálne integrovať?

- bitbucket
- kalendář(jakýkoli), interní chat (slack,...),

- Toggle
- V praxi sa vo vacsine spolocnosti pouziva ako zakladny nastroj pre komunikaciu a organizaciu OUTLOOK. Idealne by bola integracia s tymto programom, pre notifikacie, terminy v kalendari a podobne.
- Azure DevOps, JIRA
- sdileni tasku

Čo ďalšie by si od systému očakával v budúcnosti (aby bol naozaj použiteľný)?

- aby predpokladal, že môže vzniknúť veľké množstvo taskov. Systém napriek tomu musí zostať prehľadný.
- automatizovanější "pohyb"tasku od vytvoření až po jeho dokončení, tzn. bez větší nutnosti zásahu člověka do systému, který task splňuje. Vše by se odvíjelo od času, který se nastavil na začátku.
- Tmavá téma
- Autor aplikacie podchytil zakladne a dolezite funkcionality, ktore od tohto typu nastroja su ocakavane v praxi. Sucasnu formu navrhu aplikacie by som si vedela predstavit na vyuzitie riadenia a planovania malych az strednych typov projektov.
- Nějaký nástroj pro získávání zpětné vazby k projektu od zákazníka

Príloha C

Obsah CD

Priložené CD obsahuje:

- `dp.pdf` - písomná správa vo formáte PDF
- `tex` - zložka so zdrojovým tvarom písomnej práce LaTeX
- `sources` - zložka so zdrojovými kódmi aplikácie
- `sources\build` - spustiteľné súbory
- `README.txt` - návod pre spojazdnenie aplikácie