Univerzita Hradec Králové
Fakulta informatiky a managementu
Katedra informačních technologií

Návrh a implementace řešení projektového řízení pro malé a střední firmy

Diplomová práce

Autor: Bc. Patrik Štípek
Studijní obor: AI2

Vedoucí práce: Mgr. Daniela Ponce, Ph.D.

Hradec Králové Duben 2022

Prohlášení

Prohlašuji, že jsem diplomovou práci zpracoval samostatně a s použitím uvedené literatury.

V Hradci Králové dne Patrik Štípek

Poděkování

Děkuji Mgr. Daniele Ponce, Ph.D. za odborné vedení diplomové práce, za její čas a za poskytnutí cenných rad, které mi pomohly tuto práci zkompletovat.

Anotace

Tato diplomová práce se zabývá návrhem a implementací aplikace pro projektové řízení v menší firmě zabývající se vývojem softwaru. V práci jsou popsány nejčastější metodiky a přístupy projektového řízení v softwarových firmách. Vybrány existující aplikace, které jsou popsány. Dále jsou rozebrány hlavní požadavky na požadovanou aplikaci a je navrženo řešení. V poslední části jsou popsány vybrané technologie pro implementaci, způsob implementace hlavních modulů a testování výsledné aplikace.

Abstract

This diploma thesis deals with the design and implementation of applications for project management in a small software development company. The work describes the most common methodologies and approaches to project management in software companies. Selected existing applications are described. Next, the main requirements for the application are discussed and a solution is proposed. The last part describes selected technologies for implementation, how were the main modules implemented and testing of the application.

Klíčová slova

Projekt, projektové řízení, metodika, aplikace, Scrum, Kanban

Keywords

Project, project management, methodology, application, Scrum, Kanban

Obsah

Cíl práce a metodika zpracování

Cílem práce je navrhnout a implementovat aplikaci pro projektové řízení ve firmě zabývající se vývojem softwaru. Pro správné pochopení řízení projektů popsat přístupy a vybrané metodiky využívané v projektovém řízení. Kvůli lepšímu přehledu o konkurenci a způsobu fungování jiných nástrojů prozkoumat a popsat existující aplikace. U těchto nástrojů zjistit jejich hlavní vlastnosti a podporované metodiky. Dále na základě požadavků zadavatele, jeho používané metodiky pro řízení projektů, zjištěným informacím z ostatních nástrojů a výsledkům dotazníkového šetření, navrhnout výslednou aplikaci. Následně vybrat technologie pro implementaci, vytvořit požadovanou aplikaci, aplikaci otestovat a poskytnout pro používání ve firmě.

1 Úvod

V této kapitole jsou popsány a vysvětleny základní pojmy, které se objevují v projektovém řízení, vyskytují i v této práci.

1.1 Projekt

Má několik definic, jedna z nich může být tato.

„Projekt je činnost, která má jasně daný cíl, začátek a konec. Zdroje na jeho realizaci jsou omezené, protože se vymyká běžné denní praxi, tak není předem jistý jeho výsledek. Ten může být hmotný i nehmotný: realizace stavby, nová webová stránka, uspořádání školního srazu nebo rekonstrukce koupelny – to vše jsou příklady projektů různého rozsahu.“ [1]

Definice podle ISO 10006.

„Projekt je jedinečný proces sestávající z řady koordinovaných a řízených činností s daty zahájení a ukončení, prováděný pro dosažení cíle, který vyhovuje specifickým požadavkům, včetně omezení daných časem, náklady a zdroji.“ [2]

Typické znaky projektu jsou:

- Cíl – musí mít jasný výsledek, či užitek neboli něco, co se má realizovat, vytvořit, či změnit.
- Čas – sled činnosti v omezeném čase, obvykle v řádu měsíců.
- Jedinečnost – jedná se o neopakovatelný, unikátní sled činností, který vyžaduje specifický způsob řízení. [2]

1.2 Životní cyklus projektu

„Životním cyklem projektu se rozumí průběh projektu, rozdělený do dílčích částí (fází) dle společných charakteristik. Přestože je každý projekt unikátní, tak mají všechny projekty z hlediska řízení společné znaky. Jedná se hlavně o projektové fáze, kterými projekt prochází. Čtyři základní fáze jsou: zahájení, plánování, realizace a ukončení.“ [2]

1. Zahájení – identifikace problému, který má být projektem vyřešen, a formulace představy o tom, čeho má být projekčním řešením dosaženo – specifikace zadání projektu. [3]
2. Plánování – sestavení plánu řešení projektu: definice cílů projektu, vytvoření představy o „ideální cestě" ke zvoleným cílům, stanovení požadavků na zajištění

projektu potřebnými kapacitními zdroji, sestavení projektového týmu – vypracování projektové dokumentace (harmonogram postupu a rozpočet). [3]

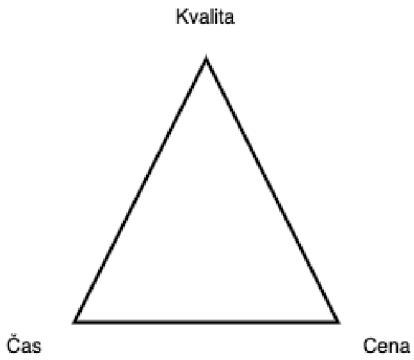
3. Realizace – fáze, ve které jsou implementovány procesy související s projektem, jsou přiřazeny úkoly a přiděleny zdroje. Metoda také zahrnuje vytváření výstupů a uspokojování požadavků zákazníků. Projektoví manažeři nebo vedoucí týmů plní úkol prostřednictvím alokace zdrojů a udržením soustředění členů týmu. [4]
4. Ukončení – zhodnocení dosažených výsledků, záznam získaných zkušeností a jejich využití, poučení pro další projekty, rozpuštění týmu – archivace záznamů. [3]

1.3 Projektové řízení

„Řízení projektu (někdy též projektové řízení) se zabývá řízením projektu, tedy časově ohraničené a ucelené sady činností a procesů, jejímž cílem je zavedení, vytvoření nebo změna něčeho konkrétního. Je to organizované úsilí s jasným časově definovaným cílem. Jeho účelem je zajistit efektivní řízení sady činností tak, aby přinesla předpokládaný výsledek v předpokládaném čase za předpokládané náklady (viz projektový trojimperativ). Při projektovém řízení je tedy třeba aplikovat znalosti, zkušenosti, dovednosti, činnosti, nástroje a techniky na projektu, aby projekt splnil požadavky na něj kladené a dosáhl svých cílů v čase, v nákladech i potřebné kvalitě.“ [5]

1.4 Projektový trojimperativ

Trojimperativ vyjadřuje tři základní parametry, kterými je měřen úspěch projektu, jimiž jsou: čas, rozpočet a kvalita. Řízení projektů přináší obvykle různé komplikace. V praxi dochází k porušení jednoho ze zmíněných parametrů. Nejčastěji dochází ke zpoždění termínů, k překročení nákladů, anebo zhoršení kvality výstupů. Všechny tyto situace jsou pro zákazníka nepříjemné. Udržení trojúhelníku v rovnováze je proto největším uměním projektových manažerů.



Obrázek 1 Trojimperativ

1.5 Metodika

„Metodika je obecně pracovní postup (metoda) nebo nauka o metodě. Ve vývoji softwaru metodika představuje souhrn doporučených praktik a postupů, pokrývajících celý životní cyklus vytvářené aplikace. Pro řešení dílčích problémů mohou být v rámci nasazení metodiky uplatněny specifické postupy – metody.“ [6]

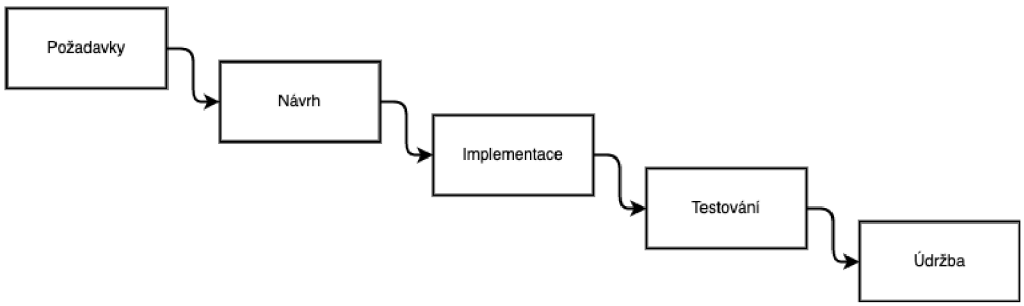
2 Přístupy projektového řízení

„Z důvodu toho, že každý projekt je trochu jiný a má jiné specifikace a vlastnosti, neexistuje jeden jediný „správný“ přístup projektového řízení. Ten je nutné vždy volit podle charakteru a podmínek konkrétního projektu. Podle toho, jaké typy projektů se ve firmě nachází. Jinak se řídí projekty vývoje softwaru a jinak výstavba nové budovy nebo výroba auta. V zásadě existují dva základní přístupy k řízení projektu - tradiční a agilní.“ [5]

Jelikož se tato práce zabývá vytvořením softwarové aplikace, tak v následujících kapitolách budou popsány hlavně metodiky zabývající se vývojem softwaru.

2.1 Vodopádový přístup

Někdy nazývaný jako tradiční přístup. V tradičních přístupech se projektové fáze dokončují jedna po druhé a v sekvenčním pořadí. Stejně jako voda, která teče shora dolů. „Je založen na důkladném naplánování na začátku projektu a řízení všech aktivit v průběhu projektu. Je vhodný pro projekty, které mají předem jasně danou podobu cíle, kde je velmi malá pravděpodobnost, že se projekt bude v průběhu měnit a kde je potřeba dobře naplánovat a odřídit všechny aktivity, návaznosti či subdodavatele. Tradiční přístup vyžaduje kvalitně popsaný cíl, výstupy a plán projektu. Mezi metodiky využívající vodopádový přístup patří například PMBOK nebo IPMA.“ [7] V diagramu níže lze vidět, jaké fáze projekt může obsahovat, a jak mohou na sebe navazovat.



Obrázek 2 Tradiční přístup

2.1.1 Výhody

Je to velmi jednoduchý přístup k použití a pochopení. Fáze na sebe navazují v daném pořadí. Projekt se do další fáze dostane teprve tehdy, kdy je předchozí fáze kompletně

dokončena. Pracuje se na projektu bez, nebo s minimální zásahem od zákazníka, který většinou vidí až konečný výsledek. Kvůli tomu zákazník nemůže tolik ovlivnit práci na projektu a tím narušit jeho proces.

2.1.2 Nevýhody

Není vhodné přístup použít pro složitější a komplexnější softwarové projekty. Ve vodopádovém přístupu není žádná zpětná vazba mezi jednotlivými fázemi. Z důvodu toho, že jakmile jedna fáze skončí, tak se už do ní nelze vrátit. Počítá se s tím, že v projektu nebudou žádné chyby, jakákoliv chyba lze opravit jen v dané fázi. Případné odstranění chyb v dalších fázích může být velmi složité. Testovací fáze, ve které lze objevit chyby nebo nežádoucí vlastnosti, přichází až v posledních fázích projektu. Další nevýhoda je složité vyhovění změnovým požadavkům zákazníka v průběhu procesu. Je potřeba, aby všechny požadavky a cíle byly správně popsány před začátkem práce na projektu. Jejich absence nebo neúplnost následně může projekt zkomplikovat.

Jednotlivé fáze se nepřekrývají, ale v jednom čase se pracuje vždy jen na jedné fázi. Zejména u vývoje softwaru, kde tým tvoří většinou více osob, je z hlediska úspory času a větší efektivity potřeba, aby všichni pracovali najednou a nemuseli čekat na ostatní členy týmu, než dokončí svou práci.

Zákazník dostane až konečný výsledek a nemá mnoho možností, jak upravit zadání projektu. Bez možnosti upravovat požadavky během procesu nastává riziko, že konečný výsledek bude nevyhovující pro zákazníka. Z tohoto důvodu budou nutné opravy.

2.1.3 Použití

Vodopádový přístup lze použít pro menší softwarové projekty a projekty, u nichž je zadání přesně definováno před začátkem práce na projektu a je malá pravděpodobnost, že by se zadání měnilo. Mimo vývoj softwaru lze tento přístup použít například u stavby domu nebo výroby automobilu.

2.2 Agilní přístup

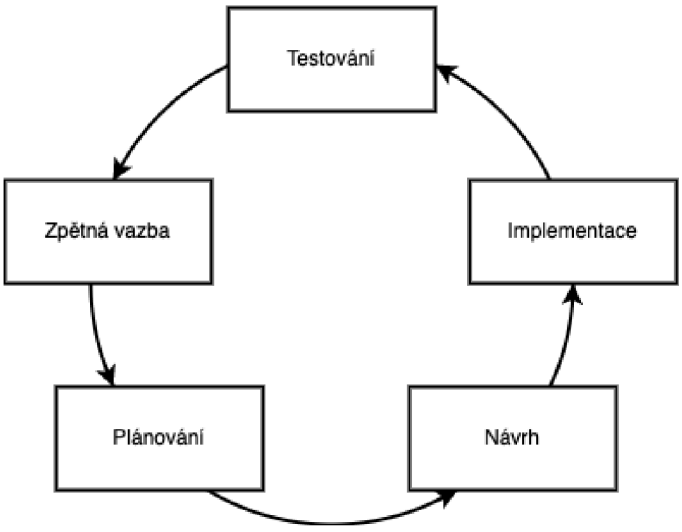
„Je založený na průběžném upřesňování cíle projektu díky interakci se zákazníkem či s koncovými uživateli, na pružných reakcích na změny a průběžném rozvrhování práce v průběhu projektu. Agilní přístup je vhodný pro takové projekty, kde dochází k vývoji

produktu. Tedy za podmínek, kdy nelze předem kvalitně popsat a naplánovat všechno do detailu a bez interakce s budoucím zákazníkem či uživatelem. Agilní přístup se často využívá ve vývoji softwaru, kde je v protikladu vůči tradičnímu přístupu, tzv. vodopádovému modelu.“ [7]

„Eliminuje riziko, že se stráví měsíce, nebo roky prací na projektu, který nakonec selže kvůli malé chybě v rané fázi. Místo toho se spoléhá na důvěru zaměstnanců a týmů, že budou pracovat přímo se zákazníky, aby pochopili cíle a poskytli řešení rychlým a postupným způsobem.“ [8]

Agilní přístup lze přirovnat k přírůstkovému modelu. V případě softwaru je produkt dodáván a vyvíjen v opakovaných přírůstkových cyklech, které se v některých metodikách nazývají „sprinty“. Během těchto cyklů, které trvají obvykle několik týdnů se u vybraných požadavků navrhne řešení, které se následně zrealizuje. Výsledkem jsou malé přírůstkové dílky produktu, přičemž každý vychází z předchozího. Každá verze je důkladně testována, aby byla zajištěna kvalita výsledného produktu.

Rozdíl oproti tradičnímu přístupu lze vidět v následujícím diagramu, kde je zobrazeno, jak se jednotlivé fáze opakují. Agilní přístup má mnoho metodik a každá má svá vlastní specifika, některé jsou popsány v dalších kapitolách.



Obrázek 3 Agilní přístup

2.2.1 Výhody

Požadavky zákazníka jsou vyřízeny během krátké doby. Zákazník pravidelně komunikuje s dodavatelem, díky tomu může pravidelně upravovat své požadavky podle aktuálního

stavu projektu. Dané změny lze snadno zařadit do požadavků, i pokud jsou dodány v pozdější fázi. Všichni členové týmu mohou pracovat zároveň na několika fázích bez toho, aniž by byly předchozí dokončeny. V rámci vývoje softwaru jsou výsledky dodávány během několika týdnů, nikoliv měsíců.

2.2.2 Nevýhody

„U menších projektů může agilní přístup proces spíše zkomplikovat než zjednodušit. Je potřeba expertního člena týmu, který umí správně rozhodnout jakým způsobem se bude daný požadavek řešit. U vývoje softwaru se tým méně zaměřuje na dokumentaci své práce, což může ztížit následnou údržbu systému.“ [9] U komplexnějších projektů je náročnější odhadnout celkovou náročnost projektu.

2.2.3 Použití

Agilní přístup lze použít hlavně u větších a komplexnějších softwarových projektů. Pokud je velká pravděpodobnost změny jednotlivých požadavků nebo částí výsledného řešení. Také pokud zadání není na začátku kompletní a bude dodáno později.

2.3 Metodiky

Představují způsob řízení projektu, mohou být přejaté, anebo vlastní upravené na míru osobitým potřebám. Následující metodiky byly vybrány z několika dostupných webových zdrojů, které vybírají v současné době nejpoužívanější metodiky. Výběr je brán hlavně k rokům 2021 a 2022. Jde především o agilní metodiky zaměřené na softwarový vývoj. [10] [11] [12]

2.3.1 Scrum

Scrum se dá použít v mnoha odvětvích jako je prodej, marketing atd., v rámci vývoje softwaru je metodika určena pro vývoj, dodání a údržbu projektů. Je určen pro menší týmy, které mají obvykle do deseti členů. „Scrum je charakterizován krátkými fázemi neboli „sprinty“, kdy dochází k práci na projektu. Během plánování sprintu se z produktového backlogu vyberou úlohy, na kterých bude tým pracovat, a které by měly být dokončeny během nadcházejícího sprintu, který obvykle trvá dva až čtyři týdny. Na

konci sprintu by zadané úlohy měly být dokončeny a práce připravena pro předání klientovi. Na konci sprintu se provádí revize a retrospektiva sprintu.“ [13]

Produktový backlog je seznam všech úkolů/požadavků, které mají být během projektu dokončeny. Jednotlivé položky jsou ohodnoceny časovou náročností a prioritou. Podle těchto hodnot jsou seřazeny. Může existovat i backlog sprintu, který obsahuje položky pro daný sprint.

V procesu se vyskytují tři role: Scrum Master, produktový vlastník (Product Owner) a Scrum tým. Scrum Master zajišťuje, aby byl dodržován proces Scrum, odstraňuje překážky, které by mohli narušit proces. Zajišťuje komunikaci v rámci týmu a komunikuje s externími subjekty. Produktový vlastník představuje zákazníka a může rozhodovat o produktu. Tato osoba vlastní seznam nevyřízených úloh a je zodpovědná za prioritizaci těchto úloh. Spolupracuje s týmem na denní bázi i poskytuje informace o produktu. Scrum tým se skládá z několika osob, které společně pracují na zadaných úkolech a jsou zodpovědní za dodání produktu. [14]

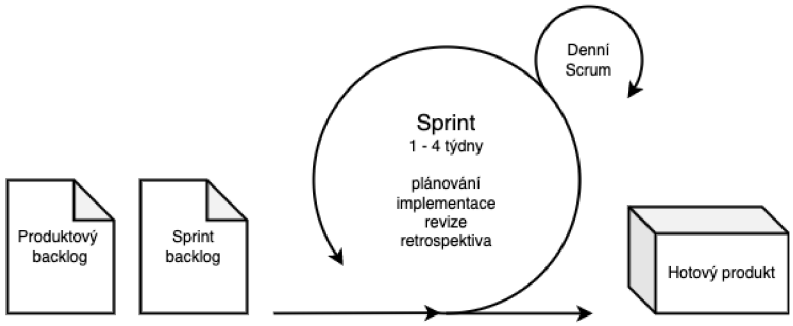
Práce na sprintu se plánuje na plánovacím meetingu, který se koná vždy na začátku každého sprintu. Na této schůzce se stanoví práce a proces na celý další sprint, měli by zde být všichni členové, kteří jsou zapojeni do projektu. Produktový vlastník vybere úkoly, které by chtěl zařadit do dalšího sprintu a Scrum tým vybere ty, které mohou být dokončeny v dalším sprintu. Vybrané úkoly jsou probrány a rozděleny mezi členy týmu. Případné nejasnosti by měl produktový vlastník vysvětlit. Tento meeting by měl být vzhledem k jeho obsahu nejdelší (může trvat i několik hodin). Během každého dne se konají týmové meetingy tzv. denní Scrum. Během těchto meetingů každý člen týmu popisuje, na čem pracoval od posledního meetingu, jaký má plán na daný den a problémy, se kterými se setkal, i se kterými potřebuje pomoct. Tyto meetingy by neměli trvat déle než 15 minut.

Na konci sprintu se koná revize a retrospektiva daného sprintu. Účelem revize je zkontrolovat konečný výsledek a předvést ho ostatním. Ostatní strany zhodnotí, čeho bylo dosaženo a dají týmu zpětnou vazbu k jejich práci. Z té pak mohou vyplynout další úkoly, které se zařadí do produktového backlogu. Vzájemná komunikace umožňuje zlepšovat produkt a dělat další práci efektivnější. Po revizi nastává retrospektiva. Zde se sejdou většinou jen členové Scrum týmu – další strany zde nejsou potřeba. Jejím cílem je zjistit, jak aktuální sprint probíhal, co se během sprintu podařilo, a co naopak ne. Nachází se zde prostor na případné návrhy a vylepšení další práce. Tyto meetingy umožňují zlepšit práci

v týmu. Tento cyklus se opakuje v průběhu celého životního cyklu projektu, dokud není projekt dokončen.

Sprint lze zrušit před jeho dokončením. Pouze produktový vlastník má pravomoc sprint zrušit. Může tak učinit i pod vlivem ostatních stran, vývojového týmu nebo Scrum Mastera. Důvody, proč by mohl být sprint zrušen jsou: cíl se stane zastaralým, nebo již nedává smysl. Kvůli krátkému trvání sprintů tato situace nastává zřídkakdy. [15]

Metodiku Scrum lze použít, pokud veškeré projektové požadavky nejsou k dispozici na začátku projektu nebo lze očekávat změny požadavků během vývoje. Důležité také je, aby byl produktový vlastník neustále k dispozici pro tým v případě, že by se objevily nejasnosti. Není vhodné metodiku použít v rámci velkých týmů, kde tato metodika není tak efektivní jako u menších týmu. Celý proces Scrum metodiky lze vidět v následujícím obrázku.



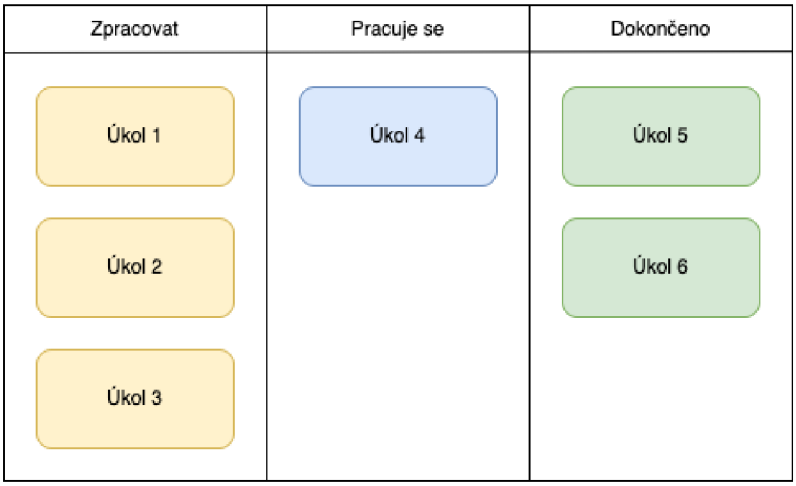
Obrázek 4 Scrum proces

2.3.2 Metoda Kanban

„Název pochází z japonského slova „kanban“, což v překladu znamená „vizuální tabule“. Poprvé tuto metodu začala používat automobilka Toyota ve čtyřicátých letech minulého století jako systém plánování výroby. Na druhé straně je termín „Kanban“ s velkým písmenem. Tento termín je znám a spojován se vznikem „Metody Kanban“, která byla poprvé definována v roce 2007.“ [16]

Kanban funguje na principu nástěnek, na kterých jsou zobrazeny úkoly a jednotlivé fáze procesu. Kanban nástěnka je vizuální způsob správy úkolů a pracovních postupů, která využívá sloupce pro zobrazení pracovního procesu a karty, které znázorňují jednotlivé pracovní položky. Základní sloupce jsou „co udělat“, „pracuje se“ a „dokončeno“. Sloupce mohou být přidány nebo upraveny podle nastaveného procesu.

Karty se přesouvají mezi jednotlivými sloupci a znázorňují tak, v jakém jsou aktuálně stavu. Směr toku je zleva doprava. Úplně poslední sloupec obsahuje dokončené úkoly. Příklad nástěnky je zobrazen na následujícím obrázku.



Obrázek 5 Kanban nástěnka

Hlavním benefitem metody je zvýšená viditelnost procesního flow. Metoda je velmi flexibilní a lze ji použít v mnoha odvětvích. Je to také velmi jednoduchá metoda pro pochopení a použití.

Kanban metoda a Scrum se v mnohém podobají, ale existují zde rozdíly. Pro použití metody Kanban není z počátku potřeba žádných velkých procesních změn, změny se aplikují postupně. Produkt je dodáván postupně, jak je potřeba, kdežto ve Scrumu jsou jasně definované termíny doručení podle sprintu. Úkoly ve Scrumu jsou jasně definované na začátku každého sprintu, kdežto v Kanbanu lze úkoly přidávat kdykoliv. Scrum umožňuje mít více rozdělané práce najednou, Kanban limituje počet rozpracovaných úkolů. Proces v metodice Scrum se mění podle každého sprintu, ale v metodě Kanban zůstává pro celý projekt proces stejný.

2.3.2.1 Principy

Kanban metoda obsahuje několik základních principů, jsou to:

1. Začni s tím, co právě děláš – metoda Kanban důrazně klade důraz na neprovádění žádných změn ve stávajícím procesu ihned. Kanban musí být aplikován přímo na aktuální pracovní postup. Jakékoliv potřebné změny mohou

nastat postupně během určitého časového období tempem, které je týmu příjemné. [17]

2. Souhlas s inkrementálními a evolučními změnami – metoda je navržena tak, aby se setkala s minimálním odporem. Zapojuje změny inkrementálně po malých částech, aby bylo jednodušší si na tyto změny zvyknout.
3. Zpočátku respektovat současné role a zodpovědnosti – Kanban nevynucuje žádné organizační změny, a proto není nutné provádět změny ve stávajících rolích a funkcích, které fungují dobře. Tým společně identifikuje a implementuje všechny potřebné změny. [17]
4. Podpora činů vedení na všech úrovních – podpora zlepšování na všech úrovních organizace. Zlepšení nemusí přijít jen od vedení, každý člen organizace může přijít se zlepšením. [17]

2.3.2.2 Praktiky

Pro úspěšné použití Kanban metody je potřeba dodržet šest základních praktik.

1. Vizualizace práce – hlavní praktika pro použití metody je vizualizace. Lze využít jak fyzickou, tak i elektronickou nástěnku, kde budou vizualizovány kroky procesu, které se v projektu používají. Podle pracovního procesu se pak odráží počet sloupců na nástěnce. Jednotlivé úkoly jsou zařazeny ve sloupcích podle jejich aktuálního stavu. Uživatel tak může sledovat v jakém stavu jsou jednotlivé úkoly.
2. Limitace práce v procesu – „jednou z primárních funkcí Kanbanu je zajistit, aby v každém okamžiku bylo jen takové množství aktivních položek, které je tým schopný najednou zvládnout. Pokud neexistují žádné limity aktivních položek, tak se nepoužívá metoda Kanban. Změna soustředění na jiný úkol v polovině procesu obecně poškodí proces a multitasking je jistou cestou k plýtvání a neefektivitě.“ [16] Nastavením maximálního počtu aktivních jednotek se zajistí, že se na další položce začne pracovat teprve až pro ni bude volná kapacita.
3. Řízení flow – řízení a zlepšování toku pracovních úkolů. Správným nastavením lze zajistit, aby práce probíhala hladce a v procesu se nezadrhávala. Vylepšením flow lze zrychlit proces. [17]
4. Udělat procesní zásady explicitní – je potřeba explicitně definovat pravidla/pokyny procesu. Správné zformulování procesů umožňuje jednodušší

pochopení pracovního procesu. Zásady mohou být v rámci celé nástěnky, jednotlivých sloupců nebo úkolů.

5. Implementace cyklů zpětných vazeb – zajišťují, že organizace adekvátně reagují na potenciální změny a umožňují přenos znalostí mezi zúčastněnými stranami. Příkladem můžou být každodenní týmové meetingy. [16]
6. Zlepšujte se společně – metoda Kanban je evoluční proces zlepšování. Pomáhá týmu přijmout malé změny a postupně se zlepšovat tempem a velikostí, které tým snadno zvládne. Podporuje použití vědecké metody – vytvoření hypotézy, otestování hypotézy a provedení změn v závislosti na výsledku testu. Klíčovým úkolem je neustále vyhodnocovat proces a neustále se zlepšovat, jak je potřeba, a jak je to možné. [17]

2.3.3 Extrémní programování (XP)

„Jedná se o metodiku určenou hlavně pro vývoj softwaru. Tato metodika nezavádí nové přístupy a postupy, ale jak název napovídá, tak extremizuje ty dosavadní. Jako příklad lze uvést programování v párech. Zjistilo se, že větší kontrolou se eliminuje více chyb? Tak se bude více programovat v párech nebo se bude klást větší důraz na revizi kódu, dále se v extrémní míře využívá testování – hlavně jednotkové testy.“ [18]

„Extrémní programování se dá použít hlavně, pokud se často mění požadavky na výsledný produkt. Nejlépe se dá XP využít v menších týmech do 12 osob, s větším počtem osob se tým stává méně efektivnější. Dále se počítá s tím, že všichni členové týmu se nachází na jednom místě, pokud někteří pracují vzdáleně, tak tato metoda nemusí být vhodná.“ [19]

2.3.3.1 Hodnoty

Základní hodnoty extrémního programování jsou: komunikace, jednoduchost, zpětná vazba, respekt a odvaha. [18]

1. Komunikace – umožňuje členům týmu sdílet své zkušenosti a postřehy. Pokud člen týmu narazí na nějaký problém, je důležité, aby se o to podělil s ostatními. Je totiž velice pravděpodobné, že tento problém již někdo v minulosti řešil a je možné problém rychleji vyřešit a ušetřit čas.

2. Jednoduchost – všechen kód musí být co nejjednodušší a nesmí obsahovat věci, které aktuálně nejsou potřeba, ale mohly by být potřeba v budoucnu. (Přístup YAGNI – you aren`t gonna need it) Vytváří se pouze to, co zákazník zrovna chce. Jelikož se požadavky mohou měnit ze dne na den, a proto se nevytváří nic navíc, co by se mohlo další den odstranit.
3. Zpětná vazba – přichází z několika zdrojů. Jedním jsou jednotkové testy, které kontrolují, zda systém funguje správně, a že poslední změny, které byly přidány, nijak nerozbily systém. Dále tým dostává zpětnou vazbu od zákazníka, který software pravidelně testuje, a díky získaným výsledkům může zasahovat do vývoje.
4. Odvaha – je důležité, aby se žádný člen týmu nebál říkat pravdu a nebál se změny.
5. Respekt – vůči ostatním, sobě a práci ostatních. Nikdo by neměl dělat změny v systému, které by mohly poškodit nebo ohrozit již hotovou práci. Členové týmu se snaží najít a použít kvalitní a nejlepší řešení.

2.3.4 Lean

„Koncept byl zaveden v automobilce Toyota, s cílem zlepšit zisk snížením nákladů namísto pouhého spoléhání se na zvýšený prodej. Pokud společnost dokáže eliminovat plýtvání, a stane se efektivnější, může ušetřit peníze, a tím zvýšit celkové zisky. Brzy se tato metoda stala velmi populární a upravila se také pro použití v IT.“ [20]

Cílem je odstranit ze softwarového vývoje všechny nepotřebné věci.

„Lean využívá další nástroje, jako je řízení zásob, které snižují počet probíhajících operací při vývoji. Kanban a mapování hodnotového toku – metoda vizualizace vývojového cyklu jako celku, od požadavku klienta až po fázi nasazení, která demonstruje a pomáhá optimalizovat čas strávený čekáním a vlastním zpracováním.“ [20]

2.3.4.1 Principy

Celkově existuje sedm základních principů, z nichž každý má za cíl urychlit dodání a přinést vyšší hodnotu pro koncového uživatele.

1. Eliminace odpadu – za odpad je považováno vše, co nepřispívá k produktu nebo co zákazník nepotřebuje. Například funkce navíc, zbytečné použití drahých nástrojů, duplikace dat nebo chyby v kódu.

2. Rozvoj kvality – díky eliminací odpadu se udržuje kvalita kódu. Dále jsou často použity metody programování řízené testy, nebo programování v párech. [21]
3. Zesílení učení – učení se zlepšuje díky rozsáhlé kontrole kódu, schůzkám s ostatními týmy a párovým programováním. Tímto je zajištěno, že znalosti nebudou mít jen jednotlivci, ale že se dostanou mezi všechny. [21]
4. Pozdější rozhodování – v tradičním projektovém řízení se často stávalo, že vytvořená aplikace se v den vydání ukáže, jako zcela nevhodná pro trh a je potřeba udělat mnoho změn. Tato metoda odkládá nevratná rozhodnutí, dokud nebude shromážděno, co nejvíce informací a veškeré požadavky nebudou prozkoumány. [20]
5. Rychlé doručení – doručením produktu zákazníkovi, co nejdříve po malých částech, umožňuje vývojovému týmu dostat zpětnou vazbu od zákazníka/konečného uživatele. Tým pak může lépe pochopit zákazníkovy požadavky nebo je zákazník může upravit. Předchází se tak tvorbě nechtěné nebo špatné funkcionalitě, která může vzniknout v případě doručení produktu najednou.
6. Respekt k ostatním – cílem je dát volnost členům týmu, než je kontrolovat manažery. Vývojáři mají tak volnost v rozhodování důležitých rozhodnutí, pokud na to mají dostatek zkušeností. Tímto je dosaženo rychlejší aplikací změn a vývojáři jsou více motivovaní. [20]
7. Optimalizace celku – rozdělení úkolu na menší části může vylepšit část systému, ale je potřeba vidět změnu v kontextu celého produktu a snažit se změnou optimalizovat celek.

2.3.5 Crystal

„Metoda Crystal se zaměřuje především na lidi a jejich interakce při práci na projektu, než na procesy a nástroje. Alistair Cockburn (tvůrce metodiky) věřil, že dovednosti a talent lidí, stejně jako způsob, jakým komunikují, mají největší vliv na výsledek projektu.“ [22]

Metoda má dva základní předpoklady:

1. „Tým si najde svou optimální cestu a metodu pro zefektivnění pracovního postupu.“ [22]

2. „Každý projekt se liší od ostatních a vyžaduje určité specifické metody a strategie.“ [22]

„Cockburn zjistil, že vlastnosti jednotlivých projektů se mění podle počtu zapojených lidí a důležitosti projektu. Menší tým může pracovat na projektu bez velkého počtu reportování stavu a tzv. papírování. Zatímco čím větší se tým stává, nebo je složitější projekt, tím roste potřeba použití tzv. papírování.“ [22]

Tato metoda se rozděluje na několik skupin, podle velikosti týmu, které jsou barevně označeny. Bílá – tým maximálně 6 osob, žlutá – maximálně 20 osob, oranžová – maximálně 40 osob, červená – maximálně 80 osob, vínová – maximálně 200 osob v týmu.

2.3.5.1 Vlastnosti

Sedm základních vlastností Crystalu jsou: [23]

1. Časté doručování – doručený produkt je otestován skupinou uživatelů, kteří vrací svojí zpětnou vazbu.
2. Reflexní vylepšení – neustále lze nějakou část produktu vylepšovat.
3. Osmotická komunikace
4. Osobní bezpečí – každý člen týmu by se měl cítit dobře a neměl by se bát sdělit svůj nápad.
5. Soustředění – každý člen týmu ví, co přesně má dělat, a to mu umožňuje se plně soustředit na daný úkol.
6. Snadný přístup k uživatelům – získává se zpětná vazba od uživatelů.
7. Technické nástroje – určuje se seznam nástrojů, které mají být používány členy týmu.

2.3.6 Vývoj řízený vlastnostmi (FDD)

„Metodika orientovaná na zákazníka má za hlavní účel často a opakovaně dodávat řešení, které zákazník požaduje. Obvykle se jedná o jedno nebo dvou týdenní vydání řešení. Práce se rozděluje do menších částí, které se dodávají zákazníkovi. Tento proces se opakuje, dokud projekt není hotový. Oproti ostatním metodikám tato klade větší důraz na tvorbu a správu dokumentace projektu, což umožňuje snadnější práci na dlouhodobých projektech a zapojení nových lidí do týmu.“ [24]

2.3.6.1 Aktivity

V FDD existuje pět základních aktivit během vývoje softwaru. První dvě aktivity proběhnou na začátku vývoje systému, kdežto zbylé tři se opakují pro každý cyklus vývoje.

1. Vytvoření celkového modelu – tvorba modelu nového systému vývojovým týmem a zákazníkem. Po sestavení modelu následuje další fáze.
2. Sestavení seznamu vlastností – sepsání všech vlastností a funkcí, které má systém splňovat a nabízet.
3. Plánování podle vlastností – v této fázi se určí, v jakém pořadí se budou jednotlivé vlastnosti vytvářet. Pořadí je většinou určeno podle několika kritérií. Například priorita zákazníka, časová náročnost, dostupní členové týmu nebo komplikovanost řešení.
4. Návrh vlastnosti – navrhuje se, jak bude řešení implementováno. Díky návrhům vlastnosti se zvyšuje informovanost o části systému.
5. Tvorba vlastnosti – tým implementuje vlastnost podle návrhu z předchozího kroku a následně otestuje funkčnost řešení. [24]

2.3.7 Prince2

„Tato metodika pochází z Velké Británie. Zde a v celé Evropě je velice rozšířená. Zkratka znamená „Pojects IN Controlled Environments“, neboli „projekty v řízeném prostředí“.“ [25]

Tato metoda se dá zařadit jak do tradičního přístupu, tak i do agilního. Dále vznikla Prince2 agile metodika, která se už řadí do agilních metodik.

„PRINCE2 je procesně založený přístup, který se zaměřuje na organizaci a kontrolu nad celým projektem od začátku do konce. To znamená, že projekty jsou před zahájením důkladně naplánovány, každá fáze procesu je jasně strukturovaná a všechny nespojené konce jsou po dokončení projektu spojeny.“ [26]

2.3.7.1 Role a odpovědnost

Pro vytvoření správného prostředí pro řízení projektu jsou definovány hlavní role a zodpovědnosti projektového týmu.

1. Projektový manažer – zodpovídá za plánování a průběhu projektu a jeho fází. Sestavuje projektový tým a kontroluje jeho efektivitu. Komunikuje se zákazníkem a projektovou radou. V rozsáhlejších projektech se může vyskytovat i týmový manažer, který komunikuje mezi týmem a projektovým manažerem. [27]
2. Zákazník – objednává produkt a projekt financuje.
3. Projektová rada – členové projektové rady mají na starosti rozhodovací proces a zastupují zákazníka. Komunikují s projektovým manažerem a schvalují jednotlivé fáze projektu.

2.3.7.2 Principy

Principy zajišťují správné řízení projektu dle Prince2 metodiky.

1. Projekty musí mít obchodní zdůvodnění – každý projekt musí mít jasně definovaný cíl, přínosy nebo zisky a detailní posouzení nákladů.
2. Průběžné učení – v každé fázi projektu by se tým měl snažit učit se z odvedené práce a poznatky se snažit použít v dalších fázích projektu.
3. Jasné rozdělení rolí a odpovědností – v každé fázi projektu by měl projektový manažer jasně definovat role a zodpovědnosti jednotlivcům a týmu. Tím se odstraní nejasnosti a každý má jasno v tom, za co je zodpovědný, a co se od něj očekává. To také poskytuje jasnou metriku výkonu pro všechny členy týmu. [28]
4. Práce je naplánována do fází – každý projekt je rozdělen do několika fází. Na konci každé fáze se kontrolují výsledky dané fáze. Tyto kontroly umožňují zjistit, zda je projekt na správné cestě k dokončení.
5. Řízení dle výjimek – definuje tolerance a povolené odchylky od plánovaného cíle, které se vyskytnou během realizace projektu. Snižuje riziko nekvalitně realizovaného projektu.
6. Zaměření na kvalitu
7. Přístup na míru – přístup ke každému projektu lze upravit podle jeho specifikací.

2.3.7.3 Témata

Poskytují přehled o tom, jak má být projekt řízen. Mohou být považovány za znalostní oblasti, nebo jak se principy uplatňují v praxi. Jsou nastaveny na začátku projektu a následně jsou monitorovány.

1. Obchodní případ – zjišťuje se, zda je projekt proveditelný a zda se vyplatí.
2. Organizace – téma vede projektové manažery k tomu, aby jasně delegovali a sledovali role a odpovědnosti týmů a jednotlivců. [28]
3. Kvalita – požadavky na produkt, které mají být splněny.
4. Plány – popisují, jak bude jednotlivých cílů dosaženo.
5. Rizika – jsou sepsána potenciální rizika nebo události, které mohou nastat a mít dopad na dokončení projektu.
6. Změna – změna může nastat v jakékoli fázi projektu. Toto téma pomáhá přizpůsobit se této změně. Každá změna by měla být odsouhlasena před jejím začleněním.
7. Postup – kontrola aktuálního stavu projektu umožňuje zjistit, jaký je aktuální postup na projektu a díky porovnáním s plánem zjistit, zda jde postup podle plánu.

2.3.7.4 Projektové fáze

Metodika rozděluje proces na sedm fází. Na tyto fáze dohlíží projektový manažer a schvaluje je projektová rada.

1. Zahájení projektu – v této fázi se vyhodnocuje, zda je projekt realizovatelný. Výstupem je projektový plán a struktura.
2. Řízení projektu – projektová rada přezkoumá a vyhodnotí zadání projektu na základě obchodního zdůvodnění a životaschopnosti pro další kolo schvalování. Rada deleguje projekt na projektového manažera. [26]
3. Nastavení projektu – definují se aspekty projektu, rozsah, náklady, časová náročnost, riziko a benefity. Projektový manažer vytvoří projektovou dokumentaci a dá ji ke schválení.
4. Kontrola etap – projektový manažer rozdělí projekt na sadu úkolů a předá je týmovým manažerům a týmům k vypracování. Projektový manažer dohlíží na průběh práce na úkolech během každé fáze a v případě potřeby pomáhá opravit případné chyby. Týmoví manažeři koordinují každodenní práci a fungují jako spojovací článek mezi projektovým manažerem a jednotlivými členy týmu, čímž pomáhají zajistit, aby vše šlo podle plánu. [26]

5. Řízení doručení – projektový manažer kontroluje, že není rozdíl mezi očekávaným výsledkem a dokončenou prací. Projektová rada provede závěrečné zhodnocení a produkt schválí, nebo vrátí na úpravu.
6. Řízení přechodu mezi etapami – na konci každé fáze manažer a představenstvo zhodnotí výsledky této fáze a ujistí se, že projekt postupuje podle plánu.
7. Ukončení projektu – projekt je předán a vyhodnocen.

2.3.7.5 Certifikace

Tato metodika umožňuje mít certifikaci pro řízení projektů. Kurzy trvají několik dní. Ceny pro kurzy se pohybují od 18 000 Kč v závislosti na typu certifikátu. Existují čtyři typy kurzů.

- Prince2 Foundation
- Prince2 Practitioner
- Prince2 Agile Foundation
- Prince2 Agile Practitioner

3 Existující nástroje pro projektové řízení

V dnešní době jsou možnosti pro tvorbu softwaru velmi rozsáhlé, a proto lze velmi dobře vytvořit aplikaci, která umí pracovat podle jednotlivých metodik zmíněných výše. Dnes již existuje mnoho aplikací, které podporují řízení projektů a další neustále vznikají. Z důvodu, že tyto aplikace umí usnadnit a zefektivnit práci, si lidé navykli používat tyto aplikace. Někteří si bez nich již práci ani nedovedou představit. Pandemie tomu o to víc pomohla a donutila pracovat více lidí z domu. Z tohoto důvodu jsou tyto aplikace ještě více zapotřebí. Podle mnohých průzkumů lze zjistit, že tyto nástroje dokáží člověku ušetřit mnoho času denně a zaměstnanci jsou více efektivní. Díky těmto aplikacím můžou týmy komunikovat neustále mezi sebou a spolupracovat v průběhu celého životního cyklu projektu. Tyto nástroje mají velkou oblibu mezi manažery. Umožňují jim plánovat a sledovat postup jednotlivých úkolů. Některé aplikace umožňují sledování odpracovaného času, nastavování rozpočtu a tak dále. Jelikož možnosti těchto nástrojů jsou velice rozmanité, lze je používat v mnoha odvětvích.

Používaných nástrojů je plná řada. Již se moc nevyskytují jako desktopové aplikace, které by uživatel musel nainstalovat do svého počítače, ale stále více existují jako webové aplikace, které uživatel používá přes webový prohlížeč a může je používat na jakémkoliv zařízení bez toho, aniž by je musel instalovat. Mnoho aplikací lze nainstalovat i na mobilní zařízení. Uživatel tak má prakticky neustálý přístup k těmto aplikacím.

3.1 Proč aplikace?

„Aplikace pro správu projektů umožňují sledovat a spravovat téměř jakýkoli druh projektu, jako je vytvoření nového produktu, stavba domu, vývoj webu nebo spuštění marketingové kampaně. Týmy, které používají aplikace pro správu projektů, obvykle mají více než jeden projekt najednou. Aplikace jim pomáhá naplánovat práci na základě termínu dokončení a podle dostupných lidských zdrojů, které jsou k dispozici.“ [29]

„Nejlepší aplikace pro správu projektů odhalí problémy dříve, než k nim dojde. Sledováním postupu práce a jednotlivých úkolů, mohou aplikace pro řízení projektů informovat uživatele, když hrozí, že termín dokončení úkolu se blíží.“ [29]

3.2 Vybrané nástroje

Pro řízení projektů existuje mnoho aplikací a nástrojů. V kapitolách níže jsou sepsány aplikace, které byly v dotazníku nejčastěji zastoupeny a zadavatel s těmito nástroji měl možnost pracovat, a proto by měly sloužit jako předloha pro navrhovanou aplikaci.

3.2.1 Trello

Je aplikace od společnosti Atlassian, která vznikla v roce 2011. Vytvořena byla společností Fog Creek Software. Nabízí webovou, mobilní a desktopovou aplikaci.

Aplikace Trello využívá pro řízení projektů systém kanban nástěnek. Každý projekt má své nástěnky, ve kterých jsou zobrazeny karty (úkoly). Nástěnku lze upravovat stejně jako v metodě Kanban dle potřeb daného projektu. Kartám lze přiřadit různé štítky, podle kterých je lze filtrovat. Je zde možnost přidat kartě seznam checklistů. Nabízí i neomezené úložiště pro ukládání souborů. Nástěnky je možné vytvářet i z mnoha dostupných šablon, které předpřipraví jednotlivé sloupce.

Mezi hlavní vlastnosti Trella určitě patří možnost použití automatizace. Ta umožňuje definovat akce, které se vykonají po spuštění nějaké akce nebo výskytu nějaké události. Automatizaci lze využít bez znalosti programování, takže ji může nastavit úplně každý. Lze ji nastavit jak pro sloupce nástěnky nebo karty. Hodí se pro automatizaci kroků, které se často provádí.

Trello využívá části metody Kanban, ze které převzal systém pracovních nástěnek. Aplikace dává týmu volnou ruku v řízení projektu a je to spíše nástroj, který umožňuje delegovat a vizualizovat stav jednotlivých úkolů.

Trello nabízí integrace s mnoha aplikacemi jako je Slack, Toggle a další. A také s dalšími aplikacemi od společnosti Atlassian.

Aplikace je velmi jednoduchá pro použití, nový uživatel by neměl mít problém se s tímto nástrojem naučit pracovat během několika minut. Nabízí také i mnoho šikovných a složitějších funkcí. Je to vhodný nástroj pro menší i větší týmy. Dá se použít i pro osobní použití například jako osobní todo list. Vzhledem k jeho dobré cenové dostupnosti je vhodný pro týmy, které nepoužívají určitou metodiku a tento nástroj jim stačí. Nicméně větší týmy nebo týmy, které používají například metodiku Scrum nemusí tento nástroj stačit a spíše budou používat jiný nástroj.

3.2.1.1 Ceník

Ceník je rozdělen do čtyř plánů, které lze zakoupit. V plánu Free je aplikace dostupná zdarma. Tento plán je hlavně limitován počtem pracovních nástěnek, které lze v aplikaci mít. Je proto vhodný spíše pro osobní použití. Dále jsou plány standard, prémium a enterprise. Tyto plány již nejsou tak limitovány, ale jsou zpoplatněny. Ceny jsou od 5 $ za uživatele ve standardním plánu, 10 $ za uživatele v plánu prémium a v plánu enterprise je cena podle počtu uživatelů, tento plán je určen pro velké organizace. Firmu s 15 registrovanými uživateli v plánu standard by tato aplikace stála okolo 1 600 Kč měsíčně.

3.2.2 Freelo

Jako jediná z vybraných aplikací je původem z Česka. Je dostupná pro web, desktop a mobilní zařízení. Z důvodu toho, že za ní stojí český tým, tak oproti ostatním nástrojům nabízí plnou českou verzi s různými návody a českou podporou.

Nenabízí nastavení žádné konkrétní metodiky. Použití této aplikace je tedy poměrně volné a lze ji využít, jak v tradičním, tak i v agilním přístupu. Tým, který používá tuto aplikaci, si musí interně nastavit svou metodiku a tuto aplikaci může použít jako nástroj pro organizaci projektů a úkolů.

Aplikace nabízí vytvoření projektů z mnoha dostupných šablon z různých odvětví. Jsou dostupné i integrace s některými aplikacemi jako jsou Gmail, Google kalendář, iDoklad a další.

3.2.2.1 Ceník

Je rozdělen do čtyř plánů. K dispozici je plán free, který je zdarma, ale je vhodný hlavně pro osobní použití. Je limitován počtem aktivních projektů, přizvaných uživatelů a kapacitou uložiště. Dále existuje plán freelance. Stojí 890 Kč měsíčně za aplikaci (nikoliv za uživatele). Je vhodný pro jednotlivce a menší týmy. Umožňuje mít neomezeně uživatelů a projektů, omezena je hlavně kapacita a chybí zde několik funkcionalit z dalších plánů. Další plán je team, který stojí 1 990 Kč. Oproti freelance nabízí další role pro uživatele. Tento plán je vhodný pro menší a středně velké firmy. Poslední plán Business stojí přes tři tisíce Kč. Společnost s 15 uživateli by tato aplikace v plánu team vyšla na 1 990 Kč měsíčně.

3.2.3 Jira Software

Je aplikace především pro řízení vývoje softwarových projektů, ale i pro sledování problémů a chyb. Je to nástroj, který podporuje agilní přístup a veškeré metodiky. Především metodiky Scrum a Kanban. Nástroj si lze upravit dle svých potřeb, aby podporoval spojení několika metodik, nebo metodiku vlastní. Software byl vytvořen společností Atlassian v roce 2002. Tato společnost také stojí za aplikací Trello.

Jira nabízí základní nástroje pro metodiku Scrum, jako jsou nástroje pro plánování a řízení sprintu, denní Scrumy a retrospektivu. Tu je možné provádět díky mnoha reportům a grafům, které je možné využít. Pro Scrum projekty existuje šablona, která pomůže správně nastavit metodiku pro vybraný projekt.

V případě, že tým používá nějakou kombinaci těchto metodik, jako je například spojení Scrum a metody Kanban (SCRUMBAN), nebo naopak má svou na míru upravenou metodiku, lze si nastavení projektu upravit podle svých potřeb.

Pro metodiku Kanban také existuje šablona, která nastaví metodiku pro vybraný projekt. V šabloně lze vybrat z několika základních procesů, podle kterých se nastaví kanban nástěnka a tu lze začít ihned používat. Dále lze sloupcům nastavit maximální limit aktivních položek, na kterých se zrovna pracuje. Je možné přidávat a upravovat jednotlivé sloupce v nástěnce a také seskupovat jednotlivé úkoly podle různých kritérií. Jsou k dispozici také různé reportovací nástroje a grafy, které umožňují například zobrazit odpracovaný čas nad jednotlivými projekty.

Tento software je celkově velmi robustní s velkou škálou integrací a dostupných funkcí. Nabízí mnoho způsobů a nástrojů pro řízení projektů. Tato aplikace se hodí spíše pro větší společnosti a týmy, které dokáží využít potenciál této aplikace. Nebo pro společnosti, které používají i ostatní nástroje od společnosti Atlassian.

3.2.3.1 Ceník

Jedná se o nejdražší software ze zmíněných aplikací. Především díky velkému množství nabízených funkcí a celkové komplexnosti softwaru. Z důvodu poměrně vysoké ceny je tento nástroj vhodný spíše pro větší společnosti a týmy.

Pro použití aplikace Jira je nabízeno několik plánů, které se liší cenou za jednotlivé uživatele a nabízenými funkcemi. Nejlevnější plán free umožňuje použití aplikace zdarma. Tato možnost je vhodná pro malé týmy nebo jednotlivce, jelikož je omezena pro deset uživatelů, dále je omezena kapacita úložiště a nedostupné jsou i některé funkce, které jsou

dostupné v placených plánech. Plány standard a prémium mají cenu závislou podle počtu uživatelských účtů, ale pro menší týmy se jedná o ceny 7,5 $ za účet ve standardním plánu a 14,5 $ v prémium plánu měsíčně a nabízí s větší kapacitou úložiště i více dostupných funkcí. V případě menší firmy, která by měla 15 registrovaných uživatelů by vyšel standardní plán okolo 2 400 Kč měsíčně.

4 Analýza a návrh systému

V kapitolách níže jsou popsány jednotlivé prvky, funkce a vlastnosti, které by výsledná aplikace měla poskytovat. Jedná se o spojení požadavků zadavatele, vyhodnocení vlastností existujících aplikací a vyhodnocení dotazníkového průzkumu. Dále je zde popsán návrh struktury a architektura výsledné aplikace.

4.1 Požadavky na systém

Zadavatel se zabývá vývojem softwaru a využívá agilní přístup se svou vlastní metodikou. Ve které se klade důraz především na rychlost dokončení a kvalitu výsledného produktu. Metodika vychází hlavně z metody Kanban. Nicméně nevyužívá například limit aktivních úkolů. Na začátku každého projektu je projednán postup a plán, jak bude práce na projektu probíhat. Následně je rozdělena práce a jsou definovány jednotlivé úkoly. Práce je rozdělena podle rozsahu produktu do sekcí, na kterých se postupně pracuje. Po dokončení nějaké ucelené části produktu se produkt ukáže zákazníkovi. Zákazník může předat zpětnou vazbu, která se může přidat do plánovaných úkolů. Zákazník má tak možnost testovat produkt již v průběhu vývoje. Po předání první ucelené verze produktu je možnost rozšiřovat a implementovat další funkce. Tyto rozšíření jsou naplánovány podle dostupných kapacit týmu, priority zákazníka a komplikovanosti řešení. Pokud pro některé úkoly není aktuálně kapacita, vloží se úkoly do projektového backlogu. Vybrané funkce, na kterých se bude pracovat, jsou projednány v týmu a rozděleny. Jakmile jsou úkoly vyřešeny, je produkt otestován a předán zákazníkovi.

Ve firmě zadavatele vývojový tým aktuálně obsahuje do pěti vývojářů, kteří jsou zodpovědní za vývoj produktů. Díky blízkému kontaktu s ostatními členy týmu se používaná aplikace využívala hlavně pro rychlý přehled, na čem kdo zrovna pracuje. Nicméně s možným růstem týmu je pravděpodobné, že zadavatel zvolí jinou, nebo upraví používanou metodiku a bude aplikaci používat trochu jinak. Proto bude aplikace navržena tak, aby bylo možné použít některou z agilních metodik. Vybrané metodiky jsou Scrum a Kanban. Kanban, jelikož aktuální metodika vychází z této metody a je tedy možné, že se začne používat naplno. Scrum z důvodu velké oblíbenosti a velké šance, že se začne používat na některých projektech používat.

Zadavatel měl možnost si vyzkoušet všechny ze zmíněných nástrojů v kapitole 3.2. Hlavní důvod opuštění aplikace Jira byla finanční stránka. Z vybraných aplikací je Jira

jednoznačně nejdražší. Aplikace za tu cenu nabízí mnoho funkcí, ale zadavatel všechny nepotřebuje. Další důvod byla náročnější práce s aplikací. Jira nabízí mnoho nastavení a funkcí, kvůli tomu se stává složitější pro ovládání. Aplikaci Freelo zadavatel používá pro práci s některými zákazníky a není s ní velmi spokojen, jelikož mu nepřijde velmi přehledná. Proto zvolil možnost, začít používat vlastní aplikaci, kterou by si mohl navrhnout podle svých potřeb.

Mimo funkčních požadavků by aplikace měla splňovat i řadu nefunkčních požadavků. Při výskytu chyby v aplikaci by aplikace neměla přestat fungovat, ale zobrazit informaci o chybě a dále poskytovat ostatní části, které nejsou chybou dotčeny. Proces nahrávání aktualizací by měl obsahovat testovací prostředí, ve kterém by se změny otestovaly, než by byly nasazeny do produkce. Pro možnost obnovy dat, by data aplikace měly být zálohovány alespoň 3 dny zpětně. Pro jednodušší správu a rozšiřitelnost by kód aplikace měl obsahovat znovupoužitelné komponenty. Kód by měl splňovat OOP standardy. Aplikaci by mělo být možné škálovat na serveru přidáním více paměti nebo CPU.

4.1.1 Zadané vlastnosti

Zadavatel používal aplikaci Jira. Požadoval proto některé funkce, které tato aplikace nabízí. Jako hlavní uvedl kanban nástěnku, kterou disponuje většina moderních nástrojů. Každý projekt by měl mít svou vlastní nástěnku s možností upravovat jednotlivé sloupce.

K projektu by měly jít přidávat úkoly, které lze přiřadit jednotlivým lidem. Úkoly by mělo jít přesouvat mezi jednotlivými sloupci, tak jako v každé kanban nástěnce. Úkolu by měla jít nastavit priorita a termín dokončení.

Aplikace by měla mít jednoduchý systém práv a rolí, které vymezují, jak se uživatel může v aplikaci chovat a kam má přístup.

Aplikace by měla mít modul pro měření času práce. Tato měření by měla jít filtrovat podle projektu, ke kterému patří. Díky tomu by mělo jít zjistit, kolik bylo odpracováno času na jednotlivých projektech.

4.1.2 Dotazník

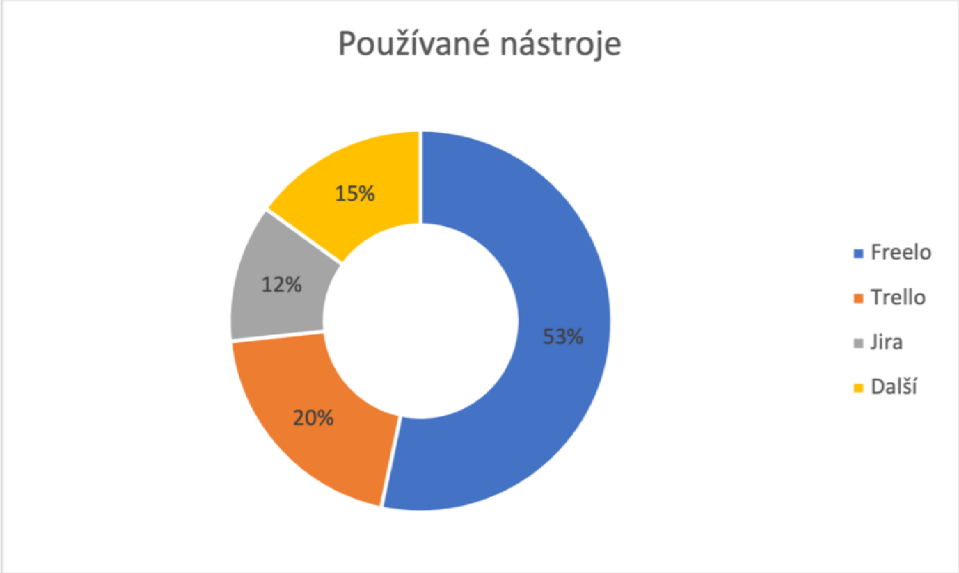
Slouží především jako zpětná vazba od uživatelů, kteří s nějakým nástrojem pracují nebo mají zkušenost. Dotazník obsahoval několik otázek, na některé bylo možné odpovědět

ano/ne a některé bylo možné odpovědět více možnostmi spolu s možností dopsat vlastní odpověď. Dotazník hlavně zjišťoval, jaké funkce uživatelé používají, jak se jim celkově s nástroji pracuje a zda existuje něco, co by změnili nebo vylepšili. Celkem dotazník vyplnilo 60 respondentů.

Díky získaným výsledkům bude možné se v návrhu a implementaci více zaměřit na konkrétní funkcionality a případně přidat další požadavky na aplikaci. U částí, které se uživatelům líbí, bude možné navrhnou funkcionalitu dle daného nástroje. Naopak u funkcionalit, se kterými nebudou všichni spokojeni, bude možné navrhnout lepší řešení.

4.1.2.1 Používané nástroje

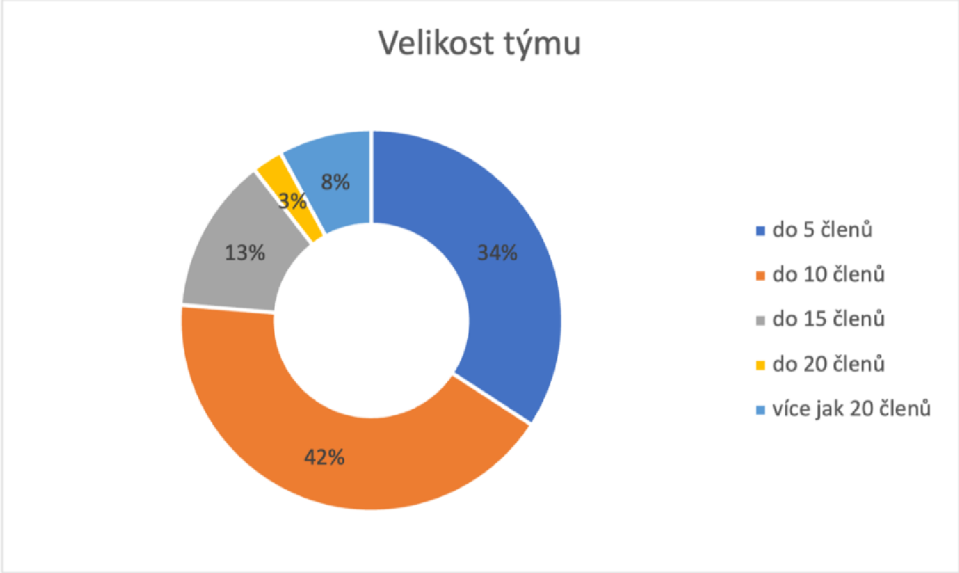
V první otázce se zjišťuje, jaký nástroj respondenti používají. Většina uvedla, že používá české Freelo. Pětina používá Trello, dvanáct procent uvedlo, že používá aplikaci Jira. Zbytek používá jiné nástroje jako jsou Basecamp, Asana, Youtrack a další.



Obrázek 6 Graf – používané nástroje

4.1.2.2 Velikost týmu

Zde respondenti odpovídali, v jak velkém týmu pracují a využívají daný nástroj. Z výsledků lze vidět, že většina lidí používá nástroj v menších týmech do deseti lidí.



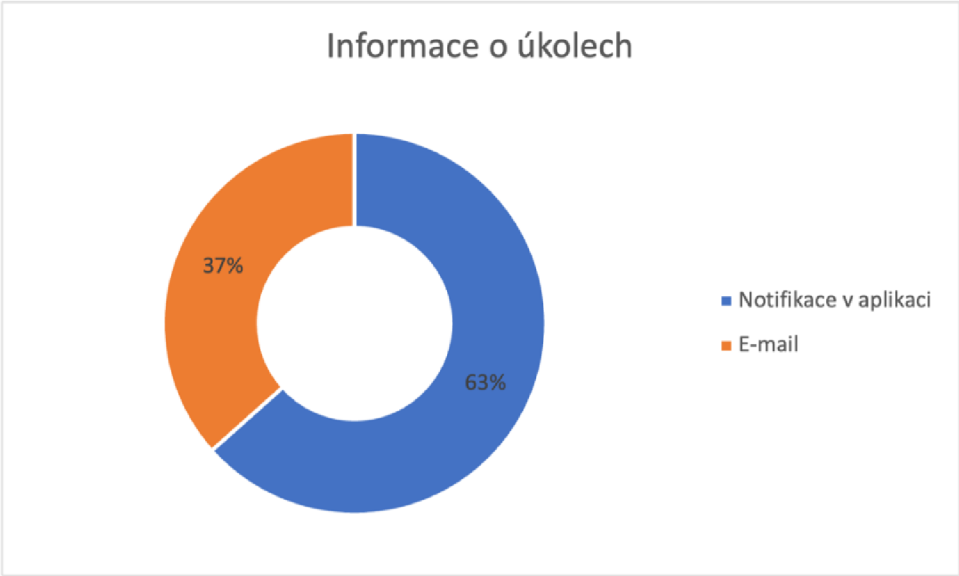
Obrázek 7 Graf – velikost pracovního týmu

4.1.2.3 Efektivita díky aplikaci

V další otázce respondenti odpovídali, zda jsou v práci více efektivní díky aplikaci. V této otázce všichni uvedli, že jsou díky nástroji v práci více efektivní.

4.1.2.4 Notifikace o úkolu

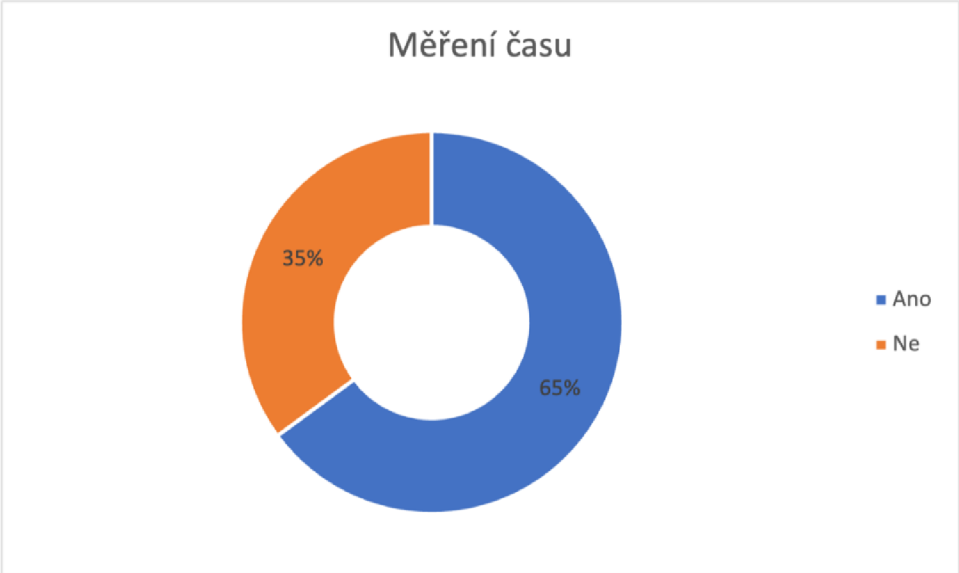
Ve čtvrté otázce se zjišťuje, jakým způsobem jsou uživatelé informováni o přiřazeném úkolu nebo nějaké změně v systému. Většina uvedla, že o změnách jsou informováni přes notifikaci v aplikaci a více jak třetina využívá především e-mailové zprávy.



Obrázek 8 Graf – informace o úkolech

4.1.2.5 Použití měření času

V této otázce respondenti odpovídali, jestli si v práci měří odpracovaný čas. Skoro dvě třetiny respondentů uvedly, že si čas neměří.



Obrázek 9 Graf – měření času

4.1.2.6 Použití nástroje podle zařízení

Tato otázka zjišťuje, na jakém zařízení aplikaci uživatelé používají. Více jak 80 % uvedlo, že nástroj používají výhradně na počítači. Zbytek nástroj používá na mobilním zařízení a počítači stejně.

4.1.2.7 Vyhovující funkce/vlastnosti

Zjišťuje se, které funkcionality se respondentům na nástroji líbí. Respondenti zde mohli vybrat více odpovědí, anebo napsat vlastní. Nejčastější odpovědi jsou zobrazeny v následující tabulce. Z výsledků si lze všimnout, že respondenti nejvíce oceňují snadné přiřazování úkolů, možnosti filtrace úkolů a vytváření šablon.

Funkce/vlastnosti
Přiřazování úkolů
Filtry nad úkoly
Šablony úkolů/projektů

Mobilní aplikace
Možnost měření času
Reporting
Kanbanová nástěnka

Tabulka 1 vyhovující funkce/vlastnosti

4.1.2.8 Nevyhovující funkce/vlastnosti

V této otázce respondenti vybírali naopak ty vlastnosti, se kterými nejsou spokojeni. Nejčastější odpovědi jsou opět zobrazeny v tabulce. Většina respondentů uvedla, že není nic, co by jim v aplikaci nevyhovovalo. Lze tedy usoudit, že nástroje nemají žádné velké nedostatky a uživatelům se s nimi pracuje dobře. Nicméně objevilo se několik odpovědí, které uvádějí, že uživatelé nejsou spokojeni hlavně s prací nad úkoly a filtrováním.

Funkce/vlastnosti
Práce s úkoly
Filtrace úkolů
Kanbanová nástěnka
Nastavení práv
Cena

Tabulka 2 Nevyhovující funkce/vlastnosti

4.1.2.9 Změna/vylepšení aplikace

V této otázce respondenti měli napsat, zda existuje v nástroji něco, co by si přáli vylepšit či změnit. Odpovídalo se možnostmi ano/ne. V případě kladné odpovědi, pak respondenti mohli v další otázce vypsat, co přesně by si přáli změnit. Dvě třetiny respondentů uvedly, že by nic neměnily.

V další otázce respondent zadával, kterou konkrétní část by si přál vylepšit, nebo změnit. Uvedené odpovědi jsou: řazení úkolů, zavedení systému práv, rychlost aplikace, lepší možnosti exportu reportů, lépe odlišit úkoly a podúkoly, přidat časový odhad pro úkoly, zlepšení filtrace úkolů, počítání bodů v historii sprintu a zlepšená možnost editace kanban nástěnky.

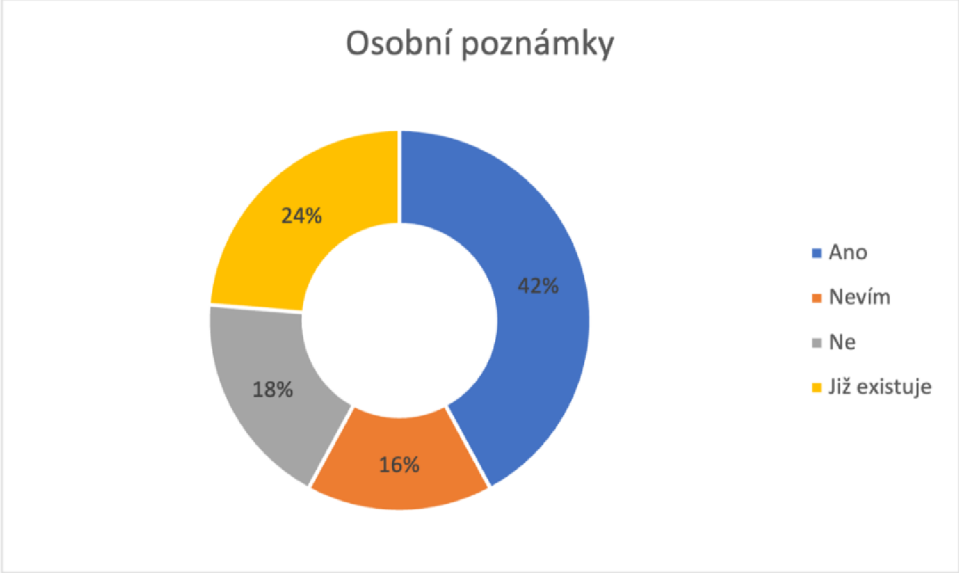
4.1.2.10 Absence funkcionality

Zde respondenti uváděli, zda jim nějaká funkcionalita v aplikaci schází. Odpovídalo se možnostmi ano/ne. Přes dvě třetiny respondentů uvedlo, že jim nic nechybí.

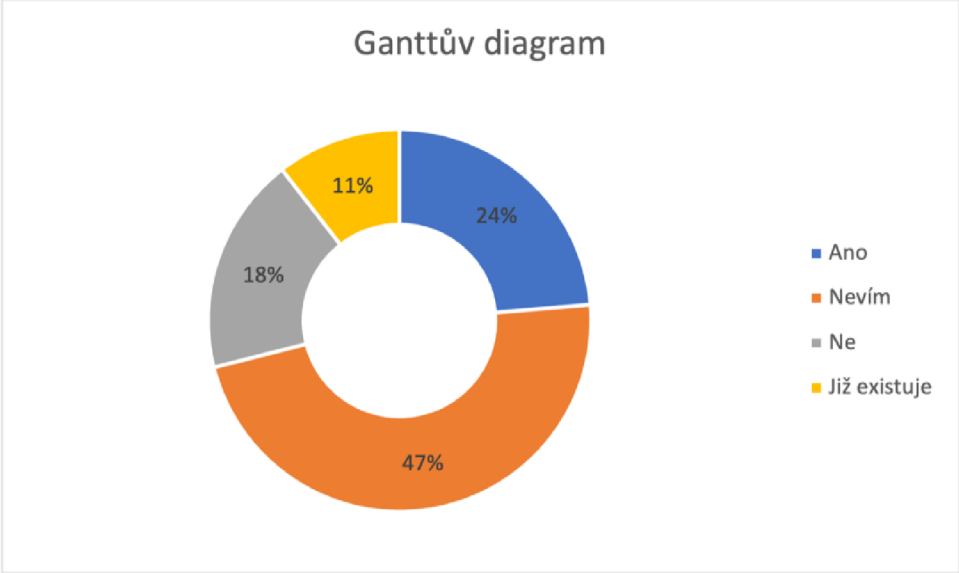
V další otázce byla možnost vypsat, co jim v aplikaci chybí. Uvedené odpovědi jsou: možnost úpravy vzhledu aplikace, jednoduchá možnost zobrazit nadřazené a související úkoly, možnost přiřazení uživatelů do pracovních skupin, zobrazení data splnění úkolu, a Ganttův diagram.

4.1.2.11 Navrhované funkcionality

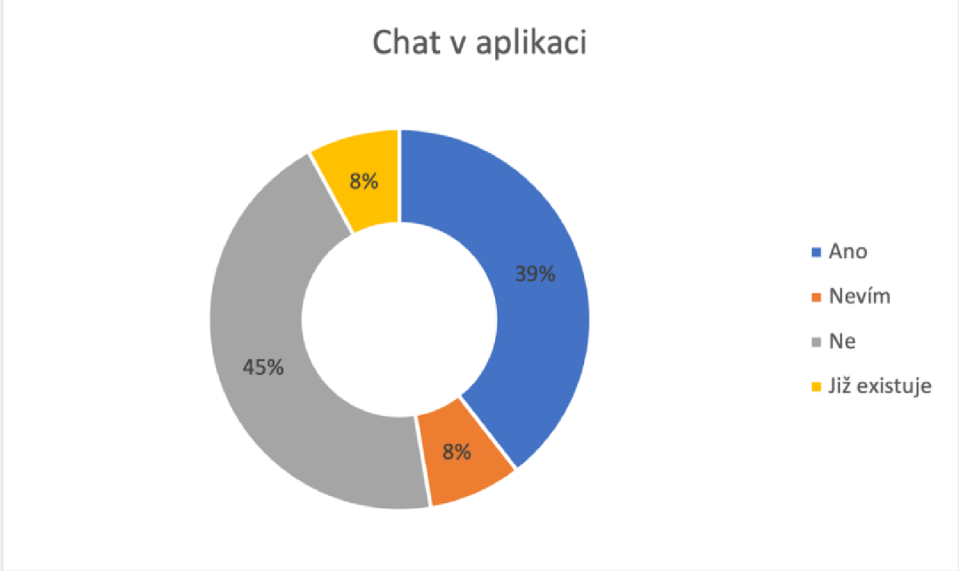
V dalších otázkách byly sepsány čtyři funkcionality, které by mohla výsledná aplikace obsahovat. Respondenti na ně odpovídali podle toho, zda by se jim líbilo mít tuto funkcionalitu v aplikaci, nebo zda již tato funkcionalita v jejich nástroji existuje. Seznam funkcionalit byl sestaven po konzultaci se zadavatelem.



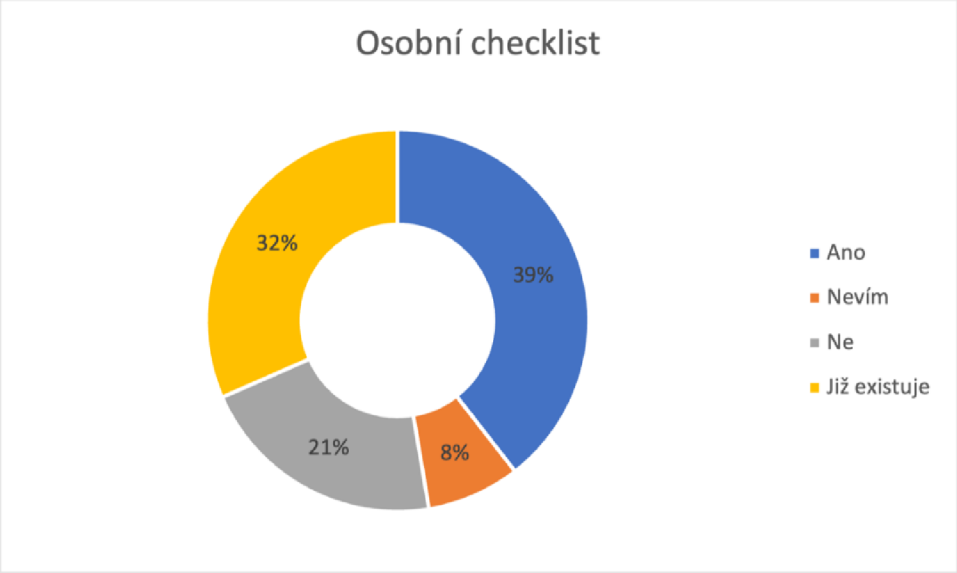
Obrázek 10 Graf – osobní poznámky



Obrázek 11 Graf – Ganttův diagram



Obrázek 12 Graf – chat



Obrázek 13 Graf – osobní checklist

Ze zobrazených výsledků je vidět, že respondenti by měli největší zájem o přidání osobních poznámek a checklistů. Je také vidět, že velké množství nástrojů již tyto funkcionality nabízí. Otázku o možnosti mít v aplikaci chat skoro polovina respondentů odmítla. Také pouhých osm procent respondentů uvedlo, že jejich nástroj chat nabízí. Na otázku, zda by chtěli mít v aplikaci Ganttův diagram 47 % respondentů uvedlo, že neví. Naopak 24 % uvedlo, že by tuto funkcionalitu uvítali a 11 %, že již tuto funkcionalitu mají. Velké množství odpovědí nevím na otázku ohledně Ganttova diagramu nejspíše vyplývá z toho, že s tímto diagramem respondenti ještě nepřišli do styku, nebo neví, jakým způsobem by tento diagram využili.

4.1.2.12 Vyhodnocení

Z dostupných výsledků dotazníku lze zjistit, že uživatelům se s vybranými nástroji pracuje dobře a nemají žádné velké problémy při jejich používání. Proto si lze z těchto aplikací vzít při návrhu příklad. Ovšem někteří respondenti uvedli, že nejsou se vším spokojeni. Týká se to hlavně práce s úkoly a jejich filtrování. Proto je potřeba dát na tuto část aplikace při návrhu a implementaci větší pozor. Především musí být tato sekce jednoduchá a intuitivní pro použití.

Z nabízených funkcionalit byl největší zájem o možnosti mít své osobní poznámky a checklisty, proto by bylo dobré tyto sekce do návrhu zakomponovat.

4.1.3 Další vlastnosti

Jedná se hlavně o ty vlastnosti, které by měla mít každá takováto aplikace, aby bylo možné s ní pracovat. Aplikace musí mít správu uživatelů a skupin. Každý uživatel bude mít roli, která mu umožní používat vybrané moduly a využívat další funkce v systému. Uživatele by mělo být možné přiřazovat do pracovních skupin. Hlavním modulem bude správa projektů. K projektům půjde přiřazovat jednotlivé uživatele nebo celé skupiny, které na projektu pracují. Projekt bude mít své pracovní nástěnky, ve kterých budou kanban nástěnky s možností upravovat jednotlivé sloupce. Půjde zde vytvářet jednotlivé pracovní úkoly. Ty půjde přiřazovat uživatelům. Úkolům půjde nastavovat základní parametry jako jsou priorita a termín dokončení. Úkol bude moct mít svůj checklist, přílohy a komentáře. O změnách v úkolu budou uživatelé informováni díky notifikacím. Notifikace budou, jak v rámci aplikace, tak budou odesílány na e-mail.

Další modul je měření času. Uživatel bude moct měřit čas, který strávil na zadaném úkolu. Statistiky záznamů o práci pak bude možné filtrovat a exportovat. Manažer bude mít možnost pak sledovat i odpracovaný čas jednotlivých uživatelů.

Každý uživatel bude mít svůj osobní checklist, který uvidí jen sám a bude si tam moct zapisovat a odškrtávat své poznámky/úkoly.

4.2 Struktura

Výsledný systém je rozdělen do dvou částí. Na klientskou část a serverovou část. Klientská část bude interagovat s uživatel a zobrazovat mu data. Serverová část se bude starat o práci s daty (ukládání, mazání, editaci), validovat uživatelské vstupy a odpovídat na požadavky klienta.

4.2.1 Architektura

Aplikace je navržena do třívrstvé architektury. Tato architektura rozděluje aplikaci na tři části: prezentační, aplikační a datovou vrstvu. Prezentační vrstva je zodpovědná za zobrazení dat a předání vstupů od uživatele aplikační vrstvě. Aplikační vrstva zpracovává požadavky z prezentační vrstvy, provádí výpočty a komunikuje s datovou vrstvou. Datová vrstva zodpovídá za práci s daty.

4.2.2 Klientská část

Je navržena jako SPA (Single Page Application). SPA je webová aplikace, která komunikuje s uživatelem dynamickým přepisováním aktuální webové stránky novými daty z webového serveru namísto načítání celých stránek. Aplikace pak vypadá více jako nativní aplikace. V SPA se stránka nikdy neobnoví celá, místo toho je veškerý potřebný kód HTML, JavaScript a CSS buď načten při prvním načtení stránky, nebo jsou příslušné zdroje dynamicky načteny a přidány na stránku podle potřeby.

Oproti vícestránkovým aplikacím má tedy tu výhodu, že se stránka nemusí při každé změně nebo akci obnovit celá. Uživatel tak nevidí, jak se celá stránka načítá pokaždé znovu, a proto je práce s aplikací plynulejší.

4.2.3 Serverová část

Aby mohla klientská část komunikovat se serverem, tak je serverová aplikační část navržena jako webové API. Aplikační vrstva předává požadavky datové části, která pracuje s databází.

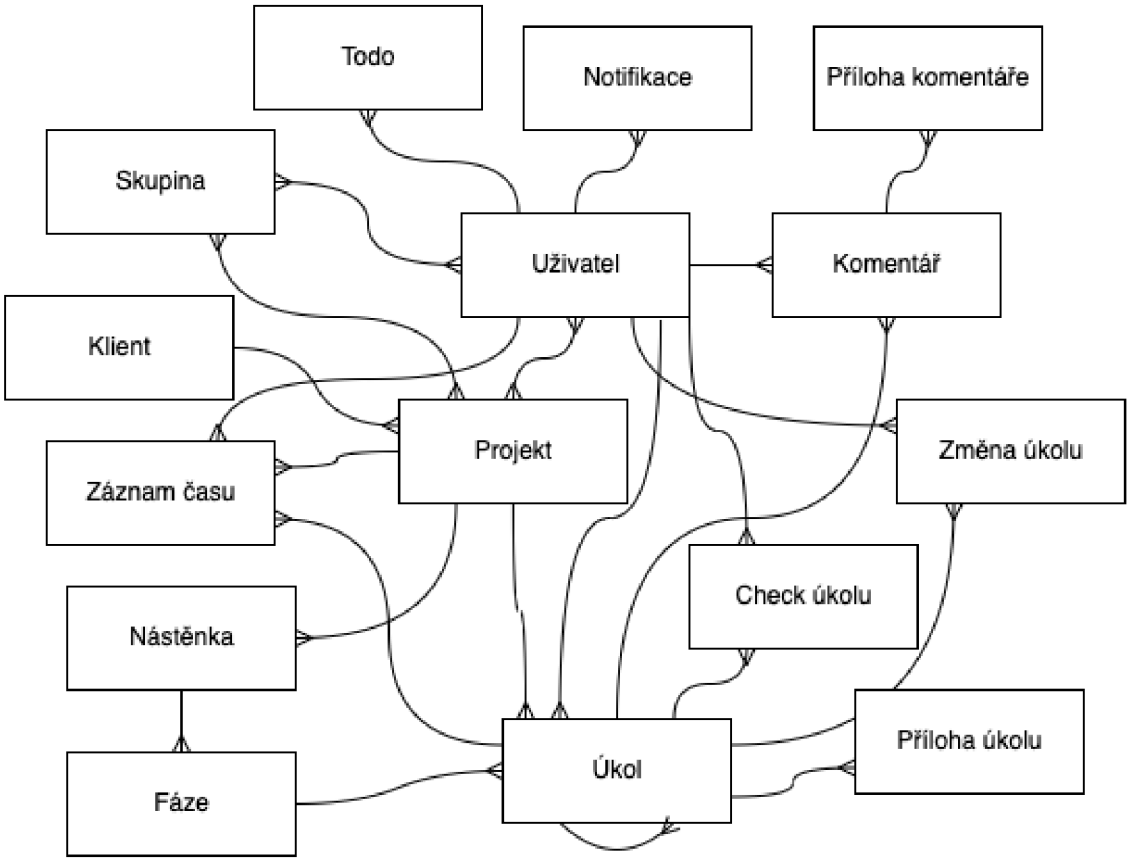
4.2.3.1 API

Pro komunikaci mezi klientem a serverem bude sloužit API. Celé API bude mít několik koncových bodů (end point), které budou obsluhovat požadavky klienta. Každá entita by měla mít své oddělené koncové body a klient bude volat ty, které zrovna potřebuje. Takto členěné API bude více přehledné a půjde jednoduše rozšiřovat. Většina koncových bodů by měla být dostupná pouze autentizovaným uživatelům. Volně přístupné by měly být pouze body pro přihlášení a obnovu hesla. Klient bude API volat přes protokol HTTP. Tento protokol umožňuje používat několik typů volání. Nejčastější jsou GET, POST, PUT a DELETE. Tyto metody volání se používají pro různé situace a použitím těchto typů volání lze dobře strukturovat API. Metoda GET slouží pro získání dat, POST pro vytváření, PUT pro úpravu a DELETE pro mazání dat. Všechny koncové body by měly klientovi vracet odpověď ve formátu JSON, který je dnes hojně rozšířený a dokáže s ním pracovat spoustu jazyků.

4.2.4 Datový návrh

Základními entitami budou: uživatel, projekt, úkol, nástěnka, odpracovaný čas, notifikace a skupina.

Uživatel bude moci být přiřazen do skupiny. Bude mít svoje odpracované pracovní záznamy, soukromé úkoly a notifikace. Projekt bude moci mít jednu nebo více nástěnek, které budou sloužit například k definování sprintů, nebo jen pro rozdělení projektu na více částí. Nástěnka bude mít své pracovní fáze, kterými se bude definovat pracovní proces. Tyto fáze budou zobrazeny v nástěnce jako jednotlivé sloupce. K těmto sloupcům půjdou přiřadit jednotlivé úkoly. Projektu bude možné přiřadit jednotlivé uživatele a skupiny. Úkol bude mít své přílohy, komentáře, checklisty a podúkoly. Celý návrh datového modelu je zobrazen v následujícím ER diagramu.



Obrázek 14 ER diagram

4.2.5 Moduly aplikace

Níže je seznam základních částí aplikace, které budou k dispozici.

4.2.5.1 Administrace

Tento modul slouží pro správu aplikace a měl by být přístupný jen uživatelům s příslušným právem. Obsahuje tři základní části: správu uživatelů, skupin a klientů.

V sekci uživatelů bude seznam všech uživatelů, kteří jsou v aplikaci. Uživatel, který má příslušná práva, bude moci uživatele spravovat, přidávat, upravovat a mazat. V této sekci bude jediná možnost, jak půjde v aplikaci vytvářet uživatelské účty.

Sekce skupiny slouží pro správu pracovních skupin v systému. Ve skupině pak půjde spravovat přiřazené uživatele. Tyto skupiny pak půjde přiřazovat například k projektům. Zadavatel tak nebude muset všechny uživatele k projektu přiřazovat jednotlivě, ale bude je moci přiřadit najednou v rámci skupiny.

Sekce klientů bude obsahovat záznamy klientů, se kterými se spolupracuje. Bude se jednat především o seznam s kontaktními informacemi.

4.2.5.2 Projekty

Toto bude hlavní modul celé aplikace. V tomto modulu bude seznam projektů, ke kterým má uživatel přístup. Bude zde i možnost jednotlivé projekty spravovat a přidávat nové. Uživatel bude moct jít do detailu projektu, kde bude seznam pracovních nástěnek. Tyto nástěnky pak budou obsahovat kanban nástěnku s jednotlivými úkoly. Zde bude možné úkoly spravovat.

4.2.5.3 Měření času

Tento modul bude obsahovat dvě části. V jedné bude možnost zadávat a zobrazit svůj odpracovaný čas a upravovat starší záznamy. Druhá bude sloužit pro zobrazení počtu odpracovaného času v aplikaci za vybrané období. Podle role bude mít uživatel možnost zobrazit si záznamy všech uživatelů, nebo jen své. Tyto záznamy pak bude možné exportovat do csv.

4.2.5.4 Moje checklisty

Jedná se o modul, který bude dostupný vždy jen pro aktuálního uživatele. Uživatel si zde bude moct vytvářet osobní poznámky/malé úkoly, které má splnit. Tyto záznamy budou dostupné jen pro něj a nikdo jiný k těmto záznamům nebude mít přístup.

4.2.5.5 Notifikace

Uživateli budou chodit notifikace ohledně toho, co se v aplikaci právě děje. Uživatel bude mít možnost si zobrazit všechny své příchozí zprávy v této sekci.

4.2.5.6 Uživatelský profil

Jedná se o sekci, která bude dostupná pro každého uživatele. V této sekci si bude moct uživatel měnit osobní nastavení a informace.

4.2.6 Uživatelské role

Uživatel je základní entita aplikace, která představuje reálnou osobu. Tento uživatel interaguje se systémem a provádí v něm změny. Aby každý uživatel neměl přístup ke všem částem systému, tak budou vytvořeny tři základní role, které vymezují, co může daný uživatel v aplikaci dělat.

4.2.6.1 Uživatel

Jedná se o základní roli aplikace, kterou by měl mít řadový zaměstnanec. Tato role nebude mít přístup do modulu administrace a nebude moct vytvářet nové projekty. Může přistupovat k přiřazeným projektům, kde bude moct vytvářet úkoly, měnit jejich stav a přidávat k nim komentáře.

4.2.6.2 Manažer

Manažer bude mít přístup do administračního modulu. A stejně jako uživatel uvidí jen své projekty. Může je vytvářet a editovat. K projektům bude moct přiřazovat pracovní skupiny a uživatele.

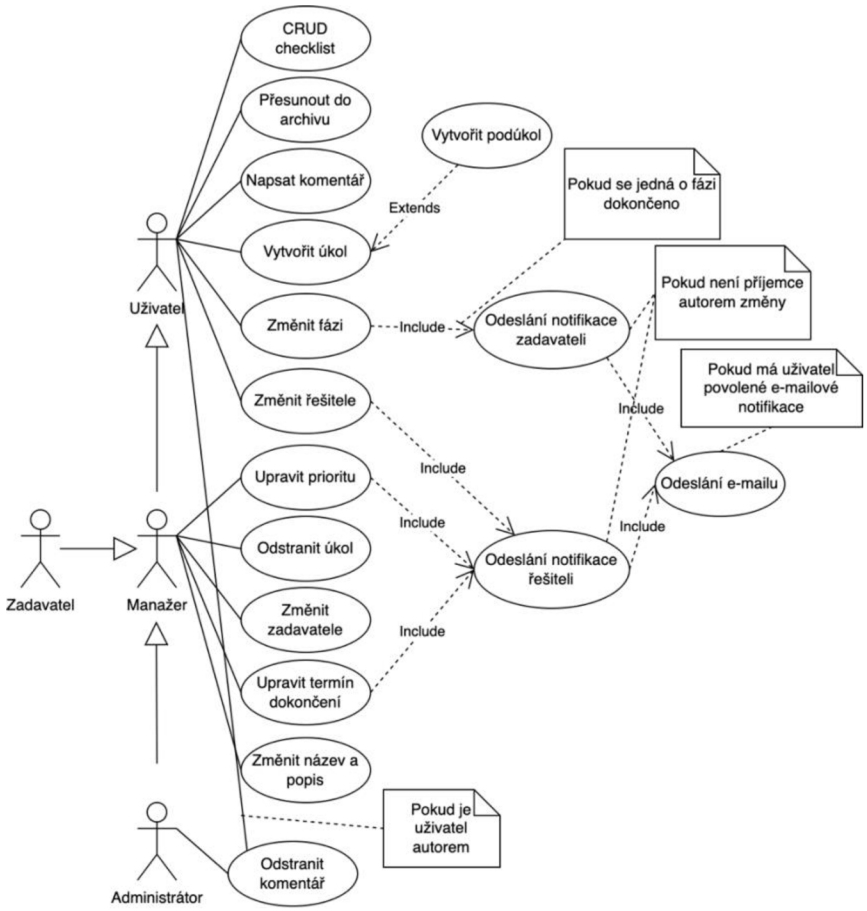
4.2.6.3 Administrátor

Uživatel s touto rolí bude mít nejvyšší pravomoc. Bude mít přístup v systému ke všem sekcím, uvidí všechny projekty v systému. Může mazat všechny komentáře, úkoly a projekty v systému.

Tato role je určena ke správě aplikace a neměla by být příliš zneužívána, jelikož pro běžnou práci s aplikací by měly stačit předchozí role.

4.3 Návrh modulu – práce s úkoly

Jedná se o základní část aplikace. Uživatelé by si v úkolu měli zadávat práci, předávat informace, upravovat jeho atributy a komunikovat spolu. Každý úkol bude mít zadavatele, který úkol vytvořil. Tento uživatel bude moct spravovat úkol, měnit termín, prioritu atd. Zadavatele bude možné změnit. K úkolu půjdou přidávat zaškrtávací malé úkoly a komentáře. Úkolu půjde přiřadit řešitele, ten by měl být o přiřazení informován notifikací a měl by mít na starosti dokončení úkolu. Po dokončení úkolu bude informován zadavatel. Notifikace by ovšem neměly chodit, pokud autor změny je i příjemce. Dále bude mít úkol seznam příloh, časový odhad, barvu a podúkoly. Úkol bude možné přesunout do jiné nástěnky a archivu, aby se dále nezobrazoval na kanbanové nástěnce. Uživatel bude moct úkol upravovat podle své role. Například prioritu a termín dokončení bude moct upravovat jen uživatel s rolí manažer nebo administrátor. Výjimku bude mít běžný uživatel, pokud bude nastaven jako zadavatel úkolu. Případy užití jsou znázorněny v následujícím diagramu.



Obrázek 15 *Use case – práce s úkolem*

5 Implementace

5.1 Technologie

V následujících kapitolách jsou sepsány jednotlivé technologie, které byly použity při vývoji aplikace.

5.1.1 JavaScript

„JavaScript (často zkrácený na JS) je interpretovaný, objektově orientovaný jazyk s mnoha funkcemi. Je nejvíce známý jako skriptovací jazyk pro webové stránky, ale používá se i v mnoha prostředích bez prohlížeče. Jedná se o prototypový skriptovací jazyk s mnoha paradigmaty, který je dynamický a podporuje objektově orientované, imperativní a funkční styly programování.“ [30]

JavaScript běží na klientské straně webu a umožňuje, aby stránka byla více interaktivní pro uživatele. Umožňuje například použití mobilního tzv. hamburger menu, přidat/odebrat obsah webu nebo ho měnit. Dále je možné vytvářet celé webové i mobilní aplikace, k tomu existuje několik frameworků. A v poslední řadě je i možnost vytvářet serverové aplikace.

5.1.2 Asynchronnost

„Asynchronní programování je forma paralelního programování, která umožňuje jednotce práce běžet odděleně od primárního aplikačního vlákna. Když je práce dokončena, upozorní hlavní vlákno (a také to, zda byla práce dokončena nebo selhala). Jeho používání má řadu výhod, jako je lepší výkon aplikací a lepší odezva.“ [31]

Naopak synchronní kód se vykonává postupně, jak je napsán řádek po řádku. Čeká se na dokončení jednotlivých operací a následující operace se provede teprve až je předchozí dokončena. Jelikož je JavaScript jednovláknový, tak jakákoliv operace zabere vlákno a uvolní ho až po jejím dokončení. To může skončit zamrznutím prohlížeče.

Pokud je kód napsán asynchronně, tak se vloží do fronty událostí, kde se čeká až se dokončí. Jakmile je operace dokončena a neprobíhá aktuálně žádný synchronního kód, tak se vezme z fronty událostí a vlákno zpracuje výsledek. Tímto je umožněno vykonávat více operací najednou.

5.1.3 Frontend – Klientská část

Jedná se o tu část aplikace, se kterou uživatel přímo interaguje. Někdy se také nazývá klientská část aplikace. Zahrnuje vše, co uživatel vidí například text, obrázky a styly. Jazyky pro zobrazení těchto dat jsou HTML, CSS a JS. HTML (Hypertext Markup Language) je značkovací jazyk, který definuje strukturu a obsah webu. CSS (Cascading Style Sheets) je jazyk pro stylování HTML elementů, popisuje, jak mají jednotlivé elementy být vykresleny. JS je skriptovací jazyk, který umožňuje, aby stránka interagovala s uživatelem.

S rozvojem JavaScriptu začaly být webové aplikace příjemnějšími pro použití, jelikož JavaScript běží v prohlížeči a nemusí se proto obnovovat stránka, aby se jakákoliv změna projevila. Uživatel proto má lepší uživatelský zážitek při používání takovéto aplikace. První velká knihovna, která usnadňovala použití JavaScriptu a poskytovala lepší uživatelskou zkušenost (UX) byla jQuery, která byla vydána v roce 2006. V té době to byla nejoblíbenější knihovna pro webová rozhraní a používá se dodnes. Avšak tato knihovna má nedostatky. Čím více se používá, tím je kód více nepřehledný, nedokáže uchovávat a pracovat s daty napříč různými stránkami aplikace. Proto tato knihovna není vhodná pro tvorbu moderních aplikací. Tyto nedostatky odstraňují frameworky a modernější knihovny.

Od roku 2010 začaly vznikat JavaScriptové frameworky, které posunuly webový vývoj na další úroveň. Umožnily totiž vytvářet SPA.

Dobrá UX je jednou z největších priorit společností pro jejich webové aplikace, uživatele nezajímá, co se děje na pozadí aplikace, ale jde mu o to, co vidí a s čím pracuje, proto je velmi důležité, aby se uživatelům pracovalo s aplikací co nejlépe. Té lze jednodušeji dosáhnout díky SPA.

Aplikace jako jsou Facebook, Netflix, Twitter a Instagram používají frontendové frameworky a knihovny pro zlepšení UX na jejich webových a někdy i na mobilních aplikacích.

Dnes existuje mnoho JavaScriptových frameworků a knihoven jako jsou Angular, Vue.js, React a Svelte. Někdy není jednoduché se rozhodnout pro jeden. Všechny mají plno skvělých funkcí. Nakonec byla pro implementace klientské části aplikace vybrána JavaScriptová knihovna React od společnosti Facebook (Meta). Hlavně z důvodu velké komunity okolo této knihovny, jednoduchému použití a existence mnoha návodů.

5.1.3.1 React

React je JavaScriptová knihovna pro vývoj moderních webových aplikací. Vytvořila ji a spravuje ji společnost Meta. První verze byla vydaná v roce 2013. Nyní je jednou z nejčastěji používaných frontendových knihoven pro vývoj webových aplikací. Podle průzkumu webu stackoverflow [32] byl za rok 2021 React zvolen jako nejoblíbenější webový framework.

React pro tvorbu aplikace používá React komponenty, ze kterých se skládá celá aplikace. Dokončená aplikace jich může obsahovat stovky. Komponenty mají svou logiku, stav, mohou být zanořené v sobě a lze je znovupoužít v aplikaci. Díky tomu lze aplikace vytvářet rychleji a efektivněji. Komponenty lze psát v JSX syntaxi.

„JSX je syntaxe podobná XML/HTML používaná v Reactu, která rozšiřuje ECMAScript tak, aby text podobný XML/HTML mohl koexistovat s kódem JavaScriptu/Reactu. Syntaxe je určena pro použití preprocesory (tj. transpilery jako Babel) k transformaci textu podobného HTML, který se nachází v souborech JavaScriptu, na standardní objekty JavaScriptu.“ [33]

5.1.3.1.1 Redux

Je stavový kontejner pro JavaScriptové aplikace. Umožňuje uchovávat stav aplikace v jednom úložišti. Každá React komponenta má k tomuto úložišti přístup a může přistupovat k libovolnému stavu, který potřebuje. Správné použití Reduxu umožňuje zpřehlednit kód a zjednodušit jeho správu, jelikož je všechen stav aplikace uložen na jednom místě.

Redux se dá rozdělit na tři základní části: akce, reducer a store. Akce jsou události, které odesílají data z aplikace do storu. Data mohou být například vstupy od uživatele. Reducery jsou funkce, které podle akce mění stav aplikace. A store obsahuje aktuální stav aplikace.

Redux není potřeba používat v každé React aplikaci. Lze ho použít i u jiných frameworků/knihoven. Naopak u menších aplikací může být použití Reduxu spíše kontraproduktivní. Správné nastavení a použití Reduxu zabere více času, který se u menších aplikací nemusí vyplatit. Redux je tedy vhodnější použít ve větších aplikacích.

5.1.3.2 Bootstrap

Je CSS Framework, který usnadňuje a urychluje stylování webových komponent, díky předpřipraveným stylům. Poprvé se objevil v roce 2011. Jedná se o velmi populární framework, který umožňuje vytvářet responzivní webové aplikace. Aktuálně je dostupný ve verzi 5. Pro vývoj aplikace byla použita verze 4.

5.1.4 Backend – Serverová část

Backend, je ta část webové aplikace, kterou uživatel nevidí. Nachází se totiž na serveru a stará se o generování webových stránek, přístup k databázi, zpracování klientských požadavků atd. Existuje celá řada jazyků a technologií pro vytvoření serverové části aplikace. Jako příklad lze uvézt PHP, .Net a Java. S příchodem Node.js přibyla možnost vyvíjet backendovou část aplikace také v JavaScriptu. To umožnilo programátorům, kteří vyvíjeli v JavaScriptu frontend, vytvářet i backend díky jednomu programovacímu jazyku. Serverová část aplikace byla vytvořena na JavaScriptovém frameworku Express.js, jako REST API.

5.1.4.1 REST API

Je soubor architektonických omezení nikoli protokol nebo standard. REST lze implementovat různými způsoby. Aby mohlo být API považováno za REST, tak musí splňovat následující kritéria.

- Architektura klient-server s požadavky řízenými prostřednictvím HTTP.
- Komunikace mezi klientem a serverem je bezstavová.
- Data jsou uložená v mezipaměti.
- Jednotné rozhraní, které umožňuje klientovi hovořit se serverem v jediném jazyce, nezávisle na architektuře backendu. Toto rozhraní by mělo poskytovat neměnné, standardizované prostředky pro komunikaci mezi klientem a serverem.
- Vrstvený systém umožňuje, aby architektura byla složena z hierarchických vrstev s omezením chování komponent.
- Kód na vyžádání (volitelně) dává možnost odeslat spustitelný kód ze serveru klientovi na požádání, čímž se rozšíří funkčnost klienta. [34]

5.1.4.2 Node.js

Node.js je multiplatformní a open-source runtime prostředí pro vývoj serverových aplikací. Byl vydán v roce 2009 a je pod licencí MIT. Běží na JavaScriptovém enginu V8 od společnosti Google. Je napsán v programovacích jazycích C, C++ a JavaScript.

„Aplikace Node.js běží v jediném procesu bez vytváření nového vlákna pro každý požadavek. Node.js poskytuje ve své standardní knihovně sadu asynchronních I/O primitiv, které zabraňují blokování kódu. Při provádění I/O operací, jako jsou čtení ze sítě, přístup k databázi nebo souborovému systému namísto blokování vlákna a plýtvání CPU čekáním, Node.js obnoví operace, až když se odpověď vrátí. To umožňuje Node.js zpracovávat tisíce souběžných připojení s jediným serverem bez zavádění zátěže související se správou souběžnosti vláken.“ [35]

„V8 je název JavaScript enginu, na kterém běží Google Chrome. Je to ta věc, která bere kód JavaScriptu a spouští ho při procházení Chromu. V8 poskytuje běhové prostředí, ve kterém se spouští JavaScript. Je nezávislý na prohlížeči, ve kterém je hostován. Tato klíčová funkce umožnila vzestup Node.js.“ [36]

„Blokování se týká zablokování další operace, dokud aktuální operace neskončí. Blokovací metody se provádějí synchronně. Synchronně znamená, že program je spouštěn řádek po řádku. Program čeká, dokud se volaná funkce nebo operace nevrátí. Jako neblokující se označuje program, který neblokuje provádění dalších operací. Neblokující metody jsou prováděny asynchronně. Asynchronně znamená, že program nemusí nutně provádět řádek po řádku. Program zavolá funkci a přesune se k další operaci a nečeká na její návrat.“ [37] Rozdíl mezi blokováním a neblokováním procesu v Node.js lze ukázat na následující ukázce kódu, který má za cíl načíst soubor.

Načtení souboru synchronním způsobem, který zablokuje ostatní operace, dokud nebude operace dokončena.

```
const fs = require('fs');
const data = fs.readFileSync('soubor.txt');
```

Zde je zobrazen stejný kód, který je napsán asynchronně, a proto neblokuje další operace.

```
const fs = require('fs');
fs.readFile('soubor.txt', (err, data) => {
  if (err) throw err;
});
```

5.1.4.2.1 Moduly

„V Node.js jsou moduly bloky zapouzdřeného kódu, které komunikují s externí aplikací na základě souvisejících funkcí. Moduly mohou být jedním souborem nebo sbírkou více souborů/složek.“ [38] Důvod proč se hojně používají je ten, že je lze znovu použít a umožňují aplikaci rozdělit na menší a přehlednější části.

Moduly jsou rozděleny na tři typy: základní moduly, lokální moduly a moduly třetích stran. Základní moduly jsou součástí Node.js. Jedná se o moduly, které pracují se souborovým systémem, operačním systémem atd. Lokální jsou v rámci aplikace, které programátor vytvoří a dále používá v rámci aplikace. Moduly třetích stran mohou být vytvořeny komunitou a jsou dostupné díky npm.

5.1.4.2.2 NPM

Node Package Manager je balíčkový manažer pro Node.js moduly. „NPM jsou dvě věci, zaprvé je to online úložiště pro publikování open-source projektů Node.js. Zadruhé je to nástroj příkazového řádku pro interakci s uvedeným úložištěm, který pomáhá při instalaci balíčků, správě verzí a závislostí.“ [39]

5.1.4.3 Express.js

Express.js nebo jen Express je open-source webový framework pro Node.js s MIT licencí. V aktuální době je to nejpopulárnější framework pro Node.js. Používá MVC architekturu (Model, View, Controller). To, že je velmi oblíbený, značí i to, že na Githubu má přes 55 tisíc hvězdiček. Vyšel v listopadu 2010 a aktuální verze je 4 a pracuje se na verzi 5, která je momentálně ve verzi alpha. Vývoj aplikací čistě na Node.js je časově náročnější a komplikovanější pro programátory, a především pro ty méně zkušené. Express se postará o základní nastavení aplikace jako je nastavení portu a zpracování požadavku. Express umožňuje vytvářet, jak full-stack webové aplikace, tak i API nebo microservice. Je to minimalistický framework, což znamená, že není tak robustní a sám o sobě má jen pár funkcionalit. Další je potřeba naimportovat z různých knihoven nebo doprogramovat.

Express podporuje relační i NoSql databáze jako jsou MySQL, MongoDB, Redis, Cassandra, Elasticsearch a další. Pro použití zvolené databáze je zapotřebí zvolit vhodný ovladač, který je nutný do aplikace nainstalovat.

5.1.4.4 Socket.io

„WebSocket je obousměrný, plně duplexní komunikační protokol iniciovaný přes HTTP. Běžně se používá v moderních webových aplikacích pro streamování dat a další asynchronní provoz.“ [40]

Socket.io je knihovna, která využívá WebSocket protokol, díky kterému navazuje obousměrnou real-time komunikaci mezi klientem a serverem. Tuto knihovnu lze použít v moderních aplikacích, jako jsou různé chaty nebo online hry, kde je potřeba mít přístup k aktuálním datům. Tato knihovna je v aplikaci využita proto, aby uživatel měl k dispozici aktuální data a nemusel pokaždé obnovovat stránku, aby měl aktuální informace. Uživatel tak bude mít možnost vidět všechny změny v nástěnce, které se právě dějí nebo získávat ihned notifikace ohledně dění v aplikaci.

5.1.5 Databáze

Datová struktura je pevně daná, a proto není důvod volit NoSQL databázi, která se dá použít hlavně, pokud struktura dat není pevně daná. Proto byla zvolena relační MySQL databáze. MySQL je open-source relační databázový systém. Pro usnadnění práci s databází byl použit ORM modul Sequelize.

5.1.5.1 Sequelize

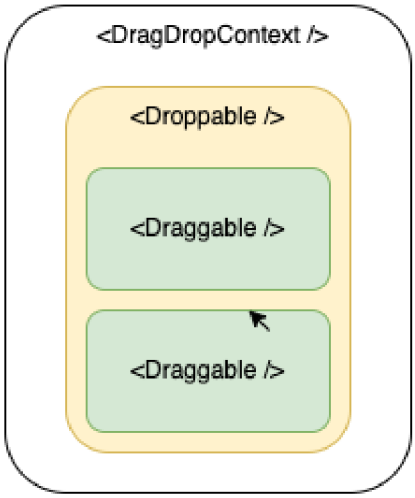
Sequelize je ORM (objektově relační mapování) pro Node.js aplikace. Sequelize podporuje řadu databází jako je PostgreSQL, MySQL, MariaDB a další.

„ORM automatizuje přenos dat uložených v tabulkách relačních databází do objektů, které se běžněji používají v kódu aplikace. Poskytuje abstrakci na vysoké úrovni nad relační databází, která umožňuje vývojářům psát kód v programovacím jazyce např. Java místo SQL pro přidávání, čtení, aktualizaci a mazání dat. Vývojáři mohou pro práci s databází místo psaní příkazů SQL nebo uložených procedur použít programovací jazyk, který jim vyhovuje.“ [41]

5.2 Implementace modulu – práce s úkoly

Pro zobrazení úkolů byly v aplikaci vytvořeny tři stránky. Dvě slouží pro zobrazení nepřiřazených a archivovaných úkolů, třetí pro zobrazení aktivních úkolů na kanbanové nástěnce. Rozdělení úkolů do více stránek umožňuje mít lepší přehled. V kanbanové

nástěnce je možné přesouvat jednotlivé úkoly mezi sloupci díky drag and dropu. Tato funkcionalita byla implementována díky modulu „react-beautiful-dnd“. Skládá se ze tří částí. DragDropContext obaluje celou část drag and dropu funkcionality v aplikaci, Droppable je místo, kam je možné přesouvat Draggable elementy viz následující obrázek.



Obrázek 16 react-beautiful-dnd

Detail úkolu se zobrazuje v modálovém okně. Pro modálová okna byla použita komponenta z bootstrapu. Tato komponenta již řeší základní fungování modálových oken a dá se upravovat. V tomto okně je formulář, který obsahuje dostupné pole a akce pro daný úkol. Celá sekce je rozdělena do dvou sloupců. V prvním jsou obecné informace o úkolu spolu s checklisty a komentáři a v druhém sloupci jsou nastavovací pole jako je priorita, termín dokončení a řešitel. Toto rozvržení bylo inspirováno hlavně nástrojem Jira. Každá změna ve formuláři se ihned ukládá a projevuje se dále v aplikaci. Uživatel tak nemusí ukládat změny kliknutím na tlačítko uložit. O úspěšném, nebo neúspěšném uložení změn je uživatel informován díky notifikacím, které se zobrazují v rohu obrazovky. Pro tyto notifikace byla použita knihovna „react-toastify“. Celé fungování úpravy úkolu je naznačeno v ukázkách níže.

Formulář úkolu:

```
Uživatel provede změnu ve formuláři
if hodnota pole se změnila
  Ulož hodnotu do stavu formuláře
  Odešli akci reduceru s novou hodnotou
```

```
  Zobraz načítací komponentu
  Počkej na odpověď ze serveru, poté skryj načítací komponentu
  if odpověď je chybová
    Vrať změny
else
  Nic nedělej
```

Reducer úkolu:

```
Pošli požadavek na server s daty poslanými z formuláře
Čekej na odpověď
if odpověď není chybová
  Zobraz notifikaci o úspěšném uložení
  Ulož vrácená data ze serveru do stavu aplikace
else
  Zobraz chybovou notifikaci
```

Kontroler úkolu na serveru:

```
if uživatel má práva pro změnu
  Najdi upravovaný úkol
  Změň požadovaná data
  if změnila se nějaká hodnota úkolu
    Vytvoř záznam o změně
    if byl změněn řešitel nebo priorita
      Odešli notifikaci řešiteli/ům
    else if úkol byl dokončen
      Odešli notifikaci zadavateli
    Ulož změny do databáze
    Pošli informace o změně ostatním členům projektu
    Odešli odpověď na požadavek s upraveným úkolem
  else
    Odešli odpověď s nalezeným úkolem
else
  Odešli chybovou odpověď se zprávou
```

5.3 Implementace požadavků na systém

Z původních požadavků zadaných od zadavatele se podařilo implementovat všechny. Systém obsahuje kanban nástěnky pro každý projekt s možností jejich kastomizace, správu úkolů (viz předchozí kapitola), systém základních uživatelských práv, modul pro měření odpracovaného času a aplikace je celkově jednoduchá pro použití a přehledná. Dále byl systém implementován tak, aby podporoval metodiky Scrum, Kanban a jejich kombinace.

Při implementaci systému byla snaha vytvořit některé části podle existujících aplikací jako je Jira nebo Trello. Uživatelům, podle výsledků z dotazníku, tyto prostředí vyhovují a případný přechod na nové prostředí pro ně nebude tak náročný. Z funkcionalit získaných z dotazníku, které uživatelé vyzdvihují se podařilo implementovat přehlednou kanban nástěnku a jednoduché přiřazování úkolů a práci s mini. Filtr nad úkoly v nástěnce byl vytvořen po vzoru aplikace Trello a umožňuje filtrovat zobrazené úkoly podle několika kritérií. Jediná část, která zatím nebyla implementována je možnost vytvářet a používat šablony u úkolů a projektů.

Z navrhovaných funkcionalit, které byly nabízeny v dotazníku byl implementován osobní checklist. Každý uživatel má svůj vlastní a může si jednotlivé úkoly odškrtávat, přidávat a mazat.

5.4 Testování výsledné aplikace

5.4.1 Uživatelské testování

Slouží pro zjištění, zda uživatelé dokáží aplikaci bez problému používat. Testováním je možné zjistit, které části aplikace by bylo případně potřeba upravit nebo přidat. Celkem se testování zúčastnily čtyři osoby ze zaměstnanců firmy. Mnoho zdrojů uvádí, že čtyři testeři odhalí zhruba 75 % nedostatků, pět 80 % a u většího počtu testerů se přírůstek nových zjištění značně zmenšuje. Lze proto říct, že čtyři testeři na testování stačí. [42]

5.4.1.1 Scénář

Byl vytvořen seznam úkolů, které testeři měli splnit. Úkoly byly sestaveny tak, aby testeři prošli celou aplikaci a vyzkoušeli si nejrůznější funkce, které systém nabízí. Testovací scénáře jsou zobrazeny v příloze 9.2.

5.4.1.2 Průběh a vyhodnocení

Každý tester měl v aplikaci sepsán seznam úkolů, které plnil. Testování s každým testerem trvalo přibližně 30 minut. Testeři během testování popisovali, na jakém úkolu zrovna pracují a sdělovali, jak na ně systém působí a případně jestli mají problém s úkolem.

Žádný z testerů neměl problém s dokončením všech zadaných úkolů. Všichni hodnotili aplikaci velmi kladně, především se jim líbila přehlednost, rychlost, vzhled a aplikace jim přišla velmi intuitivní. Oproti aplikaci Jira testerům tato aplikace přijde přehledná a jednoduchá pro použití. Poznámky testerů z testování jsou zobrazeny níže v tabulce.

Klady	**Zápory**
Vzhled – 4x	Možnost změnit přiřazené skupiny v detailu uživatele – 2x
Intuitivnost – 4x	Uživatel nevidí seznam přiřazených skupin – 1x
Přehlednost – 3x	Není možnost export měřených záznamů do pdf – 1x
Měření času v aplikaci – 2x	
Vyskakující notifikace – 2x	
Checklist v úkolu – 1x	

Tabulka 3 Výsledky testování

Ze zobrazených výsledků lze vidět, že testeři hodnotili aplikaci spíše kladně. Největší připomínky měli k nastavování skupin uživateli a jejich znázornění. Celkově lze z průběhu a výsledků testování aplikaci hodnotit jako uživatelsky přívětivou a připravenou na reálný provoz. Testerům se také líbil podobný vzhled a struktura jako u ostatních aplikací, takže ihned věděli, kde co mají hledat. Proto by noví uživatelé neměli mít s přechodem na tuto aplikaci žádný problém.

6 Shrnutí

Cílem této diplomové práce bylo především navrhnout a implementovat nástroj pro řízení projektů. Nástroj měl splňovat požadavky zadavatele a při návrhu měly být pro lepší pochopení uživatelských požadavků, prozkoumány existující řešení. V první části práce byly popsány základní pojmy projektového řízení, přístupy a vybrané metodiky.
U metodik byl popsán způsob jejich fungování, principy a jak se od sebe liší. V další části byly vybrány tři existující nástroje, které byly prozkoumány a sloužili jako inspirace a k pochopení některých funkcionalit. U nástrojů byly popsány hlavní vlastnosti a sepsány metodiky, které podporují. U každého nástroje byly také sepsány cenové balíčky, které jsou nabízeny a vypočtena cena, kterou by každý nástroj firmu měsíčně stál.

Dále byl popsán způsob řízení projektů, který používá zadavatel a požadavky, které zadavatel požadoval po výsledném systému. K těmto požadavkům přibyly i ty, které vyvstaly z dotazníkového průzkumu. Po sepsání všech požadavků na aplikaci byla sepsána struktura jednotlivých částí výsledné aplikace. Tento návrh se snažil nejlépe vyhovět veškerým požadavkům. Při návrhu byly také vybrány metodiky, které by měla aplikace podporovat. V poslední části byly sepsány technologie, které byly použity pro vývoj aplikace, implementace vybraných částí a způsob testování výsledné aplikace.

7 Závěr a doporučení

Podařilo se vytvořit aplikaci dle zadaných požadavků a zpětné vazby od dotázaných uživatelů. Díky testování dalších aplikací bylo možné některé moduly implementovat podle nich, což umožňuje aplikaci používat podobně jako ostatní. Dokončená aplikace podporuje dvě agilní metodiky a je možné ji použít i pro metodiku zadavatele. Po nasazení na server se podařilo objevit několik chyb, které se nepodařilo najít během vývoje, anebo se projevily až v reálném provozu. Tyto chyby byly odstraněny a aplikace byla podrobena uživatelskému testování. Aplikace je nyní testována v rámci firmy a připravována pro produkčního prostředí. Do budoucna je možné aplikaci dále rozvíjet a poskytovat dalším firmám.

Vývoj klientské části aplikace šel bez velkých problémů. Především díky velkému množství dostupných knihoven, které se dají použít. I díky zvolené knihovně bylo možné implementovat mnoho funkcionalit velice snadno. Jediným velkým mínusem v použití frontendové knihovny je časově náročnější vývoj. Bylo by vhodné pro celou aplikaci použít nějaký typovací jazyk. U JavaScriptu by se jednalo o TypeScript. Tento jazyk by dokázal pevně držet strukturu dat, což je někdy v netypových jazycích problém. Velkou výhodou bylo použití jednoho programovacího jazyku pro obě části aplikace. Bylo tak možné vše psát v jednom kódu a více se zaměřit na samotnou implementaci. U serverové části by bylo vhodnější zvolit robustnější framework, který by poskytoval v základu více funkcionalit. Pokud by měl vývojář, nebo tým zkušenosti s jiným jazykem bylo by možné použít ten. Použití více jazyků může do budoucna komplikovat úpravu aplikace.

Během vývoje se několikrát měnila struktura některých částí, proto by bylo dobré věnovat návrhu aplikace více prostoru a tuto část nepodceňovat. U některých částí aplikace bylo by vhodné prozkoumat i možnosti integrace s okolními systémy namísto vytváření vlastního řešení.

8 Seznam použité literatury

[1] „Co je to projektové řízení?", *shean.cz*. https://www.shean.cz/clanky/detail/co-je-to-projektove-rizeni.htm (viděno 22. leden 2022).

[2] ManagementMania, „Projekt", *ManagementMania.com*. https://managementmania.com/cs/projekt (viděno 22. leden 2022).

[3] „Životní cyklus projektu", *BusinessInfo.cz*. https://www.businessinfo.cz/navody/management-zivotni-cyklus-projektu/ (viděno 22. leden 2022).

[4] L. Brown, „The Project Management Life Cycle, and Its 5 Phases", *Invensis Learning Blog*, 26. srpen 2019. https://www.invensislearning.com/blog/5-phases-project-management-lifecycle/ (viděno 22. leden 2022).

[5] ManagementMania, „Řízení projektů (Project Management)", *ManagementMania.com*. https://managementmania.com/cs/metody-rizeni-projektu (viděno 22. leden 2022).

[6] „Metoda, metodika, metodologie — Přírodovědecká fakulta UK". https://www.natur.cuni.cz/geografie/geoinformatika-kartografie/studium/bakalarske-studium/pravidla-pro-bakalarske-prace/metoda-metodika-metodolige (viděno 22. leden 2022).

[7] „Metodiky projektového managementu, jejich obsah a smysl – Wikisofia". https://wikisofia.cz/wiki/Metodiky_projektov%C3%A9ho_managementu,_jejich_obsah_a_smysl (viděno 22. leden 2022).

[8] „What is Agile Methodology? How It Works, Best Practices, Tools", *Stackify*, 17. září 2017. https://stackify.com/agile-methodology/ (viděno 22. leden 2022).

[9] „Advantage of Agile Methodology | Disadvantage of Agile Methodology - Javatpoint", *www.javatpoint.com*. https://www.javatpoint.com/advantage-and-disadvantage-of-agile-methodology (viděno 22. leden 2022).

[10] T. D. P. Manager, „9 Of The Most Popular Project Management Methodologies Made Simple", *The Digital Project Manager*, 10. srpen 2021. https://thedigitalprojectmanager.com/project-management-methodologies-made-simple/ (viděno 18. únor 2022).

[11] „7 Popular Project Management Methodologies And What They're Best Suited For", *Zenkit*, 9. březen 2018. https://zenkit.com/en/blog/7-popular-project-

management-methodologies-and-what-theyre-best-suited-for/ (viděno 10. březen 2022).

[12] „The 18 Top Project Management Methodologies to Use in 2022", *nTask*, 13. srpen 2020. https://www.ntaskmanager.com/blog/top-project-management-methodologies/ (viděno 10. březen 2022).

[13] „Agile vs. Scrum: What's the Difference?", *Northeastern University Graduate Programs*, 25. březen 2021. https://www.northeastern.edu/graduate/blog/agile-vs-scrum/ (viděno 22. leden 2022).

[14] K. Schwaber, *Agile project management with Scrum*. Redmond, Wash: Microsoft Press, 2004.

[15] „Who can cancel a Sprint? – Kaizenko". https://www.kaizenko.com/who-can-cancel-a-sprint/ (viděno 22. leden 2022).

[16] „What Is Kanban? Explained in 10 Minutes | Kanbanize", *Kanban Software for Agile Project Management*. https://kanbanize.com/kanban-resources/getting-started/what-is-kanban (viděno 22. leden 2022).

[17] „What Is Kanban? An Overview Of The Kanban Method", 18. srpen 2013. https://www.digite.com/kanban/what-is-kanban/ (viděno 22. leden 2022).

[18] „What Is Extreme Programming (XP)? - Values, Principles, And Practices", 9. červen 2021. https://www.digite.com/agile/extreme-programming-xp/ (viděno 22. leden 2022).

[19] „What is Extreme Programming (XP)?", *Agile Alliance |*, 14. červen 2017. https://www.agilealliance.org/glossary/xp/ (viděno 22. leden 2022).

[20] „≡ Lean Software Development: Principles and Benefits for Business", *Perfectial*. https://perfectial.com/blog/lean-software-development/ (viděno 22. leden 2022).

[21] „Lean Software Development (LSD)", *GeeksforGeeks*, 10. duben 2021. https://www.geeksforgeeks.org/lean-software-development-lsd/ (viděno 22. leden 2022).

[22] V. Singh, „What is Crystal Method in Agile and How it is different?", *TOOLSQA*, 7. červenec 2021. https://www.toolsqa.com/agile/crystal-method/ (viděno 22. leden 2022).

[23] „Crystal methods in Agile Development/Framework", *GeeksforGeeks*, 10. duben 2021. https://www.geeksforgeeks.org/crystal-methods-in-agile-development-framework/ (viděno 22. leden 2022).

[24] „What Is Feature Driven Development (FDD)? & How It Works?", 2. září 2021. https://www.digite.com/agile/feature-driven-development-fdd/ (viděno 22. leden 2022).

[25] „What Is PRINCE2 - PRINCE2 Qualification Explained | EUR". https://www.prince2.com/eur/what-is-prince2 (viděno 22. leden 2022).

[26] „PRINCE2: The Reigning Project Management Methodology". https://www.wrike.com/blog/project-management-basics-prince2-explained/ (viděno 22. leden 2022).

[27] „What Is PRINCE2? Principles, Aspects, Roles & Processes", *ProjectManager.com*, 28. listopad 2021. https://www.projectmanager.com/blog/prince2-methodology (viděno 22. leden 2022).

[28] „Understanding PRINCE2 Methodology in Project Management", *Kissflow*. https://kissflow.com/project/prince2-project-methodology/ (viděno 22. leden 2022).

[29] „The Best Project Management Software for 2022 | PCMag". https://www.pcmag.com/picks/the-best-project-management-software (viděno 22. leden 2022).

[30] „About JavaScript - JavaScript | MDN". https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript (viděno 22. leden 2022).

[31] „When to Use (and Not to Use) Asynchronous Programming: 20 Pros Reveal the Best Use Cases", *Stackify*, 25. září 2017. https://stackify.com/when-to-use-asynchronous-programming/ (viděno 22. leden 2022).

[32] „Stack Overflow Developer Survey 2021", *Stack Overflow*. https://insights.stackoverflow.com/survey/2021/?utm_source=social-share&utm_medium=social&utm_campaign=dev-survey-2021 (viděno 22. leden 2022).

[33] „JSX - What Is a JSX? & Introduction to Advanced". https://www.reactenlightenment.com/react-jsx/5.1.html (viděno 22. leden 2022).

[34] „What is a REST API?" https://www.redhat.com/en/topics/api/what-is-a-rest-api (viděno 22. leden 2022).

[35] „Introduction to Node.js", *Introduction to Node.js*. https://nodejs.dev/learn (viděno 22. leden 2022).

[36] „The V8 JavaScript Engine", *The V8 JavaScript Engine*. https://nodejs.dev/learn/the-v8-javascript-engine (viděno 22. leden 2022).

[37] „Blocking and Non-Blocking in Node.js", *GeeksforGeeks*, 19. květen 2021. https://www.geeksforgeeks.org/blocking-and-non-blocking-in-node-js/ (viděno 22. leden 2022).

[38] „Node.js Modules", *GeeksforGeeks*, 24. červen 2020. https://www.geeksforgeeks.org/node-js-modules/ (viděno 22. leden 2022).

[39] Node.js, „What is npm?", *Node.js*. https://nodejs.org/en/knowledge/getting-started/npm/what-is-npm/ (viděno 22. leden 2022).

[40] „What are WebSockets? | Web Security Academy". https://portswigger.net/web-security/websockets/what-are-websockets (viděno 10. únor 2022).

[41] „Object-relational Mappers (ORMs)". https://www.fullstackpython.com/object-relational-mappers-orms.html (viděno 22. leden 2022).

[42] „Jak na uživatelské testování | PROFICIO Marketing". https://proficio.cz/jak-na-uzivatelske-testovani (viděno 8. duben 2022).

9 Přílohy

9.1 Dotazník

Jaký nástroj používáte pro project management? *

- Freelo
- Jira
- Trello
- Basecamp
- Asana
- Jiná...

Myslíte si, že jste díky nástroji v práci více efektivní? *

- Ano
- Ne

Jak se nejčastěji dozvíte o novém úkolu nebo změně v úkolu? *

- E-mail
- Notifikace v aplikaci
- Jiná...

Používáte funkci měření času?

- Ano
- Ne

Na jakém zařízení nejčastěji s nástrojem pracujete? *

- Počítač/notebook
- Telefon
- Počítač i mobil používám stejně často

Jaké funkce/vlastnosti se Vám na nástroji líbí? *

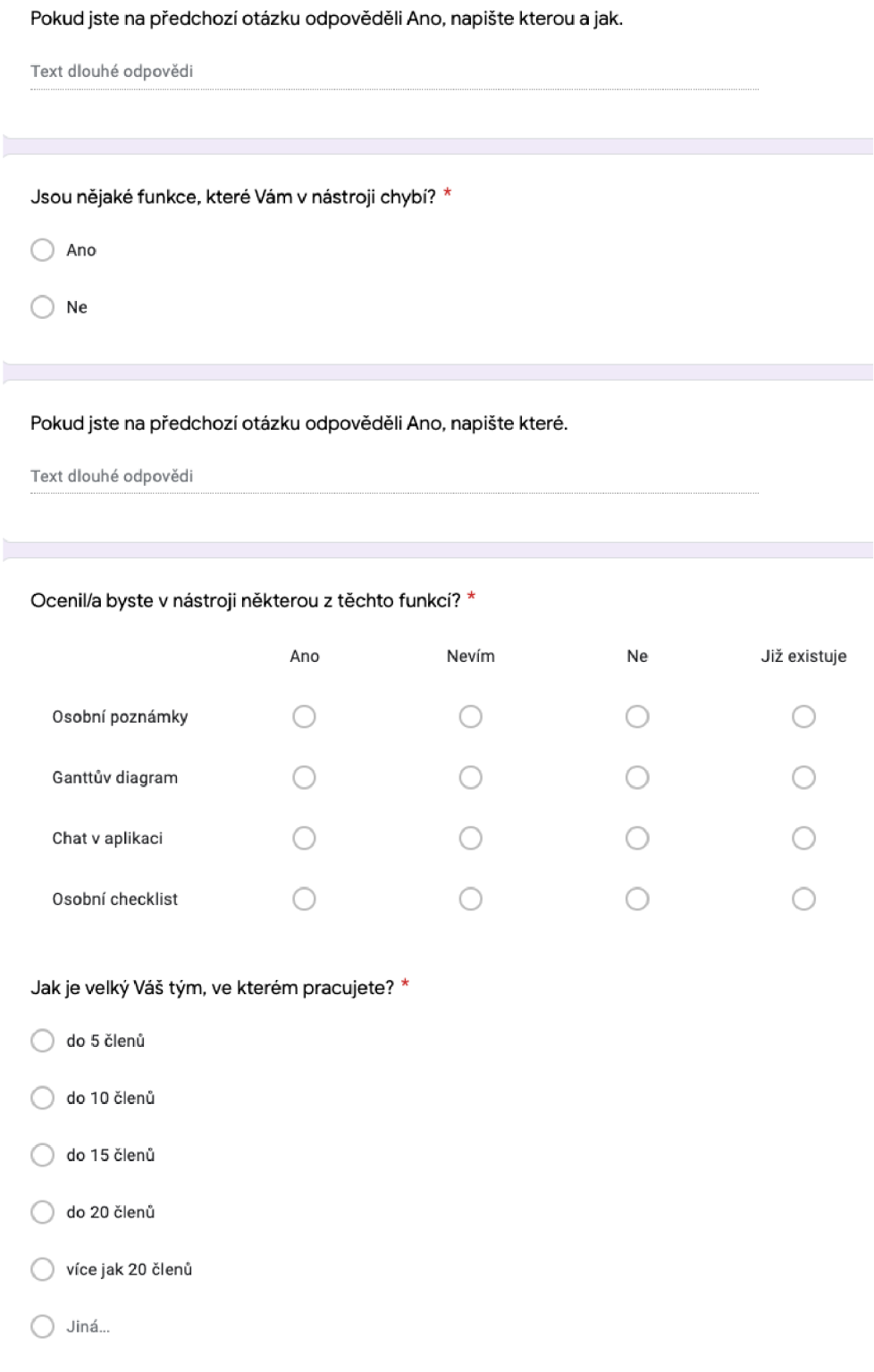
- [ ] Filtry nad úkoly
- [ ] Měření času
- [ ] Možnosti reportingu
- [ ] Šablony úkolů/projektů
- [ ] Integrace s aplikacemi
- [ ] Nastavení práv
- [ ] Přiřazování úkolů
- [ ] Mobilní aplikace
- [ ] Kanbanová tabule
- [ ] Cena
- [ ] Jiná...

Je něco, co se Vám na nástroji nelíbí nebo Vám nevyhovuje? *

- [ ] Filtrace úkolů
- [ ] Práce s úkoly
- [ ] Nastavení práv
- [ ] Vzhled
- [ ] Nástroj obsahuje málo funkcí
- [ ] Nástroj obsahuje mnoho funkcí
- [ ] Kanbanová tabule
- [ ] Cena
- [ ] Vše mi vyhovuje
- [ ] Jiná...

Vylepšil/a nebo změnil/a byste nějakou funkcionalitu? *

- ( ) Ano
- ( ) Ne



9.2 Scénáře testování

1. Přihlas se do aplikace jako administrátor a měř si čas po dobu testování.
2. Vytvoř nového uživatele se svými údaji, pozicí tester a rolí uživatel.
3. Odstraň všechny deaktivované uživatele a deaktivuj účet uživatele Jan Starý.
4. Svého vytvořeného uživatele přidej do nové skupiny s názvem „testeři“.

5. Vytvoř klienta se jménem Jago a dvěma tel. čísly a ostatní klienty odstraň.
6. Přidej nový projekt s názvem „Testování“ a klientem Jago, projekt má být přiřazen skupině testerů a dokončen do konce dubna.
7. Vytvoř novou nástěnku s termínem na konec měsíce.
8. Nástěnce přidej jeden pracovní sloupec s názvem „testování“ a omez počet úkolů na jeden.
9. Vytvoř dva úkoly a přiřaď je svému uživateli.
10. Nastav svému uživateli manažerská práva.
11. Přihlas se pod svým účtem a měř si čas.
12. Zjisti podrobnosti o přiřazených úkolech.
13. Prvnímu úkolu přidej dva podúkoly, změň popis, napiš komentář s přílohou o dokončení a úkol dokonči.
14. Jeden z podúkolů nakonec není potřeba, tak ho odstraň a druhý přesuň do archivu.
15. Druhému úkolu přidej checklist se dvěma položkami.
16. V backlogu vytvoř úkol s nejvyšší prioritou a s termínem dnes večer a přiřaď ho do nástěnky.
17. V nástěnce se pokus tento úkol vyfiltrovat.
18. Změň si heslo.
19. Ukonči časomíru a zjisti, jak dlouho testování trvalo a záznamy exportuj.

9.3 Seznam obrázků a tabulek

9.4 Zdrojové kódy a přístupy k aplikaci

Odkaz na zdrojové kódy a přístupy do aplikace jsou k dispozici v přiloženém souboru, který je odevzdán s touto textem.

UNIVERZITA HRADEC KRÁLOVÉ
Fakulta informatiky a managementu
Akademický rok: 2020/2021

Studijní program: Aplikovaná informatika
Forma studia: Prezenční
Obor/kombinace: Aplikovaná informatika (ai2-p)

Podklad pro zadání DIPLOMOVÉ práce studenta

Jméno a příjmení: **Bc. Patrik Štípek**
Osobní číslo: **I1900289**
Adresa: Jungmannova 1507, Hradec Králové – Pražské Předměstí, 50002 Hradec Králové 2, Česká republika

Téma práce: Návrh a implementace řešení projektového řízení pro malé a střední firmy
Téma práce anglicky: Design and implementation of project management solutions for small and medium-sized companies

Vedoucí práce: Mgr. Daniela Ponce, Ph.D.
Katedra informačních technologií

Zásady pro vypracování:

Cílem této práce je prozkoumat dostupné aplikace, dále podle zadaných požadavků navrhnout a vytvořit výslednou aplikaci. Součástí vypracování práce je také zpracování dotazníku, ve kterém budou osloveni uživatelé dostupných aplikací. Výsledky dotazníku pomůžou zjistit, jak se uživatelům s dostupnými aplikacemi pracuje, a budou použity pro lepší návrh aplikace, aby co nejlépe vyhovovala potřebám uživatelů. Aplikace bude používána v malé společnosti s počtem zaměstnanců do 5 osob; bude vhodná i pro distribuci mezi jiné, větší firmy.
Osnova

- Úvod, cíl, metodika
- Typy projektového řízení
- Existující řešení
- - Srovnání
- Vyhodnocení dotazníku
- Návrh systému
- - Zadané požadavky
- Implementace
- Závěr, shrnutí a doporučení

Seznam doporučené literatury:

CICIBAS, Halil; UNAL, Omer; DEMIR, Kadir Alpaslan. A Comparison of Project Management Software Tools (PMST). In: *Software Engineering Research and Practice*. 2010. p. 560-565. https://www.researchgate.net/publication/221610417_A_Comparison_of_Project_Management_Software_Tools_PMST
VARAJÃO, João; FERNANDES, Gabriela; SILVA, Hélio. Most used project management tools and techniques in information systems projects. *Journal of Systems and Information Technology*, 2020. https://www.researchgate.net/publication/345310044_Most_used_Project_Management_Tools_and_Techniques_in_Information_Systems_projects

UNGUREANU, Adrian & Ungureanu, Anca. (2014). Methodologies used in Project Management. Annals of Spiru Haret University Economic Series. 14. 47. 10.26458/1425.

Podpis studenta: Datum:

Podpis vedoucího práce: Datum: