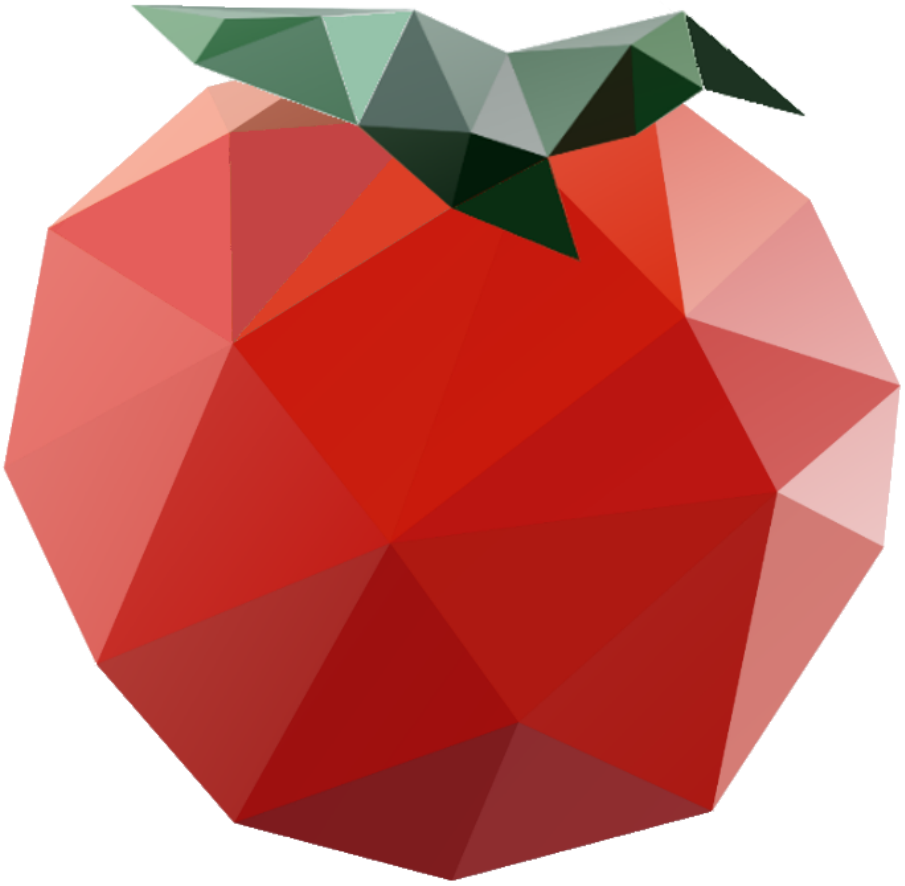
Pomodoro Timer Enterprise App

Babushkin - Doležal - Fiedler - Bulat



Obsah

Plánování

V rámci analýzy projektu Pomodoro Enterprise byly vytvořeny matice zodpovědností a také několik druhů diagramů, pomocí kterých byla odhadnuta průměrná časová náročnost projektu, rozepsány všechny aktivity, které by měly být provedeny během realizace projektu Pomodoro Enterprise, také bylo vytvořeno plánování dílčích úkolu každého členu týmu, kde byly popsány přesně definované činnosti a odhad jejích náročnosti.

Matice zodpovědností

Matice zodpovědností je tabulka, do řádku které se píšou jednotllivé aktivity, do sloupců se píšou jména členů týmu. Na průseku sloupců a řádků se označuje symbolem X událost, když člen týmu je zodpovědný za vykonání této činnosti označené v řádku.

Done Needs improvements Not ready	Andrey	Lukáš	Petr	Kyrylo
Logický rámec (obecné cíle)	X			
Logický rámec (hlavní etapy)			X	
Logický rámec (podrobnější popis etap)				X
WBS		X		
Matice odpovědností	X			
Síťový graf	X			
Rozpočet		X		
Analýza - diagram procesů		X		
Analýza - doménový model				X
Analýza - Diagram stavů	X			
Analýza - Požadavky ([ne]funkční)			X	
Analýza - Use cases		X		
Analýza - Wireframes	X			
GANTT Front-End			X	
GANTT Zákazník		X		
GANTT Implementace				X
ProjectLibre	X			
Architektura - Prezentační vrstva (VC)			X	
Architektura - Business/Doménová vrstva	X			
Architektura - Datová vrstva (M)				X
Implementace - Views (All platforms)			X	
Implementace - Databáze				X
Implementace + Tests - iOS	X			
Implementace + Tests - Server		X		X
Implementace + Tests - Web			X	
Implementace + Tests - Android				X
System testing	X	X	X	X

Logický rámec

Logický rámec je metodika projektového řízení, která řeší strategické plánování projektu, přípravu, jeho návrh a realizaci. Tento přistup je jednoduchý způsob jak popsat projekt pomoci jedné tabulky.

Logický rámec je ve formě matice, která obsahuje 5 základních řádků a 3 sloupce. Při vytváření logického rámce se popisují tyto položky:
V řádcích:

- Hlavní cíl projektu neboli obecný záměr
    - Jedná se o obecný popis přínosu pro zadavatele od projektu. Často se popisuje cíl, strategický záměr organizace a jakým způsobem tento projekt přispívá k těmto cílům.
    - Většinou by popis této kapitoly měl odpovídat na otázku „PROČ?“.
- Projektové cíle či bezprostřední cíl
    - Tato kapitola logického rámce se zabývá specifikací cílů projektu a konkrétním popisem důvodu realizace projektu.
    - Odpovídá na otázku „PROČ?“.
- Výstupy
    - Tato část popisuje, co by mělo být výstupem projektu a nejčastěji popisuje hmotné výsledky, které mají být dodány projektem. Po přečtení této kapitoly by mělo být jasné, s čím může zákazník počítat při kontrole výsledku projektu.
    - Odpovídá na otázku „CO?“.
- Aktivity
    - Tato sekce popisuje aktivity, které je nutno vykonat, aby mohli být dodány výstupy, a jakým způsobem je vykonat. To znamená, že pro každý výstup by měla být navrhnuta alespoň jedna aktivita, která se musí vykonat, aby mohl být výstup dodán.
    - Odpovídá na otázku „JAK?“.
- Negativní vymezení projektu
    - Tato kapitola není nutnou častí logického rámce, ale v některých případech může být vhodná. Vymezuje cíle, výstupy a aktivity, kterými se projekt nemá zabývat.

Ve sloupcích:

- Objektivně ověřitelné ukazatele
    - Tento sloupec má obsahovat měřitelné indikátory, podle kterých by se dalo určit, do jaké míry budou specifikace cíle dosažené. Položky tohoto sloupce mají splňovat formát kvantity, kvality a času.
- Zdroje informací k ověření
    - Tento sloupec se vztahuje k předchozímu. To znamená, že ukazatele by měly být ověřené a toto ověření musí proběhnout nějakým způsobem. V tomto sloupci se určí, jakým způsobem se ověří a případně, kde nalézt potřebnou informací k ověření.
- Předpoklady a rizika
    - Tento sloupec se nevztahuje k řádku o hlavním cíli projektu. Popisuje, které předpoklady musí být splněny pro dosažení cíle a taky rizika, se kterými projekt může setkat během vytvoření.

Cílem naší semestrální práci je vytvoření mobilní aplikace pro kontrolu úkolů na projektu, jejich rozdělení mezi zaměstnance, plánování práce, rozdělení jednotlivých úkolů na menší části a kontrola pokroku vyplněných zadání.

Technika Pomodoro je druh organizace času. Tato technika využívá časovače k rozdělení úkolu na části tradičně 25 minut dlouhé a oddělené krátkými přestávkami. Princip metody spočívá v myšlence, že časté krátké přestávky zvyšují produktivitu a během 25 minut se musí člověk soustředit jen na práci.

Tato aplikace není zaměřená na specifické zákazníky. Tento přistup se dá uplatnit na projektech různého zaměření a s různými cíli. Pro účely semestrální práce předpokládáme potenciálního uživatele této aplikace.

Jedním z hlavních cílů firmy, která se rozhodne využít naší aplikaci, je zvýšení produktivity zaměstnanců. To je jedním z klíčových faktorů úspěšností firmy. Tento cíl se dá lehce ověřit počtem udělaných úkolů, časem potřebným pro vyplnění jednotlivých úkolů a tím pádem i náklady na projekt.

Pro dosažení těchto cílů ve firmě je vhodné použit pomodoro techniku pro time managment. Jedním z možných způsobů použití této techniky je vytvoření aplikace, která by umožňovala všechny tyto funkcionality. Projektovým cílem je vytvoření aplikace Pomodoro s těmito funkcionalitami:

- Rozdělení práce mezi zaměstnance
- Rozdělení práce na časové úseky
- Rozdělení času na přestávku mezi úkoly
- Možnost odevzdání hotového úkolu
- Možnost sledování statistik vykonaných úkolů
- Sledování pokroku specifického úkolu

Ověřitelnými ukazateli je hotová aplikace Pomodoro, počet stahování této aplikace zaměstnanci firmy a taký využití aplikace pro time managment, který by se dalo ověřit zmenšením času na jednotlivé úkoly. Tyto údaje se dá ověřit přes existenci položek na Google Play pro Android a na App Store pro iOS, spuštění hotové aplikace na mobilních přístrojích a sledování výsledků práce přes aplikaci.

Výstupy projektu se dá rozdělit do několika kategorií, podle kterých by zákazník mohl ověřovat pokrok během vytvoření aplikace a na konci projektu. Výstupům aplikace přímo korespondují aktivity, které byly vykonány během realizace projektu.

Analýza

Analytická část projektu, která obsahuje analýzu business modelu aplikace, sběr požadavku od zákazníka, návrh aplikace, případů užití a doménového modelu aplikace. V rámci této části by měly být všechny aktivity řádně zdokumentované pomocí moderních nástrojů jako EA(Enterprise Architect). Tím pádem zákazník může snadno ověřit výsledky práce z této části pomoci počtu vytvořených modelů a diagramů.

Implementace

Implementační část projektu je samotný vývoj aplikace. Obsahuje část, která pokrývá vytvoření datové, business, prezentační vrstvy aplikace. Výsledky práce z této části dá se sledovat podle počtu vytvořených entit v datové vrstvě, počtu vytvořených tříd, počtu vystavených webových API pro přistup k datům, počtu vytvořených obrazovek aplikace. Pro zákazníka by se dalo všechno zjednodušit na počet napsaných řádek kódů.

Testování

Testování se zabývá otestováním funkčností aplikace a odstraněním případných defektu z aplikace do momentu, než my předáme aplikaci zákazníku. Udělanou práci dá se ověřit počtem vystavených defektů, počtem hodin strávených na testování a počtem úspěšných a neúspěšných testů. Tyto údaje se dá ověřit ze statistik nástrojů použitých k testování (HP ALM atd.).

Nasazení

Po úspěšných předchozích krocích následuje nasazení aplikace na produkční prostředí a její spuštění pro uživatele. Tyto údaje se dá ověřit pomoci počtu spuštěných serverů, uptime serverů, počtem hlášek v aplikačním logu běžící aplikace. Po úspěšném nasazení aplikace zákazník bude moct stáhnout aplikaci a vyzkoušet její na vlastním projektu, čím ověří výsledky práce nejenom této části ale i výsledek celého projektu.

	Strom / hierarchie cílů	Objektivně ověřitelné ukazatele	Zdroje informací k ověření / Způsob ověření	Předpoklady a rizika
Hl. cíl (přínos projektu)	Zvýšení produtivity zaměstnanců ve firmách	Zvýšení počtu udělaných úkolu o 20%	Výroční zpráva firmy	
		Snížení času straveného na Internetu o 20%	Logy	
		Snížení nákladů na každý projekt o 20%	Výroční zpráva firmy	
Projektové cíle (účel, specifický cíl)	1. Vytvoření aplikace pro Time Management s použitím techniky Pomodoro	Existující aplikace pro platformy iOS, Android a také verze běžící v prohlížeči	Položka na Google Play Položka na App Store Webové rozhraní	Zkušený tým programátorů Obrosvký počet podobných aplikací Otevřenost trhu a poptávka po způsobech optimalizace práce Aplikací bez orientace na pracovní trh => malá konkurence
		Využití techniky Pomodoro s nastavitelnou délkou pomodora, délkou malé pauzy, délkou velké pauzy a nastavitelným počtem pomodor, které musí uplynout mezi velkými pauzami	Spuštění časovače v aplikaci	
		Využití pro Time Management. Během pomodoru nelze prokrastinovat. Čas prokrastinace se sníží z 122 min do 60.	Spuštění časovače v aplikaci	
Výstupy (výsledky)	1.1 Analýza	Vypracovaný model případů užití (10 případů)	Analytická dokumentace	Zkušenosti se softwarovým inženýrstvím Chybějící designér Potřeba využití neznámých technologií Nespolehliví testeři Technické problémy s připojením Nespolehlivý Internet Service Provider Problémy s přístupy k obchodům s aplikacemi
		Vypracovaný model požadavků (5 funkčních a 5 nefunkčních)	Analytická dokumentace	
		Navržený doménový model (přesný relační model)	Analytická dokumentace	
		Zhotovené návrhy obrazovek (min. 5 Wireframes)	Analytická dokumentace	
	1.2 Implementace	Vypracovaný návrh architektury s přesnou hierarchií tříd a možností rosšiřování v budoucnu	Návrhová dokumentace	
		Vypracovaný relační/objektový datový model v EA	Návrhová dokumentace	
		Zhotovený návrhový model tříd s přesně stanovenou hierarchií	Návrhová dokumentace	
		Zhotovený model komunikace (Realizace případů užití)	Návrhová dokumentace	
		Vytvořená a konfigurovaná databáze vytvořená v SQLDeveloper	Databázový server	

		Zhotovená business vrstva s API interfaces v Jave	API a jeho vliv na data v databázi	
		Zhotovená prezentační vrstva ve formě aplikace pro Android	Položka na Google Play	
		Zhotovená prezentační vrstva ve formě aplikace pro iOS	Položka na App Store	
		Zhotovená prezentační vrstva ve formě webové aplikace	Webové rozhraní	
	1.3 Testování	Unit testování, test pro každou významnou funkci	Soubor pro logování výsledků unit testů	
		Testování integrace, testy pro každou skupinu tříd, běžících spolu	Soubor pro logování výsledků integračních testů	
		Systémové testy na zařízeních iPhone a některých Android devicech, testy v Chrome, Firefox, Safari a IE	Zpětná vazba od testerů	
		Testování provozního přijetí, testy všech případů užití a celkové funkčnosti aplikace	Zpětná vazba od testerů	
	1.4 Nasazení	Spuštění databázového serveru, availability 16/5, consistency 99 procent, redundancy coef. 3	Logy databáze	
		Spuštění serveru s business vrstvou a webovým rozhraním na statické IP adrese	Dostupné API Logy serveru Dostupné webové rozhraní	
		Export aplikace na Google Play Europe a úspěšný start	Položka na Google Play, hodnocení zákazníků	
		Export aplikace na App Store Europe a úspěšný start	Položka na App Store, hodnocení zákazníků	

Aktivity	1.1.1 Sběr a specifikace požadavku na aplikaci. 1.1.2 Vytvoření případů užití. 1.1.3 Komunikace analytiků se zákazníkem. 1.1.4 Vytvoření architektury aplikace. 1.1.5 Vytvoření doménového modelu. 1.1.6 Vytvoření technického designu aplikace. 1.1.7 Vytvoření grafického návrhu aplikace(Wareframe). 1.2.1 Vytvoření DB modelu. 1.2.2 Vytvoření modelu tříd. 1.2.3 Mapování mezi DB modelem a modelem tříd. 1.2.4 Konfigurace DB. 1.2.5 Vytvoření DAO pro CRUD operace s DB. 1.2.6 Service layer pro konverzaci a zpracování DB objektů na BL objekty. 1.2.7 Vytvoření komponent pro BL aplikaci. 1.2.8 Vytvoření WEB API pro přistup k BL komponentám. 1.2.9 Vytvoření FE controllerů pro obrazovky. 1.2.10 Vytvoření FE obrazovek pro IOS/Android. 1.3.1 Napsání UNIT testů pro DAO vrstvu. 1.3.2 Napsaní UNIT testů pro operace BL vrstvy. 1.3.3 Napsání UNIT testů pro kontrolu FE controllerů. 1.3.4 Vytvoření test-casů pro testování WEB API aplikace. 1.3.5 Vytvoření test-casů pro testování validaci FE obrazovek IOS/Android. 1.3.6 Vytvoření test-casů pro testování všech hlavních případů užití aplikace. 1.3.7 Testování "stress" na DB stroje. 1.3.8 Testování "stress" na WEB API servery. 1.4.1 Zakoupení DB strojů a spuštění. 1.4.2 Zakoupení serverů pro BL vrstvu. 1.4.3 Konfigurace domény pro WEB API. 1.4.4 Spuštění serverů pro BL a WEB API. 1.4.5 Nahrání aplikace na Google Play. 1.4.6 Nahrání aplikace na App Store.	Prostředky : 1. Analytický tým min 2 osoby 1.1 Komunikace se zákazníkem 1.2 Návrh aplikace a technický design. 2. Developer/Unit-test tym 5 osob : 2.1. IOS FE full stack developer. 2.2. Android FE full stack developer. 2.3. 2 x Java BE developer. 2.4. DB specialista. 3. Infra tmy 1 osoba 3.1 Zakoupení a konfigurace strojů, různých Android zařízení, nasázení aplikací a build aplikace. 4. Projekt manager 1 osoba Předběžný rozpočet : 300 tisíc korun.	Časový rámec : 1. půlka ledna - komunikace se zákazníkem a sběr požadavku 2. půlka ledna - dokumentace požadavku a vytvoření případů užití. 1. půlka února - dokončení analytické částí a návrh Wareframu/ mezi tím Infra tým řeší HW část aplikací, zakoupení strojů a licencí na SW. 2. půlka února - začátek implementace FE obrazovek a DB vrstvy / mezi tím test-tým nastuduje analytickou část a začne pracovat na vytvoření test-casů. 1. půlka března - vývoj business vrstvy, WEB API, Integrace s DB, vytvoření FE controlerů / mezi tím analytici přijímají change requesty a nové návrhy na zlepšení od zákazníků/ test-tým provádí stress-testy na DB stroje a testuje DAO vrstvu. 2. půlka března - Integrace FE controlerů a pokusy o provolání WEB API/ test-tým pracuje na validaci FE obrazovek a stress-testy na WEB API(testování webových serverů) / DEV tým pracuje na defektech od test-týmu a na případných změnách od analytiků / Analytici sbírají požadavky na nový release 1. půlka dubnu - integrace všech částí aplikace, testování případů užití, akceptace aplikace zákazníkem. 2. půlka dubnu - build aplikaci na produkčních serverech, nahrání aplikace na Google Play a App Store.	Chybějící podklady od zákazníků. Nedostatek zkušeností s některými frameworky v DEV týmu. Drahé licence na SW a HW. Nedostatek a nesledování časového planu. Podpora ze strány BA (Bussiness Analysis) týmu od zákazniku Výborný tým analytyků a designerů, což usnadní a urychlí vyvoj
				Předběžné podmínky

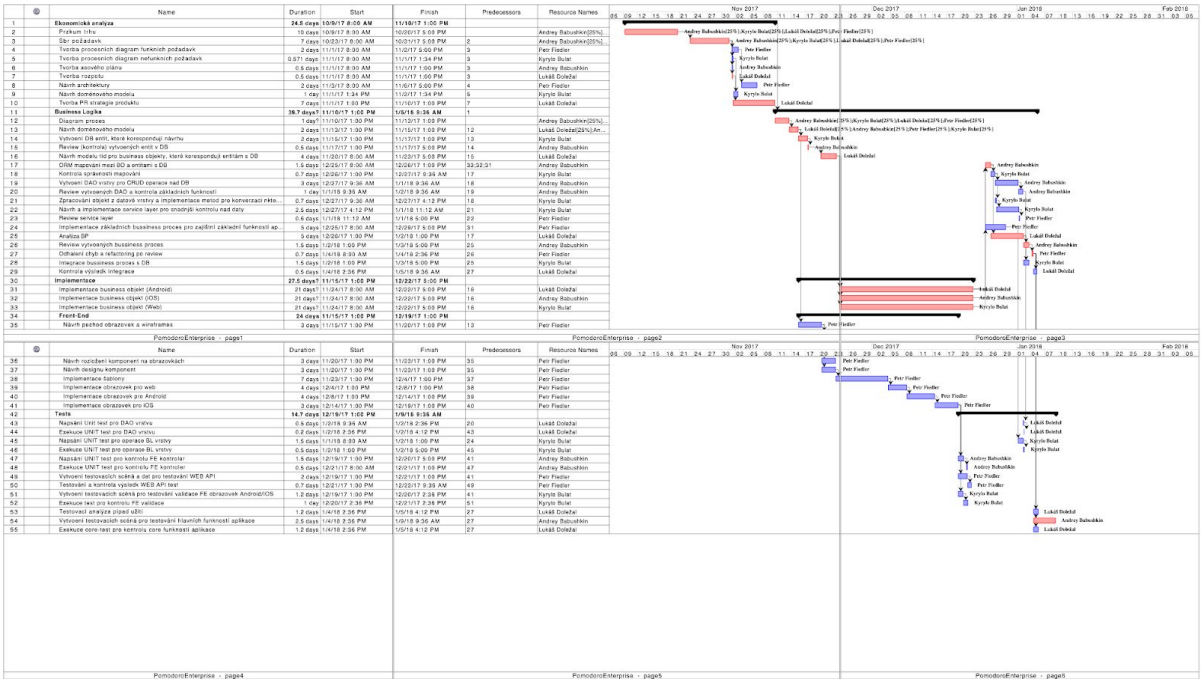
Ganttův diagram

Ganttův diagram je druh pruhového diagramu, který se využívá při projektovém řízení pro grafické znázornění posloupnosti naplánovaných činností v čase.data a času začátku od shora dolu.

Na horizontální ose Ganttova diagramu je časové trvání projektu. Čas je rozdělen do stejně dlouhých jednotek (např.: dny, týdny apod.), v našem projektu používáme jako jednotku takzvaný man-day. Pojem man-day (množné man-days) je z angličtiny jednotka, která určuje, kolik dnů by daná činnost trvala, kdyby na ni pracoval právě jeden člověk. Na vertikální ose jsou rozprostřeny jednotlivé činnosti ze kterých se projekt skládá. Jeden řádek vždy představuje právě jednu činnost. Činnosti jsou obvykle seřazeny podle data a času začátku od shora dolu.

Na ploše diagramu jsou jednotlivé činnosti reprezentovány pruhy, jejichž levý okraj označuje plánovaný začátek činnosti a pravý okraj plánované ukončení. Délka pruhu tak označuje předpokládanou délku trvání činnosti.

Červenou barvou je označena kritická cesta - nejdelší cesta realizace projektu, která zabere nejvíce času.

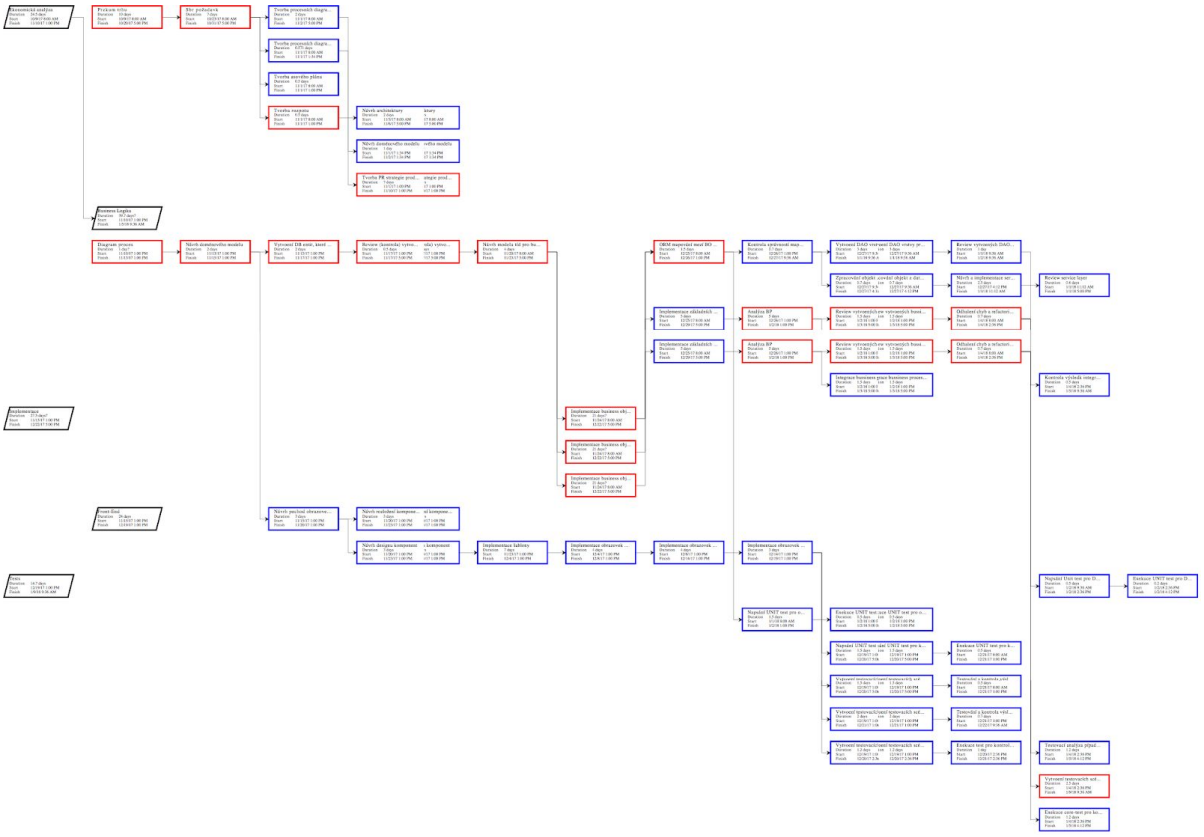


Celý projekt byl rozdělen na několik částí, každá z nich obsahuje menší aktivity. Každé aktivitě byl přiřazen jeden nebo více vykonavatelů a průměrný odhad času, který tato aktivita může zabrat. Celkem velkých kategorií aktivit je 4:

1. Ekonomická analýza. Aktivity této kategorie pokrývají analýzu trhu, sběr požadavků, tvorbu rozpočtu, předpokládaný návrh architektury budoucího projektu pro zjištění potřebných peněžních zdrojů a tvorbu rozpočtu. V rámci ekonomické analýzy by také měla být vytvořena PR strategie Pomodoro Enterprise.
2. Business Logika. V této sekci jsou popsány všechny aktivity spojené s analýzou a návrhem architektury aplikací, vytvoření databázového modelu a objektů DAO.
3. Implementace. Tady jsou pokryty všechny aktivity spojené s implementací aplikací.
4. Testování. V této sekci jsou rozepsány všechny aktivity, týkající se testování hlavních modulů aplikace a jejích integrace mezi sebou.

Síťový graf a kritická cesta

Síťový graf popisuje všechny aktivity, jejích spojení mezi sebou a pořadí vykonání. Z toho grafu je jasně vidět označenou červeně kritickou cestu - cestu aktivit, která zabere nejvíc času.



Analýza a rozpočet

Ekonomická analýza

Prvním krokem analýzy byl průzkum současného trhu, tedy hledání podobných aplikací. Zjistili jsme tak, že všechny aplikace nabízející obdobné funkce se dají rozdělit do dvou kategorií. Prvním typem jsou složité a komplexní programy na time-management, které v sobě techniku Pomodoro mají také, ale kvůli enormnímu počtu ostatních funkcí se stává používání Pomodoro timeru velmi nepraktickou činností. Programy bývají též velmi často pouze v placené verzi. Druhou skupinou jsou jednoduché freewarové aplikace, které skýtají nevýhody z přesně opačného konce. Umí pouze základní Pomodoro techniku, nejsou spolehlivě napsané a nenabízejí žádné funkce usnadňující jejich ostatní používání. Proto jsme se rozhodli vstoupit se svým produktem na trh my.

Po návrhu této základní myšlenky jsme uspořádali malý průzkum mezi zaměstnanci stejného oboru, tedy ideálně budoucími zákazníky, a ptali se na jejich požadavky na aplikaci na podobném principu. Po jejich prozkoumání jsme se rozhodli napsat novou, jednoduchou, praktickou, avšak stále finančně dostupnou aplikaci pro Pomodoro Timer. Hlavní výhodou naší aplikace by měla být jednoduchá organizace práce v týmu a online ukládání práce pro pokračování na projektech bez starostí o dosavadní výkon. Aplikace bude svými funkcemi užitečná jak pro amatérské či studentské týmy, tak pro profesionální vývojářské týmy.

Pro lepší orientaci v jednotlivých funkcích jsme sestavili procesní diagramy pro všechny plánované procesy – funkční i nefunkční požadavky. Podle složitosti úkolů jsme vhodně vytvořili časový plán pro budoucí práci a z vypočtených hodin vhodně spočetli potřebný rozpočet.

Rozpočet a cena licence

Rozpočet

Náklady na aplikaci jsme vypočítávali přímo z časového plánu a přidali nutnou rezervu pro případná zpoždění či problémy jiného typu, které by cenu projektu zvýšily. Po propočítání celkového hodinového plánu a přidání rezervy činí vypočítané náklady na dvě stě tisíc korun.

Cena licence

Aplikace je primárně určena pro firmy a obdobné organizace libovolné velikosti. K jejímu používání je zapotřebí zakoupit u vývojářské firmy příslušnou licenci. Typy licencí jsou rozděleny podle velikosti týmu a časové období je vždy jeden měsíc. Za měsíční licenci pro tým do pěti uživatelů zaplatí firma 99 Kč, za tým do dvaceti členů 299 Kč a za verzi pro neomezený počet uživatelů 499 Kč.

Požadavky

Funkční požadavky

Správa účtů

Každý uživatel se musí při spuštění aplikace přihlásit. Účty slouží k autorizaci do databáze a ovládání synchronizace. Uživatel může být přihlášen na libovolném počtu zařízeních.

Tvorba projektů

Uživateli bude umožněno vytvořit libovolný počet vlastních projektů s vlastním nastavením. Nadřízený uživatel bude moci vytvářet skupinové projekty. Přes jednotlivé skupinové projekty může každý uživatel též dostat přidělenou práci od svého nadřízeného.

Nastavení projektů

Uživatel si bude moct u každého projektu nastavit požadované délky pomodor, malých i velkých pauz a určit potřebný počet odpracovaných pomodor mezi velkými pauzami. Skupinové projekty může upravovat pouze vedoucí projektu.

Rozdělení práce podřízeným

Používání aplikace ve funkci projektového manažera přináší funkci přerozdělování práce. Manažer může vytvořit projekt s libovolným nastavením a přidělit práci na něm jednomu ze svých podřízených. Samozřejmostí je pozdější sledování výkonu práce.

Nefunkční požadavky

Nefunkční požadavky jsou požadavky na projekt, které se neřadí mezi funkční, čili nijak nerozšiřují stávající požadavky na funkcionalitu projektu. Takové požadavky specifikují například pro jaké platformy je softwarový projekt určen nebo jaké standardy by měl dodržovat. Mezi nefunkční požadavky se řadí i specifické požadavky na grafické rozhraní a design či specifické požadavky na vlastnosti projektu, které nepřidávají další funkcionalitu.

Více platforem

Aplikace bude dostupná jako položka na Google Play pro Android verze 5 nebo vyšší a jako položka na App Store na iOS verze 8 nebo vyšší. Aplikace bude mít také webové rozhraní optimalizované pro prohlížeče Google Chrome verze 54 nebo vyšší, Mozilla Firefox verze 53 nebo vyšší, Opera verze 41 nebo vyšší, Microsoft Edge verze 15 nebo vyšší a Safari verze 10 nebo vyšší.

Grafické rozhraní

Aplikace bude na všech platformách nabízet grafické rozhraní. Design webového rozhraní a design aplikace pro Android se budou řídit pravidly pro Material Design navržený společností Google Inc.

Notifikace

Aplikace bude vyvolávat upozornění po uplynutí času pomodora, skončení malé i velké pauzy či při zahájení skupinového pomodora či pauzy.

Online synchronizace

Aplikace bude veškerá data synchronizovat online, tedy například spustíme-li časovač na jednom zařízení, spustí se na všech, kde aplikace běží, a pokud spustíme aplikaci až po spuštění časovače, aplikace se otevře již se spuštěným časovačem se správným časem.

Návrh a implementace

Front-end

Návrh přechodů obrazovek (2 man-days)

Během návrhu přechodů obrazovek by se mělo vymyslet, jaké obrazovky se budou v aplikaci nacházet a z kterých obrazovek by mělo být možné přejít na jaké jiné obrazovky. Mělo by se dbát na to, aby přechody mezi obrazovkami dávaly smysl, například přechod z obrazovky pro nastavení na obrazovky nápovědy, statistik nebo zpětné vazby smysl nedávají.

Návrh rozložení komponent na obrazovkách (3 man-days)

Během návrhu rozložení komponent na obrazovkách by se mělo vymyslet, kde se na jaké obrazovce bude nacházet která komponenta. Mělo by se dbát na to, aby se tlačítka sloužící k přechodu mezi obrazovkami nacházela na místech, která dávají smysl. Žádná komponenta by se neměla nacházet na špatně viditelných částech obrazovky. Návrh rozložení komponent na obrazovkách muže ovlivnit návrh přechodů mezi nimi, a tedy je vhodné začít s tímto návrhem již v průběhu návrhu přechodů obrazovek.

Návrh designu komponent (4 man-days)

Během návrhu designu komponent by se mělo vymyslet, jak jednotlivé komponenty budou vypadat. Mělo by se dbát na to, aby tlačítka sloužící k přechodu mezi obrazovkami byla dostatečně viditelná a bylo jasné k čemu slouží. Obecně by se mělo dbát na to, aby ze vzhledu každé komponenty bylo jasné, k čemu slouží. V aplikaci by tedy neměla být například žádná “tajná tlačítka”, tedy komponenty, které po stisku vykonávají nějakou akci, ale nevypadají jako tlačítka. Návrh designu komponent může ovlivnit návrh jejich rozložení, a tedy je vhodné začít s tímto návrhem již v průběhu návrhu rozložení komponent na obrazovkách.

Tvorba wireframů (3 man-days)

Během tvorby wireframů je ideálně vhodné mít již hotový návrh přechodů obrazovek, návrh rozložení komponent na obrazovkách i návrh designu komponent, aby mohli zhotovené wireframy fungovat jako zjednodušený prototyp front-endu.

Implementace šablony (7 man-day)

Implementace šablony spočívá ve vytvoření šablony v jazyce CSS, LESS nebo SASS, tedy základních grafických tříd. Jazyky LESS a SASS se kompilují do CSS. Výhoda CSS šablony je její použitelnost na všech zamýšlených platformách. Je vhodné testovat grafické šablony na obrazovkách, pro které se šablona implementuje, je tedy vhodné zároveň s implementací šablony začít i s implementací obrazovek pro jednotlivé platformy.

Implementace obrazovek pro web (4 man-days)

Implementace obrazovek pro web spočívá ve vytvoření dokumentů v jazyce HTML, či částí HTML dokumentů, které se pak skládají dohromady na serveru. Implementace obrazovek samozřejmě zahrnuje i implementaci těchto skládacích rozhraní.

Implementace obrazovek pro Android (4 man-days)

Implementace obrazovek pro Android spočívá ve vytvoření dokumentů v jazyce XML, které popisují rozložení jednotlivých komponent, a tříd v jazyce Java, které jsou potřeba pro vykreslení nebo k definici komponent.

Implementace obrazovek pro iOS (4 man-days)

Implementace obrazovek pro iOS spočívá ve vytvoření dokumentů storyboard specifikujících vzhled aplikace.

Programování chování komponent pro web (6 man-days)

Programování chování komponent pro web spočívá ve vytvoření souboru skriptů, které definují a nastavují komponentám jejich chování při různých událostech.

Programování chování komponent pro Android a iOS (6 man-days)

Programování chování komponent pro Android a iOS spočívá ve vytvoření controllerů, které zpracovávají události vyvolané na komponentách. Controllery pro Android jsou napsané jazyce Java, případně JavaScript, a controllery pro iOS jsou napsané v jazyce Swift.

Back-end

Návrh doménového modelu (5 man-days)

Během návrhu doménového modelu by se mělo určit, které objekty se používají v aplikaci a jak mezi sebou propojeny. Při vytvoření doménového modelu nejprve návrhář se zamyslí nad entitami z business analýzy aplikace, navrhne atributy pro tyto entity, které by měly být v aplikaci sledovány pro správnou funkčnost. Po navržení základních entit je podstatný určit, jak tyto entity mezi sebou komunikují a jaké vztahy mají, popsat multiplicitu vztahů mezi entitami. Při vytváření doménového modelu návrhář vychází z objektů reálného světa a z business analýzy. Doménový model je základním modelem pro další vývoj aplikace, proto je důležité věnovat se co nejvíc času správnému návrhu.

Návrh databázového modelu (4 man-days)

Při návrhu databázového modelu se návrhář orientuje na již hotový doménový model a podle toho určuje, které entity budou implementovány v databázi, jaké atributy budou mít, jakou přesnost budou mít různé atributy u různých entit. Často databázový model má hodně rozdílů od doménového modelu, protože na této úrovní návrhář musí zachytit všechny vztahy mezi entitami a pro tyto potřeby vytvořit pomocné tabulky pro lehčí vývoj a údržbu aplikace. Databázový model hraje roli přesné specifikace dat aplikace, podle které se dá implementovat datovou část aplikace a stavit nad ní business logiku aplikace. Každá oprava databázového modelu aplikace má velký dopad na funkčnost celé aplikace, proto je podstatné věnovat se tomu dost času ve fázi před vývojem.

Implementace DB (2,5 man-days)

Po tom, co je navrhnuty databázový model aplikace, vývojář muže začít s implementaci databáze aplikace. Vytvoření databáze aplikace zahrnuje vytvoření create skriptů pro vytvoření tabulek, indexace tabulek, nastavení primárních a cizích klíčů tabulek, nastavení přesností u atributů v tabulkách. Pro odstranění dalších problémů funkčnost databáze musí být ověřena testy, proto částí implementace databáze je vytvoření testovacích skriptů. Při dobře navrhnutém databázovém modelu neměl by nastat problém při implementaci tabulek, ale počítáme se s možnými chybami v návrhu, které detekujeme v testech.

Implementace POJO (1,5 man-days)

Po vytvoření datové aplikace můžeme začít s práci na POJO(Plain Old Java Objects). Obvykle vytváření POJO se musí řídit diagramem tříd. Pro účely urychlení práce a zmenšení práce na dokumentaci tým analytiků rozhodl, že POJO bude vycházet z databázového modelu s modifikacemi pro jazyk Java. To znamená, že atributy Java tříd mohou mít jiné konzistentní datové typy, než mají odpovídající entity v DB, cizí klíče a vztahy mezi entitami jsou implementovány pomocí objektů.

Implementace DAO vrstvy aplikace (2 man-days)

Pro korektní komunikaci logiky aplikace s databázovým modelem musí být implementovány metody pro přistup k datům a pro jejich persistování v případě potřeby. DAO vrstva nejčastěji neobsahuje v sobě žádnou validaci ani business logiku a odpovídá za CRUD operace : create - vytvoření objektů a uložení do databáze, read - čtení položek databáze, update - úprava již uložených objektů v databázi, delete - odmazaní položek s databáze. Tato vrstva se vytváří pomoci dědění a implementaci generických interface, proto práci na kódu není hodně, ale všechny tyto funkčností musí být otestovány pro odhalení chyb a jejich odstranění.

Vytvoření service vrstvy (1 man-day)

Service vrstva je přechod mezi business logikou a datovou vrstvou aplikace. Pro odstranění řešení validace v business procesech aplikace, čistější a srozumitelnější kód, dořešení drobností ve vztazích mezi objekty používáme service vrstvu, která kontroluje výše uvedené věcí. U některých objektů není potřeba vytvářet service vrstvu, protože jejich persistování do DB a jejich používání v business procesech je minimální, proto zajistíme implementaci této vrstvy jen u těch tříd, které se hodně používají a hraji důležitou roli.

Vytvoření business procesů (5 man-days)

Během návrhu business procesů aplikace návrhář se orijentuje na business analýzu aplikace a dokumentuje pomocí diagramů ty aktivity, které je potřeba implementovat. Business procesy je implementace aktivit, které aplikace bude schopná vykonat. Business procesy jsou základem business logiky aplikace. Je potřeba věnovat dost času na analýzu procesu v aplikaci a jejich návrhu, aby se předešlo možným chybám při návrhu.

Implementace business komponent (6 man-days)

Při vytvoření business komponent vývojář se řídí hotovými diagramy aktivit a business procesů. Při správném návrhu jedná se o implementaci s možnými dopady na jiné funkčností a komponenty a taký konfigurace na úrovní frameworků. V momentě vytvoření business logiky aplikace vývojář by měl mít zajištěné korektní fungování DAO a service vrstvy pro zmenšení času, který se ztratí na odhalení a úpravu případných chyb. Každý business process aplikace měl by být pokrýt několika testy pro ověření core-funkčností aplikace.

Návrh WEB API aplikace (1,5 man-days)

Aplikace se záměrem na mobilní zařízení nejčastěji musí vystavovat webové rozhraní pro přistup k datům a používání core-logiky aplikace. Během návrhu webového API (Application Programming Interface) návrhář musí zjistit, jak bude komunikovat front-end část s back-end části, která data bude potřebovat front-end a jaké funkčností bude chtít využit front-end. Správný návrh webového API ušetří práci vývojáři a udělá přistup k business logice aplikace intuitivním.

Implementace WEB API apliakce (1 man-day)

Během vytváření webového rozhraní vývojář se řídí návrhem analytika. Jedná se o implementaci s využitím moderních technologií pro vystavování webového API jako Spring web-framework. Pro vytvoření funkčního webového API vývojář potřebuje mít hotové business procesy, proto tato funkčnost se implementuje jako poslední na BE.

Předpoklady a rizika

Náš projekt, cílem kterého je vytvoření aplikace, počítá se s možnými riziky, které mohou nastat během jeho realizace. Každá část logického rámce, která se týká vytvoření aplikace, uvádí vlastní předpoklady a rizika.
Při business analýze se zjistilo, že hlavní předpoklady jsou:

- Otevřenost trhu a poptávka po způsobech optimalizace práce
- Aplikací bez orientace na pracovní trh => malá konkurence
- Zkušený tým programátorů

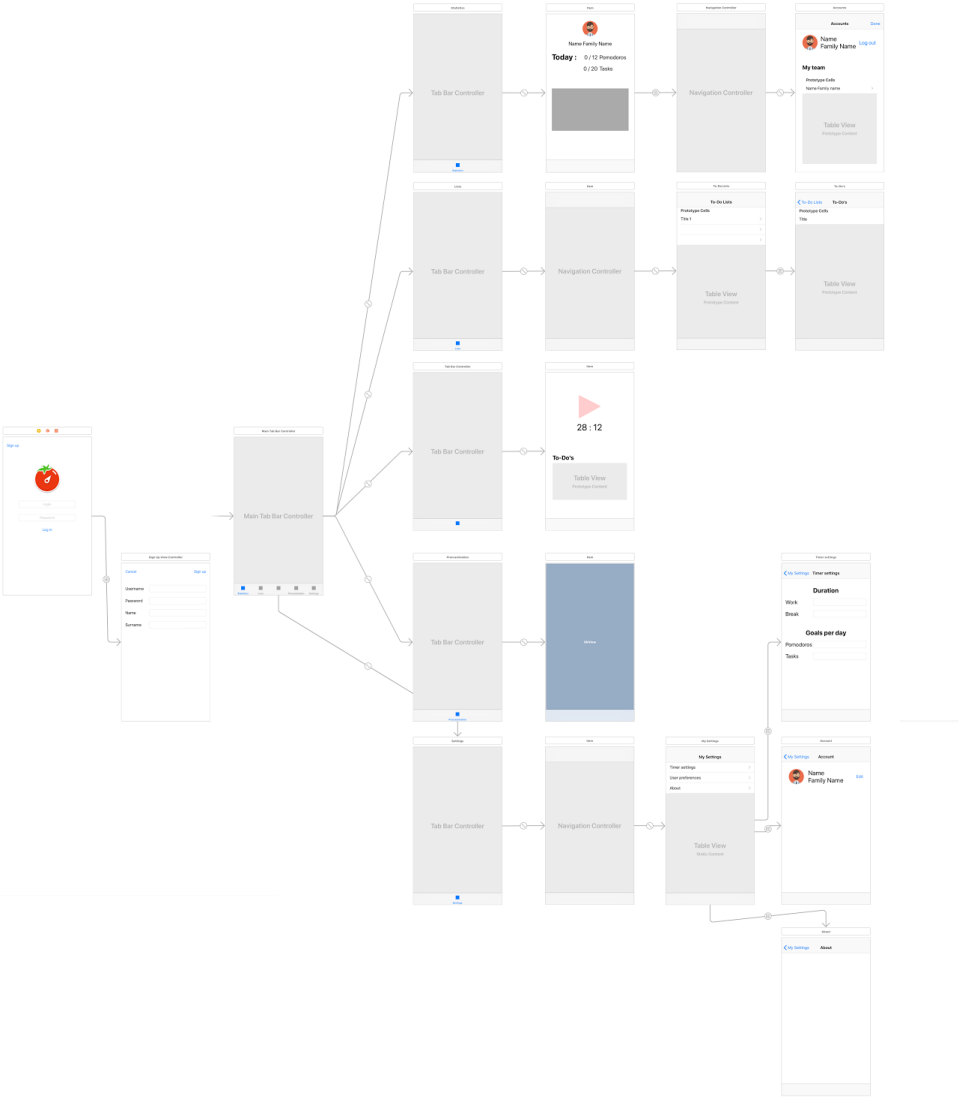
Na druhou stranu počítáme s rychlým růstem počtu podobných aplikací na trhu, což je jedním z hlavních rizik projektu.
Části pro výstupy a aktivity v projektu jsou těsně provázané mezi sebou, proto mají téměř stejná rizika a předpoklady. Hlavní rizika jsou:

- Potřeba využití neznámých technologií
- Problémy s přístupy k obchodům s aplikacemi
- Nespolehlivý Internet Service Provider
- Chybějící podklady od zákazníka
- Drahé licence na SW a HW
- Nesledování časového planu

Pro dosažení kvalitních výstupů předpokládáme aktivní spolupráci zákazníka a klademe větší důraz na práci business analytiků a designerů, což usnadní implementaci aplikace.

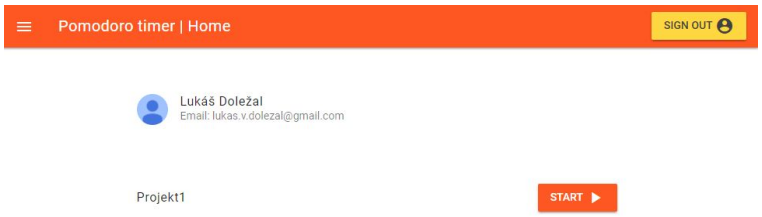
Wireframy

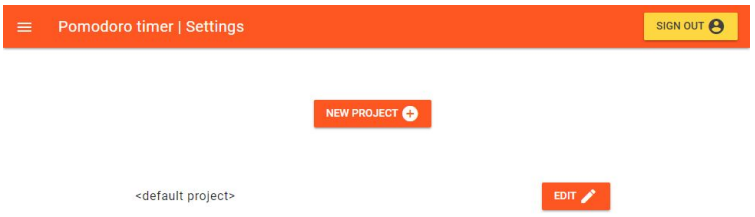
iOS wireframes

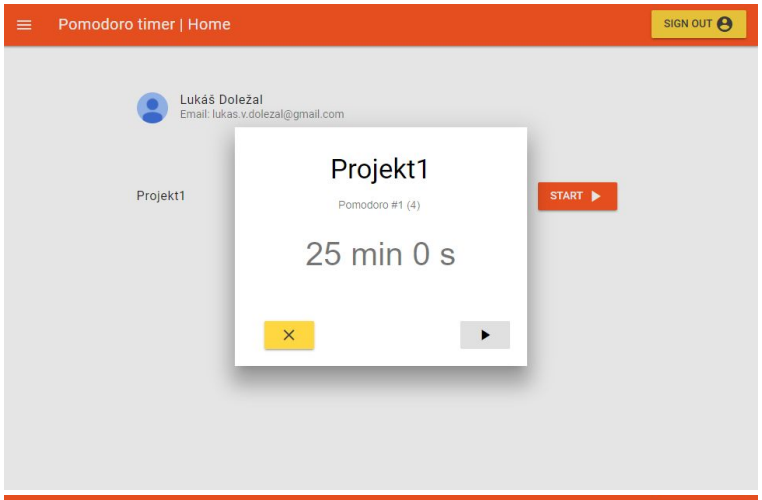


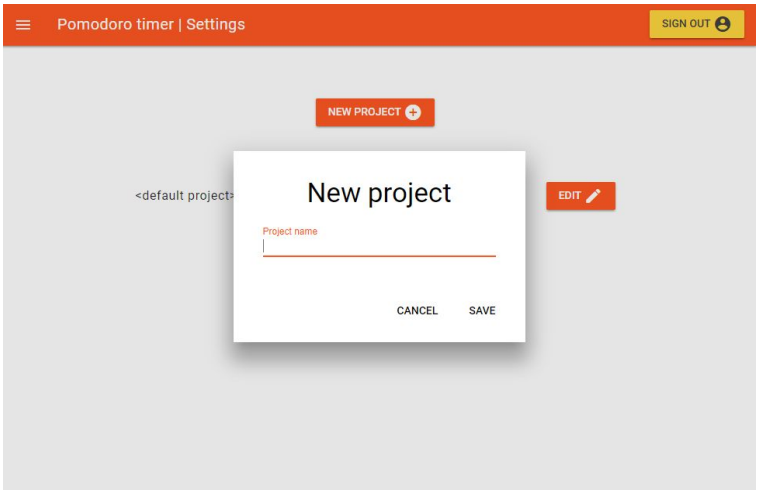
Vyexportováno ze storyboard

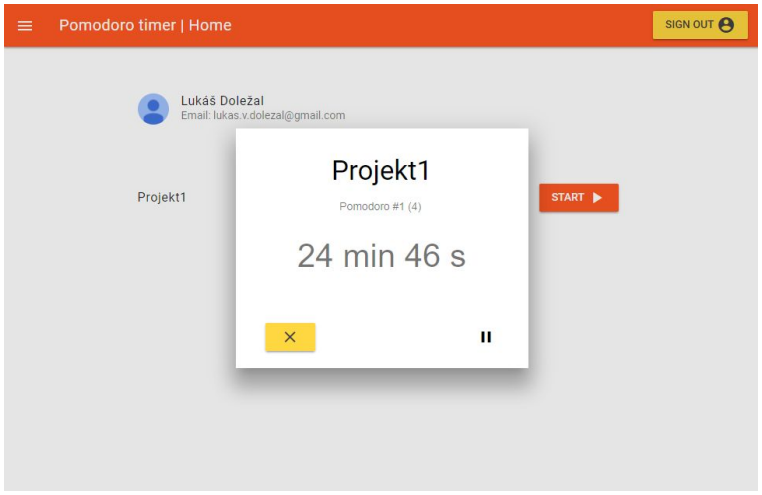
Web wireframes

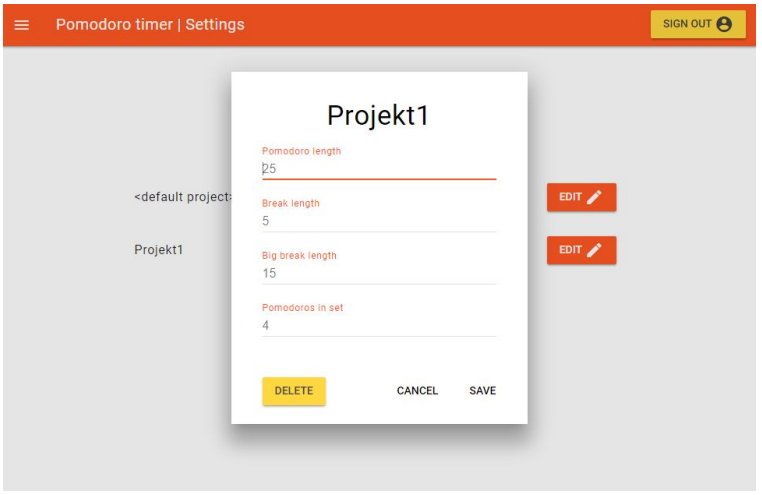












Nascreenováno z [vlastního webu na BI-SP1/2]