

Sem vložte zadání Vaší práce.



ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE  
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ  
KATEDRA SOFTWAREVÉHO INŽENÝRSTVÍ



Bakalářská práce

## Měřič času stráveného na ploše KDE pomocí služby Toggl

*Anastasiya Kachan*

Vedoucí práce: Ondřej Guth

15. května 2019



---

## Poděkování

Chtěla bych tímto poděkovat vedoucímu mé práce Ing. Ondřeji Guthovi, Ph.D. za cenné rady, trpělivost a ochotu pomoci při tvorbě této práce. Děkuji také celé své rodině a přátelům za morální podporu.



---

## Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 15. května 2019

.....

České vysoké učení technické v Praze

Fakulta informačních technologií

© 2019 Anastasiya Kachan. Všechna práva vyhrazena.

*Tato práce vznikla jako školní dílo na Českém vysokém učení technickém v Praze, Fakultě informačních technologií. Práce je chráněna právními předpisy a mezinárodními úmluvami o právu autorském a právech souvisejících s právem autorským. K jejímu užití, s výjimkou bezúplatných zákonných licencí a nad rámec oprávnění uvedených v Prohlášení na předchozí straně, je nezbytný souhlas autora.*

## **Odkaz na tuto práci**

Kachan, Anastasiya. *Měřič času stráveného na ploše KDE pomocí služby Toggl*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2019.



---

# Abstrakt

Cílem práce je navrhnout, implementovat a otestovat klient pro službu Toggl, jejíž hlavní vlastností je počítání času stráveného na každé virtuální ploše. Klient je určen uživateli desktopového prostředí KDE. Část návrhu je postavena na výsledcích získaných při analýze existujících aplikací a na analýze uživatelských požadavků. Implementační část popisuje technologii, která byla použita při realizaci programu. Klient je otestován klasickými metodami testování softwaru a ohodnocen uživateli pomocí testování použitelnosti.

**Klíčová slova** Desktopový měřič, Klient Toggl, KDE, Plasma 5.

---

# Abstract

The aim of the thesis is to design, implement and test the client for Toggl service, whose main feature is counting the time spent on every virtual desktop. The client is determined for users of the KDE desktop environment. Section Design is based on the analysis results of existing applications and on analysis of user requirements. Section Implementation describes technologies, which were used during program implementation. The client is tested with the classic testing methods and rated by users with the help of usability test.

**Keywords** Desktop's tracker, Toggl Client, KDE, Plasma 5.

---

# Obsah

<b>Úvod</b>	<b>1</b>
<b>1 Cíl práce</b>	<b>3</b>
<b>2 Analýza</b>	<b>5</b>
2.1 Současná řešení . . . . .	5
2.2 Analýza požadavků . . . . .	12
2.3 Případy užití . . . . .	13
<b>3 Návrh</b>	<b>19</b>
3.1 Uživatelské rozhraní . . . . .	19
3.2 Toggl API . . . . .	24
3.3 Diagram tříd . . . . .	28
<b>4 Realizace</b>	<b>31</b>
4.1 Nástroje, technologie a knihovny . . . . .	31
4.2 Napojení na Toggl API . . . . .	34
4.3 Výsledná aplikace . . . . .	36
<b>5 Testování</b>	<b>39</b>
5.1 Metody testování . . . . .	39
5.2 Testování použitelnosti . . . . .	39
5.3 Zhodnocení testování použitelnosti . . . . .	41
<b>Závěr</b>	<b>45</b>
<b>Literatura</b>	<b>47</b>
<b>A Seznam použitých zkratk</b>	<b>49</b>
<b>B Instalační příručka</b>	<b>51</b>

B.1	Diagram nasazení a závislosti . . . . .	51
<b>C</b>	<b>Obsah přiloženého USB</b>	<b>53</b>

---

## Seznam obrázků

2.1	Klient Toggl pro Linux . . . . .	6
2.2	Webová aplikace RescueTime . . . . .	7
2.3	Desktopový klient RescueTime . . . . .	7
2.4	Klient TopTracker . . . . .	8
2.5	Webová aplikace TimeCamp . . . . .	9
2.6	Desktopový klient Hubstaff . . . . .	10
2.7	Seznam účastníků . . . . .	14
2.8	Model případů užití. . . . .	15
3.1	WF1 Přihlášení . . . . .	20
3.2	WF2 Hlavní menu. . . . .	20
3.3	WF3 Nastavení. . . . .	21
3.4	WF4 Zobrazení časových záznamů a jejich úprava . . . . .	22
3.5	WF5 Detekce změny desktopu. . . . .	23
3.6	Příklad JSON-těla požadavku na založení záznamu . . . . .	25
3.7	Diagram tříd . . . . .	30
4.1	Příklad použití funkce XGetWindowProperty . . . . .	32
4.2	Příklad použití funkce get_property na zjištění čísla aktuálního desktopu . . . . .	32
4.3	Signály a sloty . . . . .	33
4.4	Signál a sloty třídy CTimeEntryList . . . . .	34
4.5	Spojení časovače a slot-funkce detectDesktopsProperty . . . . .	34
4.6	Příklad použití funkce authentication . . . . .	35
4.7	Příklad volání požadavku s tokenem v autentizační hlavičce . . . . .	35
4.8	Okénko s přihlášením . . . . .	36
4.9	Zobrazení časových záznamů . . . . .	37
4.10	Nastavení . . . . .	37
4.11	Tray Menu s pozastaveným měřičem . . . . .	38
4.12	Tray Menu se spuštěným měřičem . . . . .	38

4.13 Sloučení časových záznamů . . . . .	38
B.1 Diagram nasazení . . . . .	51

---

## Seznam tabulek

2.1	Přehled funkcí a vlastností trackerů . . . . .	11
2.2	Tabulka pokrytí funkčních požadavků . . . . .	17
3.1	Tabulka pokrytí wireframů . . . . .	24





---

# Úvod

V dnešní době je čas jednou z nejvýznamějších věcí našeho života. Je těžké najít čas na odpočinek nebo studium, ovšem ještě těžší je tento čas efektivně zorganizovat. Abychom tento čas dokázali najít, je vhodné se nejdříve zaměřit na to, kde jej ztrácíme. V tomto nám může pomoci statistika, kterou nabízí Toggl Service.

Tato bakalářská práce se zabývá návrhem a implementací aplikace, která dokáže daný problém vyřešit. Jistým specifikem je to, že daná aplikace bude určena pro uživatele prostředí KDE Plasma 5.

Aplikace bude implementována jako desktopový měřič s využitím služby Toggl. Takový desktopový měřič bude řešit problém managementu času stráveného u počítače za předpokladu, že každý používaný desktop bude určen pro jeden druh aktivity (např. práce, studium, zábava atd.). Uživatel daného klientu se pomocí této aplikace bude moci poté zamyslet nad organizací svého času.

Teoretická část této bakalářské práce se zabývá analýzou existujících řešení a použitím výsledků analýzy při návrhu cílové aplikace. Analýza vlastností bude provedena pouze u nejpopulárnějších trackerů.

Praktická část pak zahrnuje implementaci aplikace podle návrhu rozpracovaného v teoretické části.



---

## Cíl práce

Cílem této bakalářské práce je návrh a implementace klientu služby Toggl pro prostředí KDE Plasma 5, jehož hlavní vlastností je zaznamenávání množství času stráveného na dané ploše. Klient bude počítat čas na každém z aktivních virtuálních desktopů a bude schopen synchronizace s účtem Toggl.

Aplikace bude mít GUI, kde uživatel bude moci provádět jednotlivé akce jako sloučení a editace časových záznamů. Také umožní uživateli konfigurovat aplikaci podle vlastních potřeb.

Dalším cílem je umožnit, aby byl program šířitelný pod vhodnou open-source licencí, a otestovat výsledný klient vhodnými prostředky.



# Analýza

## 2.1 Současná řešení

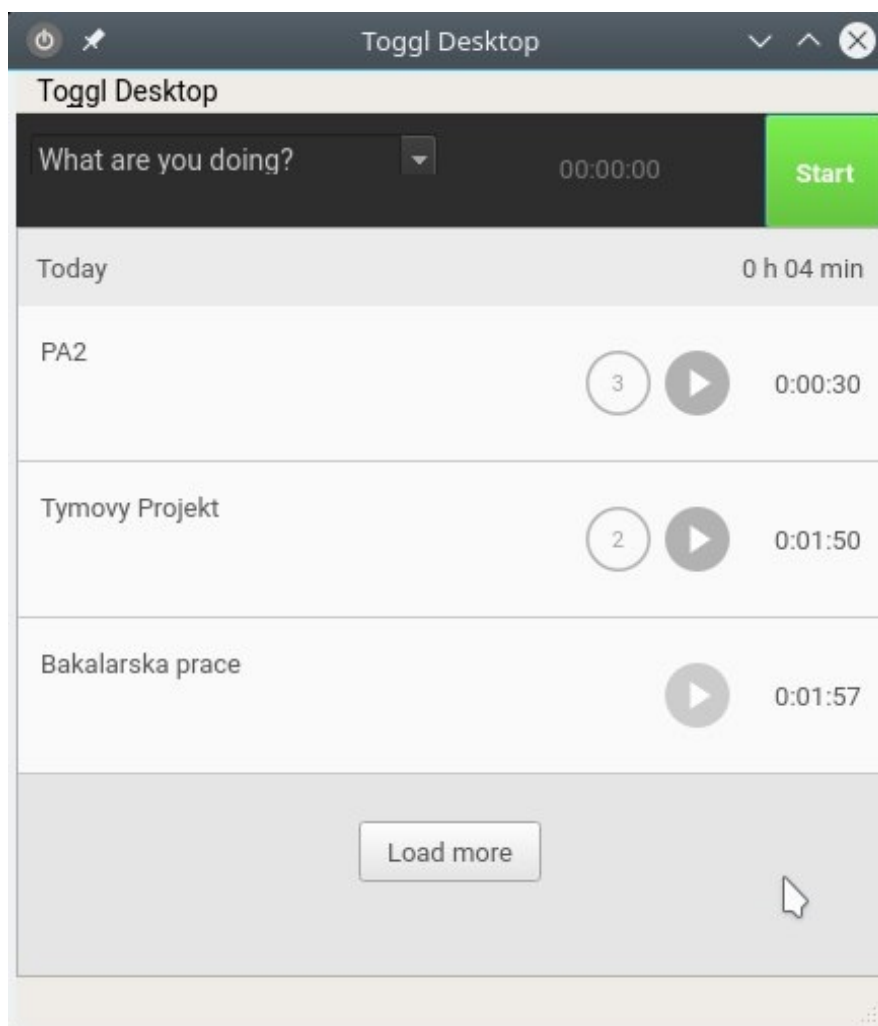
Za účelem porovnání Toggl trackeru jsem prozkoumala různé podobné měřiče času stráveného na ploše. Z důvodů existence velkého množství trackerů jsem musela řešit jen omezený počet. Trackery k porovnání jsem si vybírala podle popularity, tj. nejvyšších hodnot stažení, a podle žebříčku nejlepších trackerů posledních let sestaveného uživateli.

### 2.1.1 Služby Toggl

Mezi nejoblíbenější měřiče času pro uživatele patří Toggl [1]. Umožňuje přidání nekonečně mnoho záznamů anebo projektů, které se dají obarvit a vložit do nich další podzáznamy. Mezi hlavní výhody patří:

- na internetu dostupnost open source již existujících klientů a dokumentace,
- integrace s velkým množstvím služeb,
- přehledné uživatelské rozhraní.

Toggl je podporován jak webovými prohlížeči, tak i desktopovými a mobilními aplikacemi. Pokryty jsou skoro všechny platformy. Desktopová aplikace je dostupná pro systémy Windows, Linux (viz obrázek 2.1) a Mac OS, mobilní aplikaci je možno stáhnout na Android a iOS. Kromě toho jsou dostupná tlačítka pro internetové prohlížeče Google Chrome a Mozilla.



Obrázek 2.1: Klient Toggl pro Linux

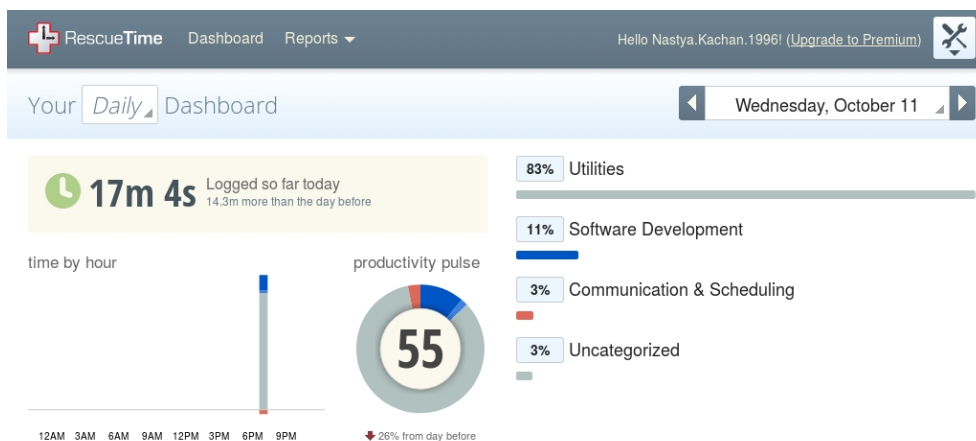
### 2.1.2 RescueTime

Tento klient [2] je jedním z nejpopulárnějších trackerů, který se skládá z aplikace pro sledování aktivity a online servisu. Je dostupný pro systémy Windows, Mac OS, Linux, existuje také mobilní aplikace pro Android (viz obrázky 2.2, 2.3).

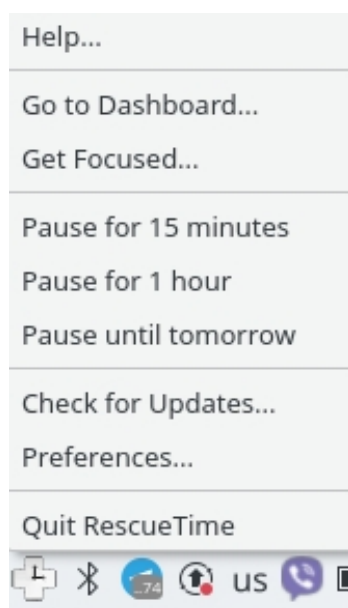
Dalšími funkcemi kromě měření času RescueTime jsou možnost sledování aktivity aplikací a webových stránek, které navštěvuje uživatel. Následuje také možnost třídění do různých kategorií (např. komunikace, utility atd.), můžeme se tak dozvědět množství stráveného času při použití těchto konkrétních nástrojů. RescueTime nabízí 4 typy výkazů, tedy využívané aplikace a navštívené webové stránky, druh aktivity, produktivitu a cíle, a to ve formě diagramů.

Doplňkovou funkcí je blokování přístupu na určité webové stránky v kon-

krétním časovém intervalu a blokování webových stránek z jedné i více kategorií.



Obrázek 2.2: Webová aplikace RescueTime



Obrázek 2.3: Desktopový klient RescueTime

### 2.1.3 TopTracker

Stejně jako RescueTime se skládá z desktopového klientu a webového servisu. Tento tracker je vhodný pro týmové používání, protože do konkrétního projektu můžeme přidávat osoby a následně tak zjistit celkový čas strávený na

## 2. ANALÝZA

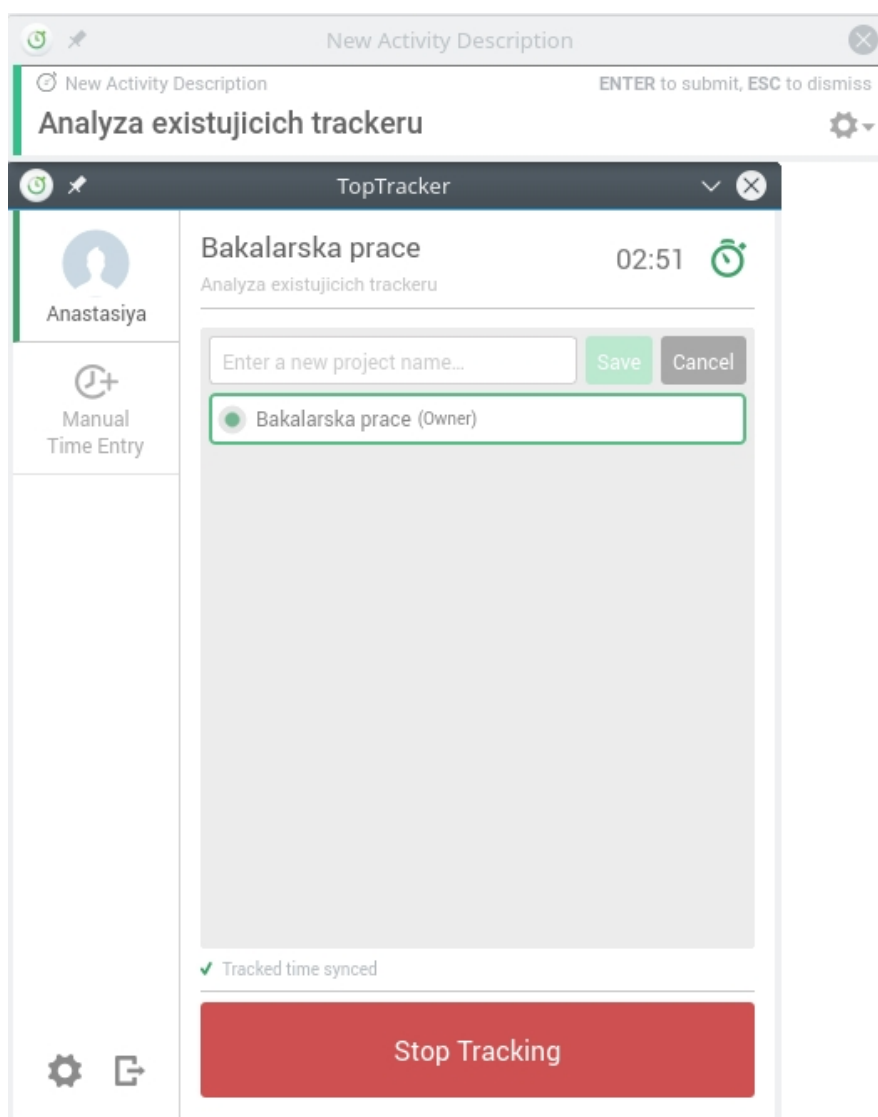
---

projektu i dobu, kterou na něm pracoval každý člen týmu. Kromě základního měření času můžeme zaznamenávat podrobné zprávy o progresu.

Další možností je zachycení screenshotu desktopu v pravidelných nebo náhodných intervalech a snímky z webcamery s možností rozmazání pro zachování konfidenciality.

Webová verze a klient jsou dostupné pro systémy Windows, Mac OS X a Debian (viz obrázek 2.4).

Nevýhodou tohoto klientu je neexistence mobilní aplikace pro Android a iOS [3].



Obrázek 2.4: Klient TopTracker

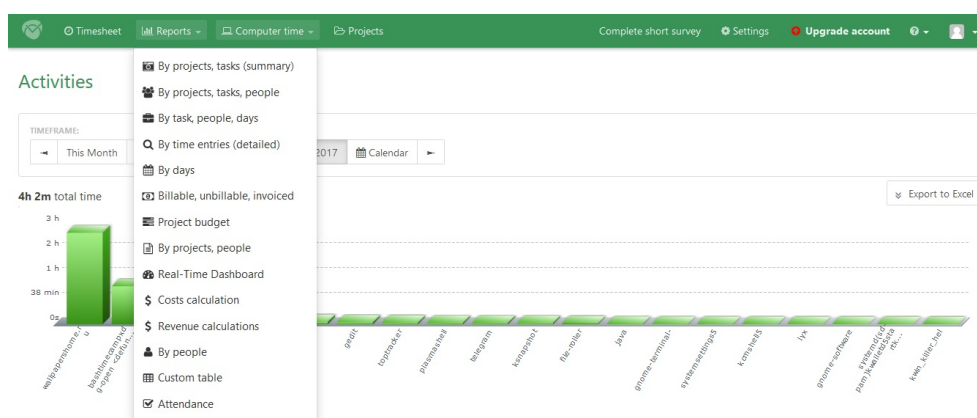


### 2.1.4 TimeCamp

Desktopový klient pro Linux poskytuje stejné služby jako klienty pro Windows, Mac OS, jimiž jsou správa časových rozvrhů, analýza produktivity, automatické sledování použitých počítačových aplikací, podrobná historie, seznam nejvíce časově náročných webových stránek a aplikací, integrovaná platební brána, kalendář svátků [4].

TimeCamp má také verzi v podobě mobilní aplikace pro Android a iOS.

Nevýhodou z mého pohledu je to, že se více zdůrazňuje webová verze (viz obrázek 2.5). Desktopová aplikace funguje na pozadí (sleduje aktivní počítačové programy) a má chudé desktopové uživatelské rozhraní.

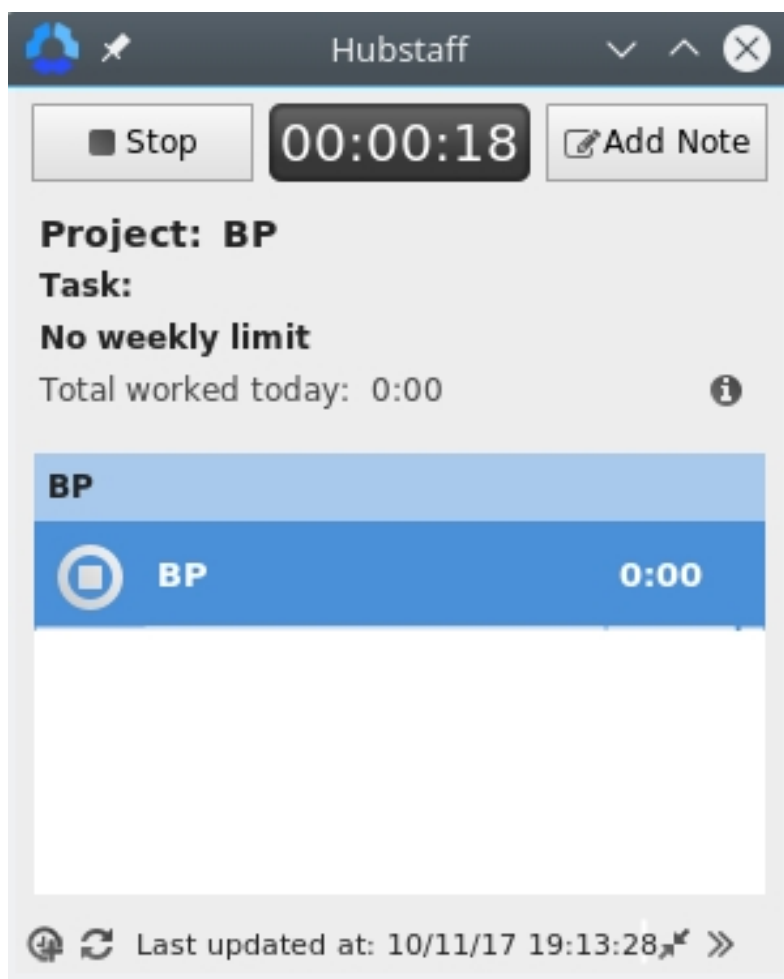


Obrázek 2.5: Webová aplikace TimeCamp

### 2.1.5 Hubstaff

Hubstaff [5] je vhodný jak pro jednoho uživatele, tak i pro kancelářskou práci v týmu stejně jako TopTracker. Tento nástroj je výhodný v oboustranném směru. Zaměstnavatel má možnost kontroly práce svých podřízených. Ze strany zaměstnanců výhoda spočívá v tom, že mohou být informováni o svých příjmech. Také je zde nabízena funkce zachycení screenshotu desktopu a sledování aktivních počítačových a webových aplikací pro vedení statistik.

Hubstaff je jedním z nejpopulárnějších trackerů. Existují desktopové klienty pro Windows, Mac OS X, Linux (viz obrázek 2.6) a mobilní aplikace pro Android a iOS. Dále má uživatel možnost instalace tohoto klientu do svého prohlížeče Google Chrome.



Obrázek 2.6: Desktopový klient Hubstaff

### 2.1.6 Souhrn

Seznámila jsem se s výše uvedenými měřiči času v podobě instalace klientů, do kterých jsem se přihlásila svým vlastním účtem, prozkoumala jsem je a zjistila, že ani jeden z nich neumožňuje automatickou detekci změny aktuální plochy v prostředí KDE Plasma 5 (jako i neumožňují ostatní měřiče pro unixové systémy). Více méně některé klienty jsou dobře zpracované, což může představovat inspiraci pro návrh a implementaci vlastního klientu. Pro přehlednost jsem vytvořila tabulku 2.1 porovnání existujících měřičů mezi sebou.

Tabulka 2.1: Přehled funkcí a vlastností trackerů

	RescueTime	TopTracker	TimeCamp	Hubstaff	Toggl
Přidávání lidí do týmu	-	+	+	+	+
Klient pro Windows	+	+	+	+	+
Klient pro Mac OS	+	+	+	+	+
Klient pro Linux	+	+	+	+	+
Klient pro Android	+	-	+	+	+
Klient pro iOS	-	-	+	+	+
Připomínání	-	-	-	-	+
Sledování počítačových aplikací a webových stránek	+	-	+	-	+
Statistika	+	-	+	-	+
Screenshots	-	+	-	+	-
Blokování webových stránek	+	-	-	-	-
Integrace s platebním účtem	-	-	+	+	-
Export dat	+	-	-	+	+

### 2.2 Analýza požadavků

V této sekci jsou uvedeny požadavky na aplikace. Ke každému požadavku jsem definovala úroveň priority v závislosti na hlavním cíli úkolu.

#### 2.2.1 Funkční požadavky

##### F1 Přihlášení/Odhlášení

Uživatel bude schopen se přihlásit ke svému účtu Toggl, pokud je připojen k internetu. Respektive může se odhlásit.

*Priorita: vysoká*

##### F2 Zobrazení záznamů

Klient zobrazuje všechny existující záznamy v podobě Desktop\_1, Desktop\_2 atd. (pokud uživatel nezvolil jiné názvy, klient nabízí defaultní názvy desktopů). Počet záznamů odpovídá skutečnému počtu virtuálních ploch.

*Priorita: střední*

##### F3 Smazání/přidání záznamů

Klient bude schopen reagovat na smazání uživateli virtuální plochy tím, že smaže poslední časový záznam (stejným způsobem klient bude schopen přidat nový záznam, pokud uživatel založí novou virtuální plochu).

*Priorita: vysoká*

##### F4 Úprava záznamů

Uživatel má možnost upravit záznamy (např. název).

*Priorita: střední*

##### F5 Spouštění/Zastavení měřiče

Uživatel může spustit resp. zastavit měřič.

*Priorita: vysoká*

##### F6 Sloučení záznamů

Uživatel má možnost sloučit vybrané záznamy do jednoho.

*Priorita: nízká*

##### F7 Detekce změny desktopu

Klient bude schopen reagovat na přepnutí desktopu uživatelem tak, že se automaticky přepne na záznam týkající se nyní aktuálního desktopu (měřič předchozího záznamu se zastaví, nový se spustí).

*Priorita: vysoká*

### F8 Nastavení

Klient umožňuje zobrazit profil nastavení a provést zde dílčí změny:

- automatické nebo ruční spuštění aplikace po zapnutí systému;
- nastavení časového intervalu připomínky;
- offline, resp. online režim aplikace (online režim – synchronizace se serverem, offline – vypnutí synchronizace);

*Priorita: střední*

### 2.2.2 Nefunkční požadavky

#### N1 Klient v prostředí KDE

Klient bude implementován pro desktopové prostředí KDE Plasma 5.

*Priorita: vysoká*

#### N2 Synchronizace

Klient bude schopen načíst data na server, pokud se s ním podaří spojit.

*Priorita: střední*

#### N3 Jazyk aplikace

Uživatelské rozhraní klientů bude v anglickém jazyce.

*Priorita: střední*

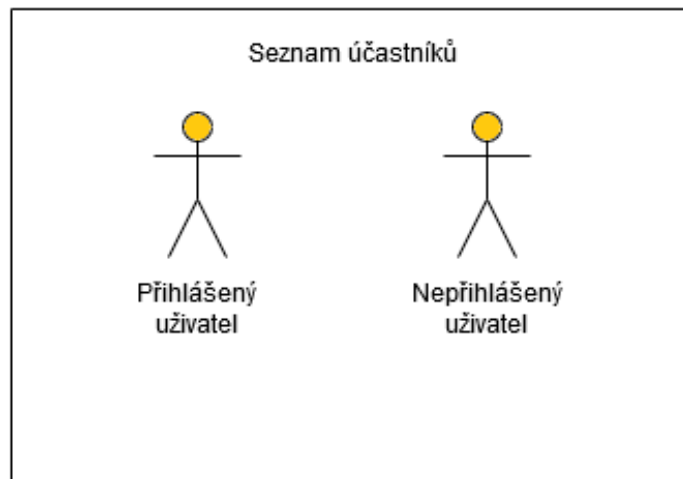
## 2.3 Případy užití

V této části jsou uvedeny případy užití aplikace ve formě scénáře, diagramu, identifikace účastníků. Všechny požadavky jsou pokryty, což se potvrzuje tabulkou pokrytí požadavků na konci této části.

### 2.3.1 Seznam účastníků

Uživatel naší aplikace se může nacházet ve dvou stavech: buď to přihlášený uživatel nebo nepřihlášený (viz obrázek 2.7).

- **Přihlášený uživatel** - uživatel, který je přihlášen ke službě Toggl.
- **Nepřihlášený uživatel** - uživatel, který není přihlášen ke službě Toggl.



Obrázek 2.7: Seznam účastníků

### 2.3.2 Scénáře případů užití

#### UC1 Přihlášení

*Hlavní scénář:*

1. Případ užití se začíná, když uživatel spustí aplikaci poprvé nebo byl-li odhlášen.
2. Otevře se přihlašovací okénko.
3. Uživatel zadá e-mail a heslo pro účet Toggl.

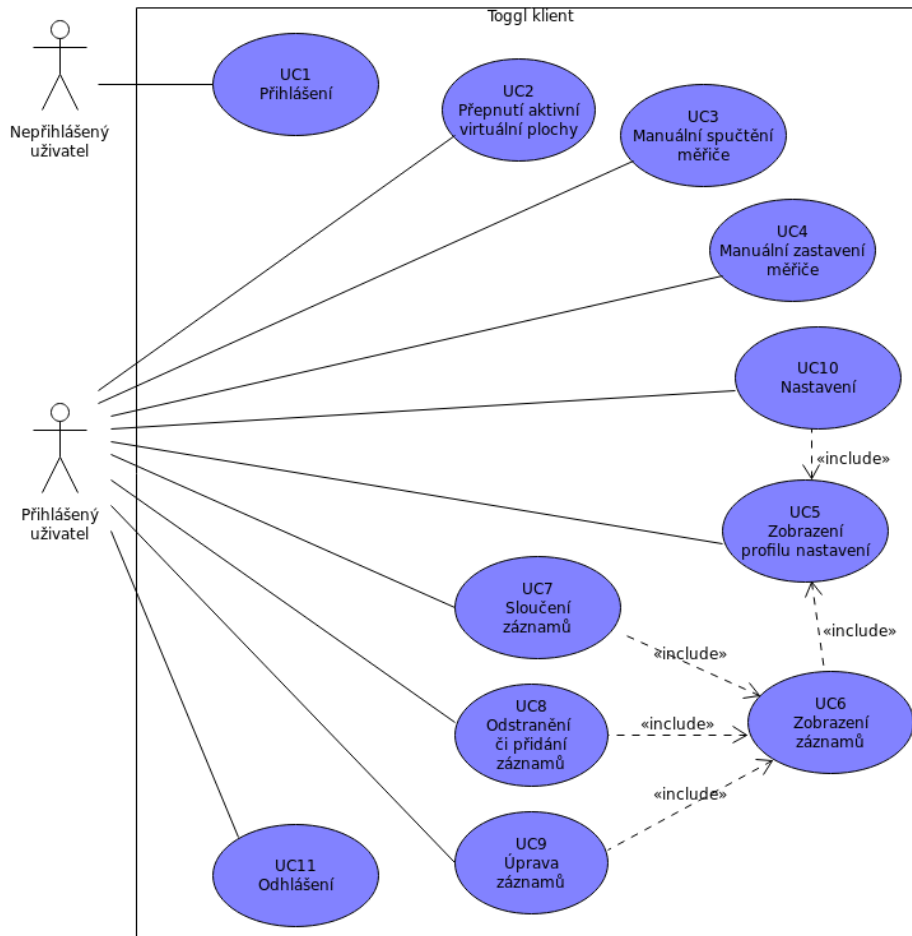
*Alternativní scénář:*

- 3.1. Uživatel zadá e-mail a heslo pro Google účet.
4. Uživatel je přihlášen, pokud je připojen k internetové síti a zadané údaje jsou správné.

#### UC2 Přepnutí aktivní virtuální plochy

*Hlavní scénář:*

1. Případ užití se začíná, když se chce uživatel přepnout z plochy X do plochy Y.
2. Uživatel přepne plochu.
3. Systém definuje plochu X jako neaktivní a plochu Y jako aktivní.



Obrázek 2.8: Model případů užití.

**UC3 Manuální spuštění měřiče***Hlavní scénář:*

1. V případě, kdy měřič není aktivní, systém nabídne uživateli jej spustit.
2. Uživatel požádá systém o spuštění měřiče tím, že zmáčkne tlačítko „Start“.
3. Systém spustí měřič na aktivní ploše.

**UC4 Manuální zastavení měřiče***Hlavní scénář:*

1. V případě, kdy měřič je aktivní, systém nabídne uživateli zastavit jej.
2. Uživatel požádá systém o zastavení měřiče tím, že zmáčkne tlačítko

## 2. ANALÝZA

---

„Stop“.

3. Systém zastaví měřič.

### UC5 Zobrazení profilu nastavení

*Hlavní scénář:*

1. Příklad užití se začíná, je-li uživatel přihlášen.
2. Uživatel zvolí položku „Preferences“.
3. Systém zobrazí profil nastavení.
4. Systém nabídne uživateli složku systémového nastavení a složku nastavení desktopových záznamů.

### UC6 Zobrazení záznamů

*Hlavní scénář:*

1. Příklad užití se začíná, když si uživatel chce prohlédnout seznam záznamů.
2. *Include* UC5 Zobrazení profilu nastavení.
3. Uživatel zvolí složku nastavení desktopových záznamů.
4. Systém zobrazí seznam záznamů.

### UC7 Sloučení záznamů

*Hlavní scénář:*

1. Příklad užití začíná, chce-li uživatel sloučit některé záznamy.
2. *Include* UC6 Zobrazení záznamů.
3. Uživatel vybere záznam, který chce přidat do jiného, a přitáhne jej (drag-and-drop).
4. Systém sečte čas obou záznamů a uloží do cílového.

### UC8 Odstranění/přidání záznamů

*Hlavní scénář:*

1. Příklad užití se začíná, jestliže uživatel chce smazat, resp. přidat některou virtuální plochu.
2. Uživatel smaže, resp. přidá virtuální plochu.
3. Systém odstraní poslední časový záznam, resp. přidá nový.

### UC9 Úprava záznamů

*Hlavní scénář:*

1. Příklad užití se začíná, jestliže chce uživatel upravit záznam.
2. *Include* UC6 Zobrazení záznamů.
3. Uživatel zvolí záznam, který chce upravit.
4. Uživatel provede úpravu záznamu (např. přejmenování).
5. Systém uloží změny.

### UC10 Nastavení

*Hlavní scénář:*

1. Příklad užití se začíná, jestliže uživatel chce provést nastavení u měřiče.
2. *Include* UC5 Zobrazení profilu nastavení.



3. Uživatel provede jednotlivé nastavení.
4. Systém uloží změny.

**UC11 Odhlášení***Hlavní scénář:*

1. Příklad užití se začíná, je-li uživatel přihlášený a chce se odhlásit.
2. Uživatel zvolí tlačítko „Log out“.
3. Systém odhlásí uživatele.

**2.3.3 Mapování případů užití**

Následující tabulka 2.2 ilustruje pokrytí požadavků na aplikace případy užití.

Tabulka 2.2: Tabulka pokrytí funkčních požadavků

	F1	F2	F3	F4	F5	F6	F7	F8
UC1	x							
UC2							x	
UC3					x			
UC4					x			
UC5								x
UC6		x						
UC7						x		
UC8			x					
UC9				x				
UC10								x
UC11	x							



## Návrh

### 3.1 Uživatelské rozhraní

Tato část práce pojednává o uživatelském rozhraní budoucího klientu, které jsem představila ve formě wireframů. Hlavním cílem wireframů je ukázat funkčnost klientu.

Klient bude vypadat jako standartní KDE Plasma 5 aplikace a bude reprezentován jako ikona v dolním pravém rohu na hlavním panelu. Po otevření klientu se objeví grafické uživatelské rozhraní, které je potřebné na provedení jednoduchých úprav nastavení klientu.

#### 3.1.1 Přihlášení

Při prvním spuštění klientu se zobrazí přihlašovací okénko. Uživatel zadá svůj email pro Toogl účet (případně Google účet), heslo a potvrdí je. Klient odešle data na server, a pokud jsou správné, uživatel je úspěšně přihlášen (viz obrázek 3.1).

#### 3.1.2 Hlavní menu, spuštění a zastavení měřiče, odhlášení

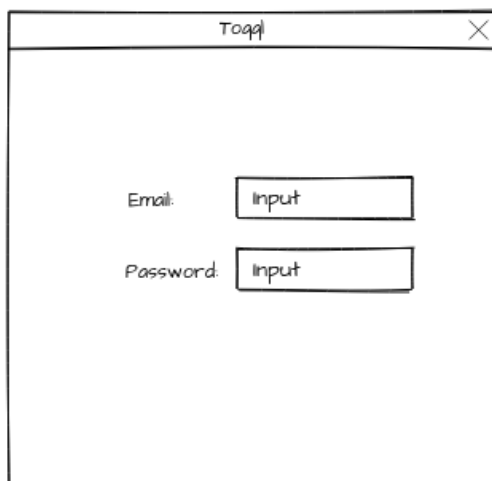
Hlavní menu klientu závisí na tom, jestli je uživatel přihlášen nebo ne.

Pokud uživatel není přihlášen, menu se skládá ze třech položek: „Open Toggl“ – otevření aplikace, pokud je spuštěna, ale její hlavní okno není otevřeno; „Go to website“ – link na webovou stránku Toggl service; „Exit“ – zavření celé aplikace.

Po přihlášení se hlavní menu rozšíří o položky „Start...“, resp. „Stop...“ – umožňují spustit, resp. zastavit měřič bez nutnosti otevírání hlavního okna, „Log out“ – odhlášení (viz obrázek 3.2).

### 3. NÁVRH

---

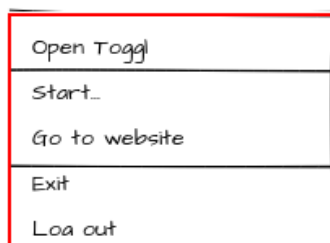


Tagg

Email:

Password:

Obrázek 3.1: WF1 Přihlášení



Open Tagg

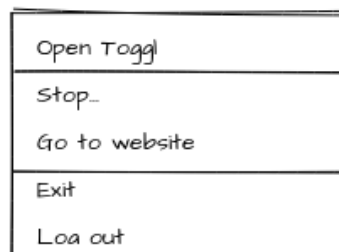
Start\_

Go to website

Exit

Loa out

(a) Přihlášený uživatel, měřič je zastaven



Open Tagg

Stop\_

Go to website

Exit

Loa out

(b) Přihlášený uživatel, měřič je spuštěn



Open Tagg

Go to website

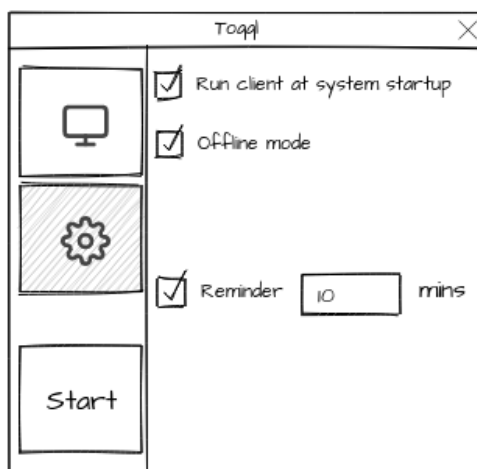
Exit

(c) Nepřihlášený uživatel

Obrázek 3.2: WF2 Hlavní menu.

### 3.1.3 Zobrazení profilu nastavení

Po kliknutí na položku „Open Toggl“ v hlavním menu systém otevře hlavní okénko s nastaveními a časovými záznamy. Intuitivně (podle obrázku) je možné poznat, která složka odpovídá nastavení. Uživatel zvolí správnou složku a systém zobrazí profil, kde uživatel může provést jednotlivé úpravy (viz obrázek 3.3).



Obrázek 3.3: WF3 Nastavení

### 3.1.4 Zobrazení záznamů a jejich úprava

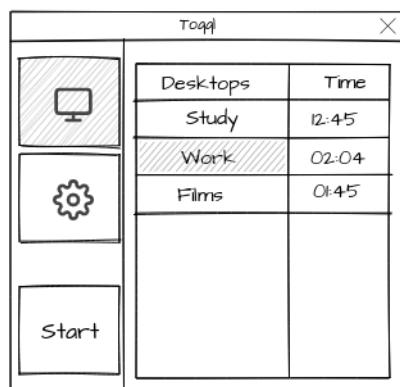
Když uživatel zvolí složku „Open Toggl“ v hlavním menu, otevře se okénko s seznamem časových záznamů pro každou plochu. Tady má každá plocha své vlastnosti: název a celkový čas strávený na této ploše během aktuálního dne. Uživatel má možnost upravit některé z těchto vlastností, například může přejmenovat název plochy. Také se dá sloučit některé záznamy pomocí metody drag-and-drop (viz obrázek 3.3).

### 3.1.5 Detekce změny desktopu

Klient umožňuje zobrazovat informaci, která plocha je aktivní (tzn. kterou plochu aktuálně uživatel používá), což ukáže tak, že se obarví záznam týkající se aktivní plochy jinou barvou než ostatní. V případě, kdy je měřič spuštěn, systém přestane počítat čas na neaktivní ploše a začne počítat čas na právě aktivní (viz obrázek 3.5).

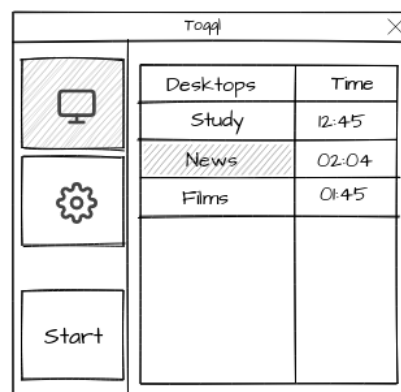
### 3. NÁVRH

---



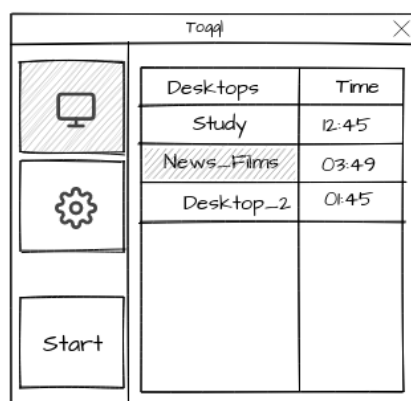
Desktops	Time
Study	12:45
Work	02:04
Films	01:45

(a) Zobrazení časových záznamů před úpravou



Desktops	Time
Study	12:45
News	02:04
Films	01:45

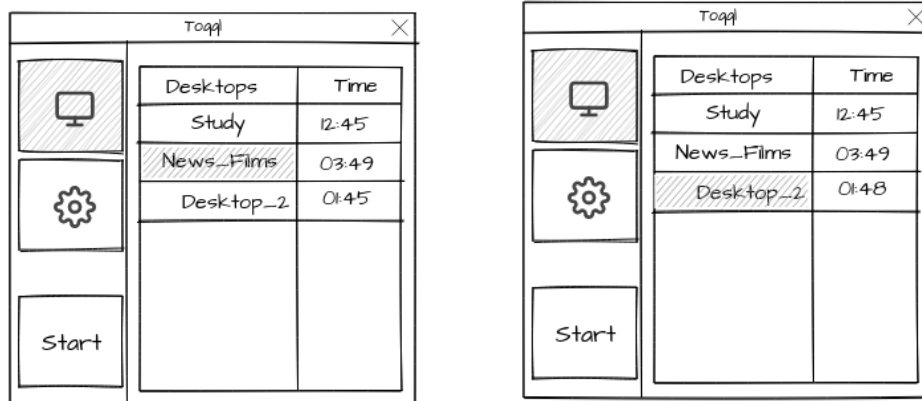
(b) Zobrazení časových záznamů po úpravě



Desktops	Time
Study	12:45
News_Films	03:49
Desktop_2	01:45

(c) Zobrazení časových záznamů po úpravě

Obrázek 3.4: WF4 Zobrazení časových záznamů a jejich úprava



(a) Uživatel je na ploše „News\_Films“

(b) Uživatel je na ploše „Desktop\_2“

Obrázek 3.5: WF5 Detekce změny desktopu.

### 3.1.6 Shrnutí

V této části byly představeny wireframy ve formě obrázku, které ilustrují úplné pokrytí případů užití a navrhují design budoucího klientu. Tabulka 3.1 potvrzuje, že všechny případy užití jsou pokryty wireframy.

	WF1	WF2	WF3	WF4	WF5
UC1	x				
UC2					x
UC3		x		x	
UC4		x		x	
UC5			x		
UC6				x	
UC7				x	
UC8				x	x
UC9				x	
UC10			x		
UC11		x			

Tabulka 3.1: Tabulka pokrytí wireframů

## 3.2 Toggl API

Toggl Service nabízí veřejné rozhraní sloužící pro přístup k ovládání účtu služby na serveru [6]. Toto rozhraní pokrývá celou funkčnost služby. Toggl REST API akceptují jenom požadavky ve formátu JSON kódu, ve stejném formátu také vrací odpověď. Výsledek každé provedené akce je přenášen pomocí standardních kódů odpovědi HTTP.

Jelikož těchto metod je velké množství a pro účely mé aplikace nejsou potřeba všechny z nich, projdu podrobněji jenom ty, které jsou nejzajímavější.

### 3.2.1 Autentizace

Autentizace probíhá pomocí **HTTP Basic Auth** [7], kde autentizační hlavička požadavku vypadá následovně:

- username a heslo se zřetězí do tvaru „*username:heslo*“ nebo „*<users\_token>:api\_token*“ při používání tokenu;
- výsledný řetězec se kóduje pomocí kódování **Base64**;
- zakódovaný řetězec se kombinuje tímto způsobem: „*Basic <encoded\_string>*“ (všimněte si mezery mezi „Basic“ a zakódovaným řetězcem);
- výsledná hlavička má tvar například: „*Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==*“;

V případě úspěšné autentizace požadavek poskytne informaci o uživateli.



### 3.2.2 Časové záznamy

Časy a data používají standart **ISO 8601**, konkrétně podmnožinu popsanou v **RFC 3339** [8]. Každý časový záznam [9] má následující atributy:

- *string* **description** – jméno záznamu, požadováno;
- *integer* **wid** – ID workspace, požadováno pokud nejsou definovány tip a pid;
- *integer* **pid** – ID projektu, není požadováno;
- *integer* **tid** – ID tasku, není požadováno;
- *boolean* **billable** – jenom pro workspace, jinak není požadováno;
- *string* **start** – ISO 8601 datum a čas, požadováno;
- *string* **stop** – ISO 8601 datum a čas, není požadováno;
- *integer* **duration** – doba trvání v sekundách. Jestliže časový záznam aktuálně běží, atribut bude mít negativní hodnotu označující čas a datum zadávání času v sekundách od epochy (1 ledna 1970), požadováno;
- *string* **created\_with** – název klientů, který používá API, požadováno;
- *array of strings* **tags** – poznámky, není požadováno;
- *boolean* **duronly** – zde má Toggl ukazovat čas začátku a konce tohoto času, není požadováno;
- *datetime* **at** – timestamp, který byl odeslán v odpovědi, čas poslední úpravy;

JSON-tělo požadavku na založení záznamu může vypadat následovně:

```

1 {
2   "time_entry":
3   {
4     "description": "Bakalarska prace",
5     "duration": 1200,
6     "start": "2013-03-05T07:58:58.000Z",
7     "created_with": "Client Toggl"
8   }
9 }
```

Obrázek 3.6: Příklad JSON-těla požadavku na založení záznamu

### 3. NÁVRH

---

#### **Založení**

metoda: **POST**

url: [https://www.toggl.com/api/v8/time\\_entries](https://www.toggl.com/api/v8/time_entries)

#### **Spuštění**

metoda: **POST**

url: [https://www.toggl.com/api/v8/time\\_entries/start](https://www.toggl.com/api/v8/time_entries/start)

#### **Zastavení**

metoda: **PUT**

url: [https://www.toggl.com/api/v8/time\\_entries/<time\\_entry\\_id>/stop](https://www.toggl.com/api/v8/time_entries/<time_entry_id>/stop)

#### **Získání detailů o záznamu**

metoda: **GET**

url: [https://www.toggl.com/api/v8/time\\_entries/<time\\_entry\\_id>](https://www.toggl.com/api/v8/time_entries/<time_entry_id>)

#### **Aktuálně běžící záznamy**

metoda: **GET**

url: [https://www.toggl.com/api/v8/time\\_entries/current](https://www.toggl.com/api/v8/time_entries/current)

#### **Aktualizovat záznam**

metoda: **PUT**

url: [https://www.toggl.com/api/v8/time\\_entries/<time\\_entry\\_id>](https://www.toggl.com/api/v8/time_entries/<time_entry_id>)

#### **Smazání**

metoda: **DELETE**

url: [https://www.toggl.com/api/v8/time\\_entries/<time\\_entry\\_id>](https://www.toggl.com/api/v8/time_entries/<time_entry_id>)

#### **Získání časových záznamů v určitém časovém intervalu**

metoda: **GET**

url: [https://www.toggl.com/api/v8/time\\_entries](https://www.toggl.com/api/v8/time_entries)

#### **Hromadná aktualizace úkolů**

metoda: **PUT**

url: [https://www.toggl.com/api/v8/time\\_entries/<id1,id2,id3,...>](https://www.toggl.com/api/v8/time_entries/<id1,id2,id3,...>)

### **3.2.3 Další požadavky**

Kromě požadavků na provádění akcí s časovými záznamy API umožňují:

#### **Klienty**

- vytvoření,
- získání detailů,
- aktualizace,
- odstranění,

- nastavení viditelnosti klientů pro uživatele,
- získání klientských projektů.

### Skupiny

- vytvoření,
- aktualizace,
- odstranění.

### Projekty

- vytvoření,
- získání detailů,
- aktualizace,
- odstranění,
- získání uživatele projektu,
- získání úkolů projektu,
- odstranění více projektů.

### Uživatelé projektu

- vytvoření uživatele projektu,
- aktualizace uživatele projektu,
- odstranění uživatele projektu,
- přidání více uživatelů do projektu,
- aktualizace více uživatelů projektu,
- odstranění více uživatelů z projektu.

### Poznámky

- vytvoření,
- aktualizace,
- odstranění.

### Úkoly

- vytvoření,
- získání detailů,
- aktualizace,
- odstranění,
- aktualizace více úkolů,

### 3. NÁVRH

---

- odstranění více úkolů.

#### **Uživatelé**

- získání detailů a časových záznamů uživatele,
- aktualizace aktuálních uživatelských dat,
- resetování API tokenu,
- registrace nového uživatele.

#### **Pracovní prostory (workspaces)**

- získání uživatelských pracovních prostorů,
- získání uživatelů pracovního prostoru,
- získání klientů pracovního prostoru,
- získání skupin pracovního prostoru,
- získání projektů pracovního prostoru,
- získání úkolů pracovního prostoru,
- získání poznámek pracovního prostoru.

#### **Uživatelé pracovních prostorů**

- pozvat uživatele,
- aktualizovat uživatele,
- odstranit uživatele,
- získat uživatele pro pracovní prostor.

#### **Panel nástrojů (dashboard)**

- získání obecného přehledu o daném týmu.

## 3.3 Diagram tříd

Na obrázku 3.7 je představen diagram tříd cílové aplikace.

Každá třída dědí z příslušné abstraktní třídy její rozhraní. Takový postup implementace byl zvolen z testovacích důvodů (rozhraní umožňují vytvářet *mock třídy*).

Třída *CSinglenon* zajišťuje existenci pouze jedné instance celé aplikace. Realizaci této funkčnosti nabízí Qt třída *QApplication*, ze které dědí *CSingleton*.

*CWindowApplication* dědí z Qt třídy *QMainWindow* a je hlavním oknem aplikace. Třída vykonává funkce controlleru, který rozhoduje, zda se má ukázat přihlašovací okénko nebo vnitřní rozhraní aplikace. Navíc je v její zodpovědnosti takzvané Tray Menu.

Intuitivně je možné pochopit, že třída *CLogin* zodpovídá za přihlašování. Jedná se o GUI model, a proto se dědí z *QWidget* třídy.

Na provádění autentizace *CLogin* potřebuje instanci třídy *CHTTPReceiver*, která se zabývá komunikací se severem pomocí Toggl REST API.

Třída *CToggl* je GUI model realizující kostru vnitřního rozhraní aplikace. Je logickým controllerem vnitřního menu a spravuje vzhled aplikace v závislosti na událostech (jestli uživatel chce otevřít menu nastavení nebo chce si prohlédnout seznam časových záznamu, případně zastavit nebo spustit měřič).

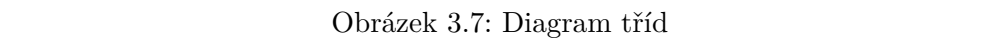
Třída *CSettingMenu* je GUI modelem a nabízí uživateli možnost konfigurovat aplikaci dle vlastních potřeb.

Třída *CTableModel* se dědí z *QAbstractTableModel* a má přepsané metody ze základní třídy. Třída simuluje chování tabulky a pracuje s jejími daty: vložit řádek, smazat řádek, editovat obsah atd.

Třída *CFileManager* je zodpovědná za kteroukoliv akci spojenou s prací na konfiguračních souborech. Instance této třídy jsou potřebné u *CSettingsMenu* (uložení do konfiguračního souboru nastavení), *CHTTPReceiver* (uložení do konfiguračního souboru tokenu a jména uživatele podle Toggl účtu), *CTableModel* (uložení nového jména časového záznamu po uživatelských úpravách) a *CTimeEntryList* (aktualizace v konfiguračním souboru záznamů po zastavení měřiče).

*CTimeEntryList* detektuje změny statusu virtuálních ploch (například přepnutí z jedné plochy do jiné). Navíc je grafickým modulem, který zobrazuje časové záznamy (obsahuje instanci *CTableModel* a obsah její tabulky reaguje na změny statusu ploch).

*CTimeEntryItem* popisuje časové záznamy, obsahuje metody pro generaci JSON-těla požadavků.



30

---

# Realizace

## 4.1 Nástroje, technologie a knihovny

### 4.1.1 Desktopové prostředí

KDE je desktopové prostředí pro unixové operační systémy, které používají grafické systémy X Window System a Wayland. Pro účely mé bakalářské práce bylo použito KDE poslední generace, tj. KDE Plasma 5 (verze KDE v momentě realizace 5.12.0), která je postavena na Qt5.

### 4.1.2 Vývojové prostředí

Celá aplikace byla vyvinuta v **Qt Creator** – oficiální multiplatformní IDE určené k vývoji C++ aplikací postavených na Qt toolkit(verze IDE v momentě realizace 4.9.0).

### 4.1.3 Repozitář a verzovací nástroje

Verzovacím nástrojem byl zvolen **Git** s repozitářem na školním GitLabu.

### 4.1.4 Dokumentace

Pomocí nástroje **Doxygen** byly vygenerovaná HTML stránky s exportovanými komentáři a grafy závislosti.

### 4.1.5 X11

X Window systém je okenní systém, který se skládá z několika základních komponent: X Server, X protokol, knihovna Xlib [10]. Server zprostředkovává vstup uživatele a přijímá výstupní požadavky z různých klientských programů umístěných buď na stejném počítači, nebo jinde v síti. Xlib je knihovna napsaná v jazyce C, kterou aplikační programy (klienti) používají k propojení se systémem oken pomocí připojení k datovému proudu.

### Knihovna Xlib

Hlavním důvodem použití této knihovny byl problém detekce statusu virtuálních ploch. Xlib poskytuje metodu `XGetWindowProperty`, která byla použita jako řešení tohoto problému. Na obrázku 4.1 je uveden kód vlastní funkce, která používá `XGetWindowProperty` [11]. Na základě argumentu funkce `prop_name` poté můžeme určit potřebné vlastnosti.

```

1 char * CTimeEntryList::get_property (Display *disp, Window win, Atom xa_prop_type,
2                                     char *prop_name, unsigned long *size) {
3     Atom xa_prop_name;
4     Atom xa_ret_type;
5     int ret_format;
6     unsigned long ret_nitems;
7     unsigned long ret_bytes_after;
8     unsigned long tmp_size;
9     unsigned char *ret_prop;
10    char* ret;
11    xa_prop_name = XInternAtom(disp, prop_name, False);
12    if (XGetWindowProperty(disp, win, xa_prop_name, 0, MAX_PROPERTY_VALUE_LEN / 4, False,
13                          xa_prop_type, &xa_ret_type, &ret_format,
14                          &ret_nitems, &ret_bytes_after, &ret_prop) != Success) {
15        return nullptr;
16    }
17    if (xa_ret_type != xa_prop_type) {
18        XFree(ret_prop);
19        return nullptr;
20    }
21    tmp_size = (static_cast<unsigned long>(ret_format) / 8) * ret_nitems;
22    ret = new char[tmp_size + 1];
23    memcpy(ret, ret_prop, tmp_size);
24    ret[tmp_size] = '\0';
25    if (size) {
26        *size = tmp_size;
27    }
28    XFree(ret_prop);
29    return ret;
30 }

```

Obrázek 4.1: Příklad použití funkce `XGetWindowProperty`

```

1 int CTimeEntryList::getCurrentDesktop() {
2     Window root = DefaultRootWindow(QX11Info::display());
3     char * cur_desktop = nullptr;
4     if (!(cur_desktop = get_property(QX11Info::display(), root,
5                                     XA_CARDINAL, NET_CURRENT_DESKTOP.toLocal8Bit().data(), nullptr))) {
6         if (!(cur_desktop = get_property(QX11Info::display(), root,
7                                         XA_CARDINAL, WIN_WORKSPACE.toLocal8Bit().data(), nullptr))) {
8             return EXIT_FAILURE;
9         }
10    }
11    return static_cast<int>(*cur_desktop);
12 }
13

```

Obrázek 4.2: Příklad použití funkce `get_property` na zjištění čísla aktuálního desktopu

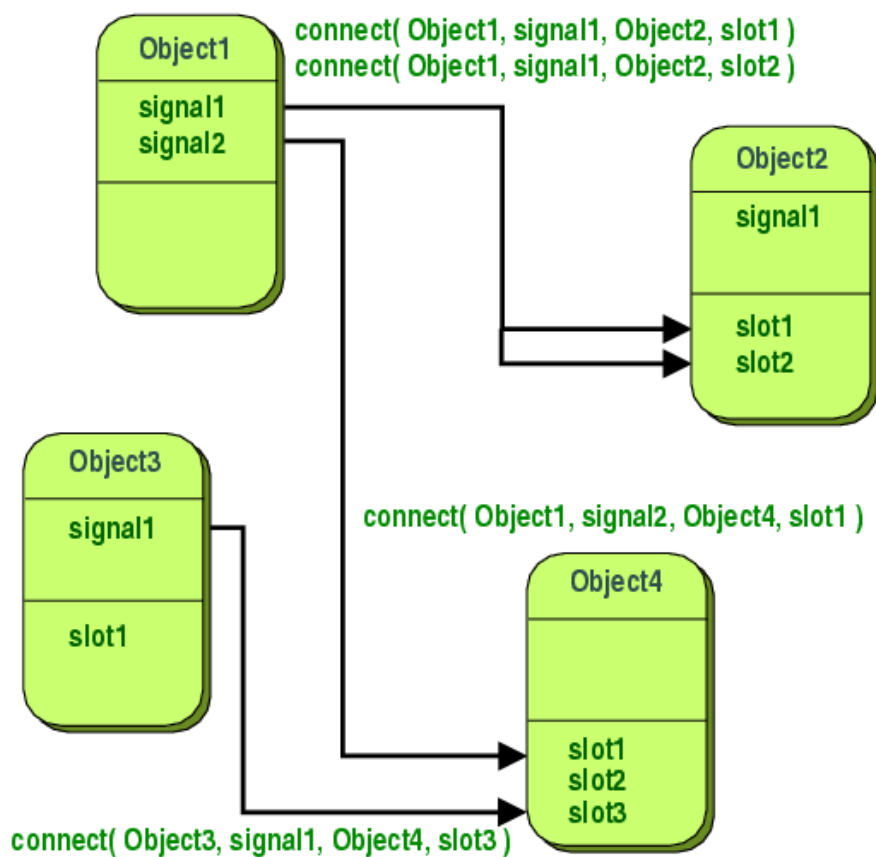
Například pro zjištění pořadového čísla aktuálně používaného desktopu stačí zavolat `get_property` s argumentem `prop_name = „_NET_CURRENT_DESKTOP“`, ukázka na obrázku 4.2.



#### 4.1.6 Qt

##### Signály a sloty

Signály a sloty [12] tvoří mechanismus (jazykovou konstrukci) některých programovacích jazyků, který se používá pro komunikaci mezi objekty a umožňuje realizovat *Observer pattern*. Koncepce je následná: jeden komponent může posílat signály, které popisují určitou událost (například stisknutí tlačítka). Jiné komponenty mohou přijímat tyto signály pomocí speciálních funkcí - slotů. Signální a slotový systém je vhodný pro popis grafického uživatelského rozhraní. Všechny třídy, které zdědí `QObject` nebo jednu z jeho podtříd (např. `QWidget`), mohou obsahovat signály a sloty (viz obrázek 4.3).



Obrázek 4.3: Signály a sloty

## 4. REALIZACE

---

Ve svém programu signály a sloty používám nejen v práci s GUI, ale i při realizaci spojení časovače a funkce *getCurrentDesktop* z předchozí sekce.

Třída *CTimeEntryList* mimo jiné obsahuje slot *detectDesktopProperty*.

```
49 public slots:
50     void detectDesktopsProperty();
51     void changeTotalDesktopsNumber();
52
53 signals:
54     void valueChanged();
55
```

Obrázek 4.4: Signál a sloty třídy *CTimeEntryList*

Řešením problémů detekce změny statusu ploch je spojení časovače a funkce *detectDesktopsProperty*.

```
17 connect(timer, SIGNAL(timeout()), this, SLOT(detectDesktopsProperty()));
```

Obrázek 4.5: Spojení časovače a slot-funkce *detectDesktopsProperty*

Jelikož *QTimer* je Qt třída (na obrázku 4.5 timer je instancí třídy *QTimer*), která dědí z *QObject* a má signál *timeout*, signál bude poslán, jakmile časovač dojde k 0, čímž se vyvolá funkce *detectDesktopsProperty*. Takový systém umožňuje kontrolovat status v určitém nastaveném časovém intervalu.

## 4.2 Napojení na Toggl API

Jelikož Toggl API jsou ve formátu HTTP požadavků a odpovědí, pro realizaci komunikace mezi klientem a Toggl serverem byla použita Qt knihovna *QtNetwork*, která právě tuto možnost nabízí. Pro implementaci této funkčnosti stačí využít tři třídy z knihovny QtNetwork: *QRequest*, *QReply* a *QNetworkAccessManager* [14], které umožňují posílat požadavky a přijímat odpovědi. V mé aplikaci odpovědnost za REST API nese třída *CHTTPReceiver*.

Jelikož každý uživatel má svůj vlastní token, bylo rozhodnuto použít email a heslo pro požadavek na autentizaci, který vrátí uživatelský token. Další požadavky, jako jsou například založení nového časového záznamu, se budou odesílat s tokenem v autentizační hlavičce.

### Autentizace

Pokud se jedná o první přihlášení uživatele, musí být zadány osobní údaje (email a heslo pro Toggl účet) pro autentizaci. Na obrázku 4.6 je ukázka implementace pomocí QtNetwork knihovny.

```

1 void HTTPReciever::authentication(const QString username, const QString password) {
2     QNetworkRequest request;
3     request.setRawHeader("Authorization", "Basic " +
4         QByteArray(QString("%1:%2").arg(username).arg(password).toUtf8()).toBase64());
5     request.setUrl(QUrl(AUTH_URL));
6     _reply = _manager->get(request);
7     connect(_reply, SIGNAL(finished()), this, SLOT(replyClientDetailsFinished()));
8 }

```

Obrázek 4.6: Příklad použití funkce authentication

Autentizační hlavička je nastavena dle pravidel popsanych v sekci 3.2.1. Požadavek na autentizaci je volán metodou **GET**. Díky tomu, že signál `finished()` a slot `replyClientDetailsFinished()` jsou spojené, jakmile bude odpověď přijata, zavolá se funkce `replyClientDetailsFinished()`, jež kontroluje, zdali byl požadavek úspěšný.

### Spuštění/zastavení časového záznamu

Podobným způsobem jako autentizace probíhá spuštění a zastavení časového záznamu, pouze s tím rozdílem, že autentizační hlavička bude obsahovat token namísto emailu a hesla. Spuštění záznamu je realizováno pomocí metody **POST**, zastavení pomocí **PUT**. Na obrázku 4.7 je příklad metody, která se tímto zabývá. Po spuštění záznamu a přijetí úspěšné odpovědi na požadavek je zjištěno identifikační číslo tohoto záznamu, které bude následně použito pro zastavení.

```

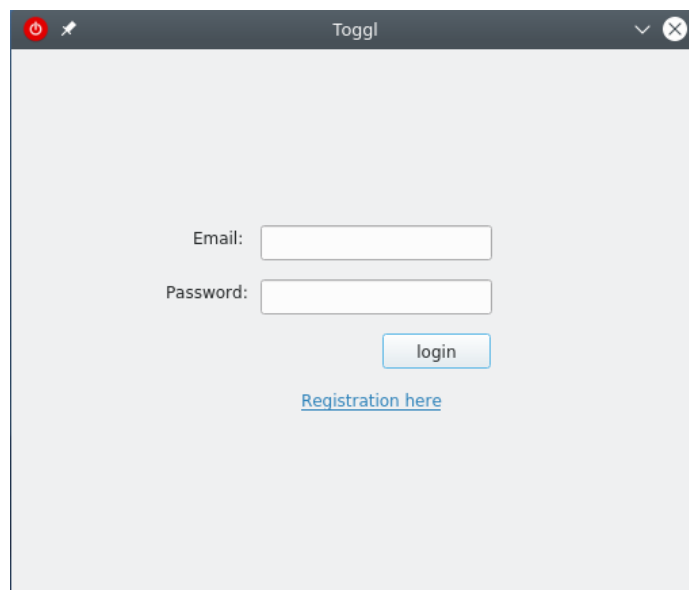
1 void HTTPReciever::sendRequest(CTimeEntryItem * entry, Action act){
2     QNetworkRequest request;
3     request.setHeader(QNetworkRequest::ContentTypeHeader, "application/json");
4     request.setRawHeader("Authorization", "Basic " +
5         QByteArray(QString("%1:%2").arg(_token).arg(API_TOKEN).toUtf8()).toBase64());
6     switch(act){
7     case Action::START: {
8         QUrl url(TIME_ENTRY_DETAILS_URL + START_ENTRY);
9         request.setUrl(url);
10        QString id = entry->getId();
11        _actual_request_body = entry->timeEntryToJSON();
12        _reply = _manager->post(request, _actual_request_body.toUtf8());
13        _item = entry;
14        connect (_reply, SIGNAL(finished()), this, SLOT(replyFinished()));
15        break;
16    }
17    case Action::STOP:{
18        QString id = entry->getId();
19        QString link = TIME_ENTRY_DETAILS_URL + "/" + id + "/stop";
20        _actual_request_body = QTime::currentTime().toString();
21        request.setUrl(QUrl(link));
22        _reply = _manager->put(request, "");
23        connect (_reply, SIGNAL(finished()), this, SLOT(replyFinished()));
24        break;
25    }
26    default: break;
27 }
28 }

```

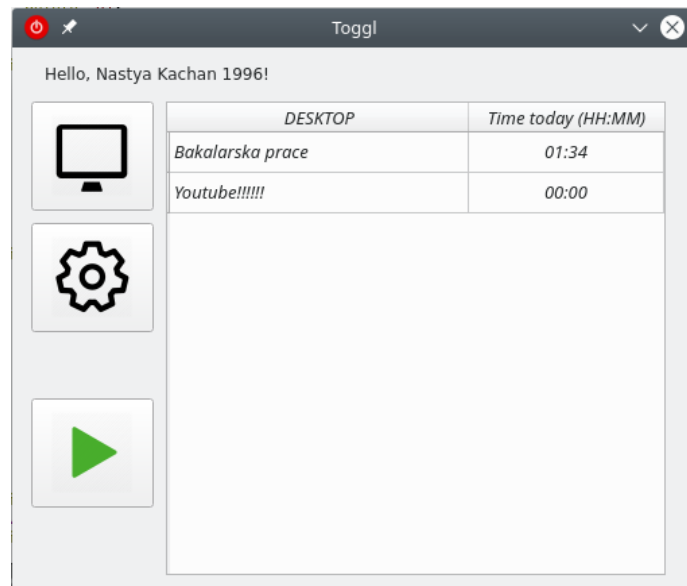
Obrázek 4.7: Příklad volání požadavku s tokenem v autentizační hlavičce

### 4.3 Výsledná aplikace

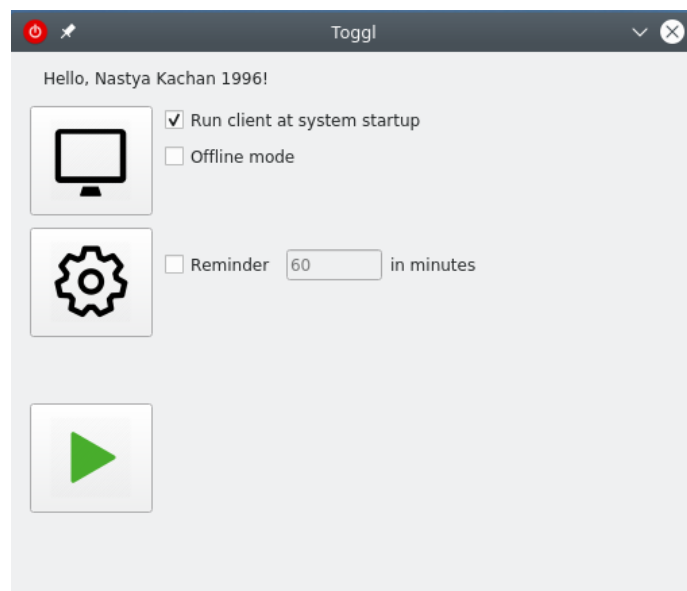
Při tvorbě svého programu jsem používala různé návody a inspirovala jsem se již existujícími aplikacemi. Pro implementaci aplikace v jazyce Qt jsem použila stránky oficiální dokumentace, kde je vše přehledně a srozumitelně popsáno, včetně příkladů aplikací, které Qt Creator nabízí. Vhodný popis interfacu knihovny Xlib jsem pak našla na neoficiálních stránkách.



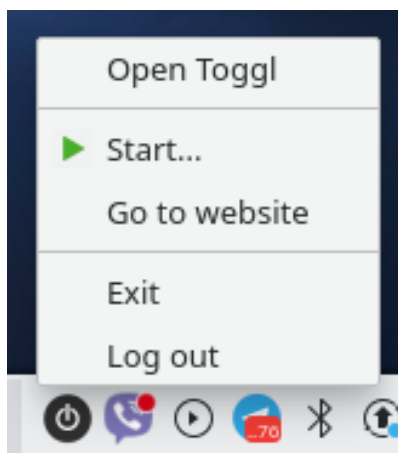
Obrázek 4.8: Okénko s přihlášením



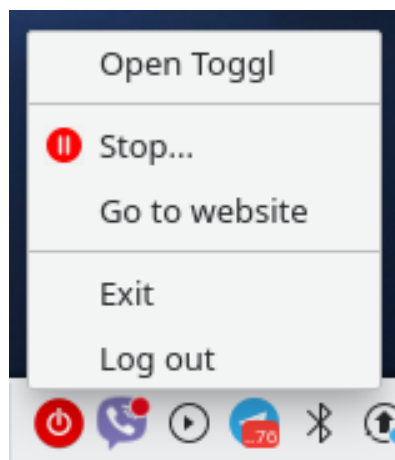
Obrázek 4.9: Zobrazení časových záznamů



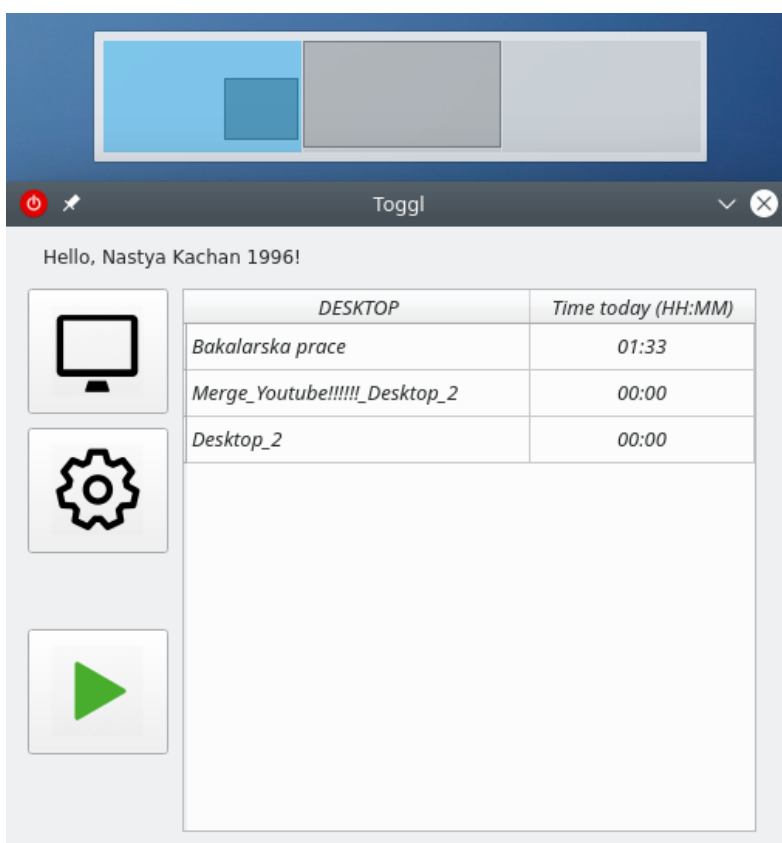
Obrázek 4.10: Nastavení



Obrázek 4.11: Tray Menu s pozastaveným měřičem



Obrázek 4.12: Tray Menu se spuštěným měřičem



Obrázek 4.13: Sloučení časových záznamů

# Testování

## 5.1 Metody testování

Výsledná aplikace byla testována několika druhy testu. Během implementace byly kód a GUI testovány unit-testy [15] metodou *white box* pomocí Qt Test Framework [16]. Závislost určité třídy na jiné se řešila pomocí *mock tříd*.

## 5.2 Testování použitelnosti

Dalším druhem testování bylo testování použitelnosti. Jedná se o kontrolu softwarového produktu z hlediska snadnosti používání aplikace. Pomocí testů použitelnosti můžeme určit ergonomii (vhodnost pro použití) programu. Kontrola použitelnosti aplikace tkví v těchto bodech:

- posouzení shody návrhu aplikace s její funkčností dle specifikace zákazníka;
- analýza použitých grafických prvků, barevný design z hlediska vnímání;
- vyhodnocení snadnosti navigace a referenční struktury;
- vyhodnocení použitelnosti aplikačních funkcí;
- analýza návrhu písma textu;
- odstranění více úkolů.

Abych zjistila, jaké části je třeba vylepšit či změnit, nebo které jsou již dobře zpracovány, vytvořila jsem scénář, jakým budou postupovat uživatelé z testovací skupiny, a dotazník. Testovací skupina byla sestavena z uživatelů různých oblastí lidské činnosti i věkových skupin.

### 5.2.1 Testovaný scénář

Aby funkčnost aplikace byla co nejlépe analyzována, byl vytvořen scénář, který pokryl všechny funkční požadavky:

1. přihlásit se pomocí testového účtu Toggl [F1];
2. prohlédnout si časové záznamy existujících ploch [F2];
3. odstranit plochu a podívat se na změny v aplikaci [F3];
4. přejmenovat libovolné záznam [F4];
5. spustit měřič [F5];
6. sloučit dva libovolně záznamy [F6];
7. při spuštění měřiči přepnout si na jinou plochu a podívat se na změny v aplikaci [F7];
8. změnit konfigurace v nastaveních [F8].

### 5.2.2 Dotazník

Po vyzkoušení zadaného scénáře byl nabídnut dotazník, v němž jednotliví uživatelé mohli odpovědět na otázky týkající se funkčnosti aplikace a ohodnotit je.

Dotazník se skládá z následujících otevřených otázek:

1. Ohodnoťte aplikaci od 1 do 10.
2. Co se Vám líbilo v aplikaci?
3. Jak jednoduché a pochopitelné bylo používání?
4. Co by se dalo zlepšit?
5. Jaká funkce Vám chyběla?



## 5.3 Zhodnocení testování použitelnosti

Vyzkoušení scénáře každým uživatelem bylo nahráváno na diktafon.

### 5.3.1 Zhodnocení průchodu scénářem

#### **Uživatel 1**

1. neměl žádné problémy se splněním úkolu,
2. neměl žádné problémy se splněním úkolu,
3. neměl žádné problémy se splněním úkolu,
4. neměl žádné problémy se splněním úkolu,
5. neměl žádné problémy se splněním úkolu,
6. nebylo hned jasné jak se má provádět sloučení časových záznamu (není zřejmě vidět, že se má provést metoda drag-and-drop),
7. neměl žádné problémy se splněním úkolu,
8. neměl žádné problémy se splněním úkolu.

#### **Uživatel 2**

1. neměl žádné problémy se splněním úkolu,
2. neměl žádné problémy se splněním úkolu,
3. neměl žádné problémy se splněním úkolu,
4. nejdříve se pokusil pravé tlačítko myši a až potom poklepat,
5. neměl žádné problémy se splněním úkolu,
6. snažil se poklepat a až potom zkusil metodu drag-and-drop,
7. neměl žádné problémy se splněním úkolu,
8. neměl žádné problémy se splněním úkolu.

#### **Uživatel 3**

1. neměl žádné problémy se splněním úkolu,
2. neměl žádné problémy se splněním úkolu,
3. neměl žádné problémy se splněním úkolu,
4. neměl žádné problémy se splněním úkolu,
5. neměl žádné problémy se splněním úkolu,
6. neměl žádné problémy se splněním úkolu,
7. neměl žádné problémy se splněním úkolu,
8. neměl žádné problémy se splněním úkolu.

### 5.3.2 Výsledky dotazníku

Testovací skupině bylo nabídnuto odpovědět na otázky dotazníku v libovolné formě (odpovědět psanou formou nebo nahrát odpovědi na diktafon). Všichni testující zvolili druhý způsob zaznamenání odpovědí. Následně tedy uvedu jejich odpovědi citované z audionahrávky:

1. Ohodnotte aplikaci od 1 do 10.
  - a) *Uživatel 1* 8.
  - b) *Uživatel 2* 7.
  - c) *Uživatel 3* 8.
2. Co se Vám líbilo v aplikaci?
  - a) *Uživatel 1* Sloučení záznamu, ovšem to nebylo jasné na první pohled.
  - b) *Uživatel 2* Synchronizace s Toggl servicem.
  - c) *Uživatel 3* Počítá čas správně, online synchronizace s Toggl servicem.
3. Jak jednoduché a pochopitelné bylo používání?
  - a) *Uživatel 1* Ohodnotil bych jednoduchost jako 9 z 10.
  - b) *Uživatel 2* Většinou ano, ale pro jiné uživatele věci jako sloučení záznamu nebo jejich úprava nemusí být tak zřejmé jako pro mě.
  - c) *Uživatel 3* Aplikace je poměrně intuitivní na používání díky jednoduchosti uživatelského rozhraní.
4. Co by se dalo zlepšit?
  - a) *Uživatel 1* Velikost hlavního okna je příliš velká, šlo by to zmenšit.
  - b) *Uživatel 2* Tlacitko „Log out“ hlavního okna a duplikace funkčnosti aplikace v minimalizovaném okně.
  - c) *Uživatel 3* Nic.
5. Jaká funkce Vám chyběla?
  - a) *Uživatel 1* Nastavení připomínky pro každý desktop zvlášť, aby šlo limitovat čas strávený na každé ploše.
  - b) *Uživatel 2* Denní statistika bez ohledu na to, že to nabízí webová aplikace.
  - c) *Uživatel 3* Počítání i sekund u měřiče.

Z odpovědí plyne, že uživatelé jsou s aplikací poměrně spokojeni, přestože postrádali některé funkce. Díky těmto odpovědím a testování použitelnosti jsem poměrně jasně schopna určit směr, kterým se ubírat při vylepšování mého programu.



---

## Závěr

Cílem této práce bylo navrhnout, implementovat a otestovat klienta služby Toggl pro desktopové prostředí KDE.

Nejdřív jsem analyzovala již existující desktopové měřiče unixových systémů a zjistila jsem, že ani jeden neumožňuje automatickou detekci změny virtuální plochy a počítání času zvlášť na každé z nich. Na základě výsledků získaných analýzou byly sestaveny jednotlivé případy užití a navrženo uživatelské rozhraní. Po navržení celé aplikace proběhl vývoj, který je popsán v implementační části této práci.

Hotová aplikace byla testována jak unit-testy, tak i testy použitelnosti. Testy použitelnosti měly konkrétní přínos ve formě definice možného směru zlepšení aplikace a nových myšlenek i inspirace pro implementaci nové funkce daného programu.

Výstupem práce je funkční desktopový tracker služby Toggl splňující definované zadání.

V budoucnu by bylo možné aplikaci dále vylepšit o nové grafické rozhraní a rozšířit její funkce.



---

## Literatura

- [1] *Toggl* [online], [cit. 2018-10-11]. Dostupné z: <https://toggl.com>
- [2] *RescueTime* [online], [cit. 2018-10-12]. Dostupné z: <https://www.rescuetime.com/>
- [3] *TopTracker* [online], [cit. 2018-10-12]. Dostupné z: <https://www.toptal.com/tracker>
- [4] *TimeCamp* [online], [cit. 2018-10-12]. Dostupné z: <https://www.timecamp.com/>
- [5] *Hubstaff* [online], [cit. 2018-10-12]. Dostupné z: [https://hubstaff.com/?ab=masthead\\_video2](https://hubstaff.com/?ab=masthead_video2)
- [6] Toggl. *Toggl API Documentation*. V: *GitHub* [online], červenec 2017, [cit. 2019-03-03]. Dostupné z: [https://github.com/toggl/toggl\\_api\\_docs](https://github.com/toggl/toggl_api_docs)
- [7] Authentication. *Toggl API Documentation*. V: *GitHub* [online], červenec 2017, [cit. 2019-03-03]. Dostupné z: [https://github.com/toggl/toggl\\_api\\_docs/blob/master/chapters/authentication.md](https://github.com/toggl/toggl_api_docs/blob/master/chapters/authentication.md)
- [8] Klyne G.: Date and Time on the Internet. *Timestamps* [online], červenec 2002, [cit. 2019-03-11]. Dostupné z: <https://www.ietf.org/rfc/rfc3339.txt>
- [9] Time Entries. *Toggl API Documentation*. V: *GitHub* [online], červenec 2017, [cit. 2019-03-03]. Dostupné z: [https://github.com/toggl/toggl\\_api\\_docs/blob/master/chapters/time\\_entries.md](https://github.com/toggl/toggl_api_docs/blob/master/chapters/time_entries.md)
- [10] Gettys J.: Xlib - C Language X Interface. *X Consortium Standard* [online], 2002, [cit. 2019-02-07]. Dostupné z: [ftp://www.x.org/pub/current/doc/libX11/libX11/libX11.html#Introduction\\_to\\_Xlib](ftp://www.x.org/pub/current/doc/libX11/libX11/libX11.html#Introduction_to_Xlib)

- [11] Christophe Tronche: The X Window System: *The Xlib Manual* [online], [cit. 2019-02-04]. Dostupné z: <https://tronche.com/gui/x/xlib>
- [12] Signals & Slots. *Qt Documentation* [online], [cit. 2019-02-04]. Dostupné z: <https://doc.qt.io/qt-5/signalsandslots.html>
- [13] Qt Documentation Archives: Using Drag and Drop with Item Views. *Qt Documentation* [online], [cit. 2019-03-08]. Dostupné z: [https://doc.qt.io/archives/qtjambi-4.5.2\\_01/com/trolltech/qt/model-view-dnd.html](https://doc.qt.io/archives/qtjambi-4.5.2_01/com/trolltech/qt/model-view-dnd.html)
- [14] QNetworkAccessManager Class. *Qt Documentation* [online], [cit. 2019-01-29]. Dostupné z: <https://doc.qt.io/qt-5/qnetworkaccessmanager.html>
- [15] Coppola D.: Bits of Bytes. *GUI unit testing with Qt Test – part 1 – introduction* [online], leden 2017, [cit. 2019-02-04]. Dostupné z: <http://blog.davidecoppola.com/2018/01/gui-unit-testing-with-qt-test-introduction/>
- [16] Qt Test Overview. *Qt Documentation* [online], [cit. 2019-04-11]. Dostupné z: <https://doc.qt.io/qt-5/qtest-overview.html>



## Seznam použitých zkratek

- GUI** Graphical user interface
- XML** Extensible markup language
- API** Application programming interface
- REST** Representational State Transfer
- HTTP** HyperText Transfer Protocol
- MIME** Multipurpose Internet Mail Extensions
- JSON** JavaScript Object Notation
- UI** User interface
- IDE** Integrated Development Environment
- HTML** HyperText Markup Language



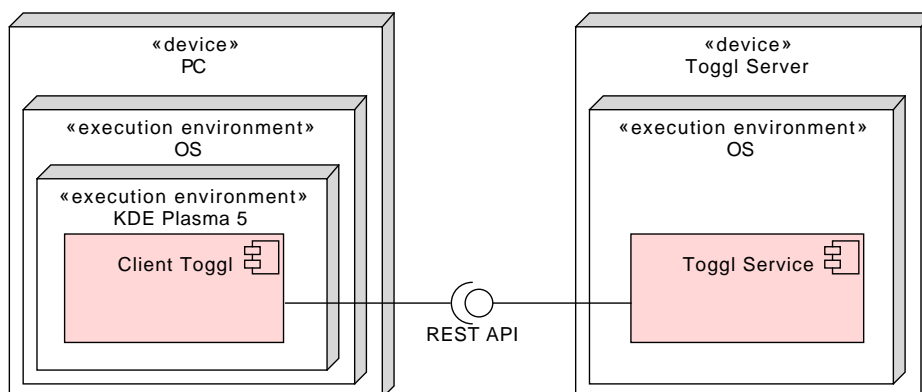
# Instalační příručka

## B.1 Diagram nasazení a závislosti

Aplikace je určena pro desktopové prostředí KDE Plasma, přičemž v okamžiku implementace bylo použito prostředí KDE Plasma 5 ve verzi 5.15.4. Minimální požadovaná verze KDE je 5.8. Při implementaci aplikace bylo použito Qt5 verze 5.12.2.

Kde Plasma 5 již obsahuje Qt5. Dále je nutné, aby byly nainstalovány následující balíčky: *libqt5x11extras5-dev*, *xorg-dev* a *libglib2.0-dev*. Pro jednoduchost jsem vložila nutné balíčky do *installation.sh* skriptu, který je po spuštění nainstaluje a zkompile zdrojový kód. Podrobná instrukce se nachází v README.md souboru.

Na obrázku B.1 je diagram nasazení, kde je vidět propojení se serverem služby Toggl. Diagram byl vytvořen pomocí nástroje **UMLet**.



Obrázek B.1: Diagram nasazení



## Obsah přiloženého USB

README.txt.....	stručný popis obsahu USB a návod na instalaci
installation.sh .....	instalační soubor
bin .....	adresář se spustitelnou formou implementace
src.....	zdrojové kódy implementace
├─ images.....	obrázky ikoněk
doc	
├─ html .....	dokumentace
├─ Doxyfile.....	konfigurační soubor na vytváření dokumentace
unit_tests	
├─ src.....	zdrojové kódy testů
├─ bin.....	adresář se spustitelnou formou testů
├─ run_tests.sh	
usability_tests .....	audionahrávky testování použitelnosti
thesis .....	zdrojová forma práce ve formátu L <sup>A</sup> T <sub>E</sub> X
text .....	text práce
├─ thesis.pdf .....	text práce ve formátu PDF
├─ thesis.ps .....	text práce ve formátu PS