Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačních technologií

Diplomová práce

Automatický time tracking - implementační studie

Michal Bartoš

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Bc. Michal Bartoš

Systémové inženýrství a informatika
Informatika

Název práce

Automatický time tracking – implementační studie

Název anglicky

Automatic time tracking – implementation study

Cíle práce

Cílem práce je návrh a následné ověření řešení pro automatické sledování času aktivit pracovníků v oblasti profesionálních služeb, sběr dat včetně návazných analytických nástrojů a technik, které umožní snadné zaúčtování času na projekty, projektové úkoly, zákazníky apod.

Dílčí cíle:
- time tracking, časové řady a analytické metody vyhodnocování dat
- alternativní řešení a jejich hodnocení
- návrh architektury řešení a konkrétní implementace na bázi open-source
- vytvoření prototypu systému sběru dat z OS Windows a Linux
- integrace s existující aplikací pro time tracking
- návrh analytických postupů pro automatické zaúčtování většiny času
- výsledky a diskuze, formulace závěrů práce

Metodika

Time tracking a možnosti jeho využití. Vysvětlení způsobu a metody práce s časovými řadami, analytické techniky používané ve vyhodnocování dat. Přehled a hodnocení alternativních řešení. Návrh architektury řešení a návrh konkrétní implementace s využitím existujících open-source komponent. Sběr dat a analýza časových řad. Návrh prototypu systému sběru dat včetně návazných analytických nástrojů a technik, které v dané případové studii umožní automatické zaúčtování většiny času na předdefinované úlohy v projektovém systému. Předpokládá se sběr aktivity z operačních systémů Windows a Linux a vybraných aplikací, detekce stavu „away from keyboard“ a sběr hodnot z vybraného IoT senzoru – pohyb nebo identifikace přítomnosti dané osoby v místnosti.

Doporučený rozsah práce

50 – 60 stran

Klíčová slova

Sledování času, výkazy, monitoring, analýza dat, časové řady, projektový systém, automatické měření času

---

Doporučené zdroje informací

ActivityWatch – Open-source time tracker [online]. 2021. Dostupné z: https://activitywatch.net/

GitHub – ActivityWatch/aw-research: Tools to analyse and experiment with ActivityWatch data [online]. 2021. Dostupné z: https://github.com/ActivityWatch/aw- research

MARTINEZ, Juan a Brian T. HOROWITZ. The Best Time Tracking Software for 2021 | PCMag [online]. 2020. Dostupné z: https://www.pcmag.com/picks/the-best- time-tracking-software

Redmine [online]. 2021. Dostupné z: https://www.redmine.org/

SIMMONS, David G. Pushing IoT Data Gathering, Analysis, and Response to the Edge – DZone IoT [online]. 2018. Dostupné z: https://dzone.com/articles/pushing-iot-data-gathering-analysis-and-response-to-the-edge

---

Předběžný termín obhajoby

2022/23 ZS – PEF

Vedoucí práce

doc. Ing. Jiří Vaněk, Ph.D.

Garantující pracoviště

Katedra informačních technologií

Elektronicky schváleno dne 7. 7. 2021

doc. Ing. Jiří Vaněk, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 19. 10. 2021

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 01. 02. 2023

Čestné prohlášení

Prohlašuji, že svou diplomovou práci „Automatický time tracking - implementační studie“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 30. března 2023 _____

Poděkování

Rád bych touto cestou poděkoval doc. Ing. Jiřímu Vaňkovi, Ph.D. za odborné rady a připomínky, které mi poskytl v průběhu zpracování této práce. Chtěl bych také poděkovat kolegům ze společnosti Orchitech Solutions, s.r.o. za všechny cenné rady, které mi v průběhu zpracování této práce poskytli. Dále děkuji své rodině a přítelkyni za podporu během studia.

Automatický time tracking - implementační studie

Abstrakt

Diplomová práce je zaměřena na návrh řešení a ověření tohoto řešení pro automatické sledování času ve firmě Orchitech Solutions, s.r.o. V práci jsou vydefinovány základní pojmy týkající se sledování a vykazování času, dostupných nástrojů a podpůrných prostředků. Pro vybrané aplikace je vytvořen návrh architektury řešení, který byl uveden do provozu. Získaná data z provozu jsou analyzována, díky čemuž je výsledné řešení automatického sledování času vylepšeno tak, aby odpovídalo potřebám a požadavkům pracovníků firmy.

Klíčová slova: Sledování času, výkazy, monitoring, analýza dat, časové řady, projektový systém, automatické měření času

Automatic time tracking - implementation study

Abstract

The diploma thesis is focused on designing a solution and verifying it for automatic time tracking at Orchitech Solutions, s.r.o. The thesis defines basic concepts related to time tracking and reporting, available tools, and support resources. A solution architecture design has been created for selected applications, which has been put into operation. The data obtained from the operation is analyzed, which improves the resulting solution for automatic time tracking to meet the needs and requirements of the company's employees.

Key words: Time tracking, reports, monitoring, data analysis, time series, project system, automatic time tracking

Obsah

1 Úvod

Výsledkem naší práce může být celá škála podkladů, které je možné hodnotit různými kritérii. Podklad jako počet řádků či přidaných komentářů v IT odvětví pozbývá smyslu, protože ani jedno z těchto kritérií není zárukou kvality a efektivity výsledného řešení. Vývoj software bývá specifický tím, že mezi nedílnou součást pracovní náplně celého týmu se řadí také měření a vykazování času stráveného na jednotlivých úlohách. V IT firmách bývá typické, že se současně vyvíjí více produktů a poskytují se služby vícero zákazníkům, a to i v průběhu jednoho dne. Z tohoto důvodu je nutné, aby si každý zaměstnanec vedl evidenci odpracovaného času, například v projektovém systému. Tato mnohdy neoblíbená činnost je velice důležitá nejen pro správné zaúčtování vykonané práce zákazníkovi, ale také pro efektivní rozvržení kapacit týmu na jednotlivých projektech.

Pravidelné a průběžné vykazování k úlohám poskytuje projektovému manažerovi či managementu lepší přehled o aktuálním stavu projektů. Umožní tak včasné odhalení případných alokačních konfliktů nebo nereálné datum pro předání implementovaných změn zákazníkovi. Někteří zákazníci mohou požadovat detailní specifikaci každé vykonané práce, aby měli jasný přehled o tom, za co a jak efektivně byly jejich finance vynaloženy. Průběžné zaznamenávání odvedené práce je klíčové, jelikož se obtížně dohledává několik dnů či týdnů zpětně, na čem daný zaměstnanec pracoval a kolik času na dané úloze strávil. S narůstajícím časem, ve kterém zaměstnanec zpětně dohledává potřebné informace, roste riziko zkreslení informací o čase, ale také náplni a charakteristice vykonané práce. V důsledku nepřesných informací se taktéž zkreslují přehledy o tom, jak časově náročná daná úloha byla a jak kapacitně náročné je možné očekávat podobné úlohy.

Pro zpětné dohledávání je možné použít historii prohlížeče, historii ve verzovacím systému, např. GitLab nebo historii v projektovém systému. Pokud bude mít zaměstnanec štěstí, nějakou informaci získá, avšak tato informace s největší pravděpodobností již bude zkreslená a neposkytne zaměstnanci konkrétní a přesný přehled toho, na čem a jak dlouho pracoval. Nedostatečná vykázanost nebo podhodnocené výkazy mohou v zaměstnanci evokovat pocit nedostatečného množství času, které věnoval pracovním činnostem, i přesto, že v kanceláři strávil více než stanovenou pracovní dobu. Tento neutuchající pocit může vést až k tomu, že zaměstnanec bude pracovat nad rámec své pracovní doby, což sebou může nést různé následky, od nechuti až po možné vyhoření jedince.

Na trhu v dnešní době existuje mnoho řešení na sledování času, avšak mnohé vyžadují manuální zásah pro spuštění a ukončení záznamu činnosti, také existuje řada řešení na automatické sledování času. Již existující nástroje umí zobrazit, kolik času

zaměstnanec strávil danou činností, ale žádný z nich neumí automaticky stanovit a přehledně zobrazit, kolik času a k jaké úloze se má zákazníkovi naúčtovat. Podklady, které budou poskytovat tato data a poskytnou tak zaměstnanci informaci o stráveném čase, jsou výstupem této práce. Tento přehled sledovaného času navíc nemusí uživatel používat pouze pro zpětné dohledávání záznamu činností, ale může sloužit také pro bližší pochopení toho, jak se uživatel na počítači chová a jak jej využívá.

2 Cíl práce a metodika

2.1 Cíl práce

Cílem práce je návrh a následné ověření řešení pro automatické sledování času aktivit pracovníků v oblasti profesionálních služeb, sběr dat včetně návazných analytických nástrojů a technik, které umožní snadné zaúčtování času na projekty, projektové úkoly, zákazníky apod. Vznikne návrh architektury řešení a návrh konkrétní implementace. Implementace by měla využívat existující open-source komponenty. Dále bude vytvořen prototyp systému sběru dat na základě případové studie zadané konzultantem ze zvolené reálné společnosti. Předpokládá se sběr aktivity z operačních systémů Windows a Linux a vybraných aplikací, detekce stavu „away from keyboard“ a sběr hodnot z vybraného IoT senzoru – pohyb nebo identifikace přítomnosti dané osoby v místnosti. Systém bude integrován s existující aplikací pro zaznamenávání času (start-stop systém). Dále vzniknou návrhy analytických postupů, které v dané případové studii umožní automaticky zaúčtovat většinu času na předdefinované úlohy v projektovém systému.

2.2 Metodika

Práce bude rozdělena na dvě hlavní části. Teoretická část práce bude založena na shromáždění, studiu a analýze dostupných odborných a vědeckých informačních zdrojů z oblasti sledování a vykazování času. V teoretické části proběhne explorace přístupů ke sledování času, průzkum alternativních řešení a jejich specifikace. Bude vysvětlen pojem časová řada a bude diferenciován od pojmu časová osa, jenž se využívá k rekonstrukci dat. Praktická část bude založena na analýze dostupných nástrojů a zjištění současného stavu v konkrétní společnosti. Na základě zjištěných informací proběhne navržení a implementace prototypu systému, který bude uveden do provozu. Získaná data z provozu budou analyzována a výstupy budou syntetizovány do prototypu.

3 Teoretická východiska

3.1 Sledování a vykazování času

Žádný manažer, zaměstnanec nebo dodavatel nežije mimo čas. Čas je i nadále univerzální měnou pro oceňování, organizování, plánování a řízení práce. Bez sledování našeho času není možné měřit nebo řídit produktivitu, ziskovost a efektivitu. Nevidíme, kde zaostáváme, kde nám nejvíce ubývá čas nebo které projekty jsou pro nás ztrátové.

3.1.1 Přínos

Zlepšení projektového managementu

Z projektového hlediska potřebujeme vědět, na čem tým pracuje, kdo má volnou kapacitu, kdo je přetížený, kdo je zodpovědný za určité úkoly a je zapotřebí znát priority. Nedodstatečná znalost aktuální práce v týmu vede k mnohem náročnějšímu dodržování termínů, neefektivnímu plánování, snížení efektivity procesů a náročnějšímu řízení nadměrné spolupráce se zákazníkem. (Timely, 2022a)

Možnost vidět veškerou práci, která je součástí každého projektu, znamená, že je možné provádět přesné odhady budoucí práce a upřesňovat návrhy a plány projektů. Pochopení toho, které zdroje budou potřeba, poskytuje možnost plynulejšímu delegování. Znalosti a informace o tom, jak dlouho trvají určité typy úkolů, umožní proaktivně odhalit a vyřešit jakékoli problémy nebo nedostatky dříve, než se vyčerpá plánovaný rozsah, ohrozí se datum dodávky nebo se projekt prodraží.

Produktivita práce

Není možné být produktivnější, aniž by si člověk nejprve uvědomil neefektivitu svého pracovního postupu. Sledování času umožňuje identifikovat největší časové ztráty, přerušení a rozptýlení a také to, které úkoly jsou nejnáročnější. Umožňuje také identifikovat nevědomé chování a zjistit, kdy je člověk nejproduktivnější. (Timely, 2022d)

Zamezení přepracování

Sledování času umožňuje zaměstnancům dodržovat smluvenou dobu a zajistit, aby udržovali zdravý rozvrh. Člen týmu pracující z domova je obzvláště ohrožen vyhořením, takže jasný záznam o tom, kolik hodin každý týden odpracoval, napomáhá dodržování hranic. Sledování času zaznamenává vše, dokumentuje tak veškeré přesčasy, a to poskytuje příležitost zaměstnancům odpovídajícím způsobem kompenzovat volno a dává manažerům vědět, že některé pracovní vytížení nemusí být reálné. (Timely, 2022d)

Zaměstnavatel by měl mít zájem na tom, aby pomáhal zaměstnancům udržet zdravé hranice mezi prací a osobním životem. Měl by porozumět tomu, co zaměstanci dělají - je důležité vědět, zda je nastavená pracovní zátěž realistická a přiměřená nebo zda je zaměstnanec přetížený.

Viditelost práce

Pokud zaměstnavatel nebo manažer ví, na čem zaměstnanec pracuje, dokáže více ocenit hodnotu, kterou zaměstnanec společnosti přidává. Je schopen vidět, kdy jednotlivci potřebují novou výzvu, zda mají dostatek rozmanitosti ve své práci, aby byla udržena jejich stimulace nebo se rozpoznalo, kdy je připraven převzít vedoucí roli. Viditelnosti práce může být dosaženo právě sledováním času na projektech a úkolech. (Timely, 2022a)

Zvýšení ziskovosti

Přesné sledování času stráveného na projektech zajišťuje, že jsou zaúčtovány všechny interní náklady a že nedojde k přehlédnutí žádných skrytých fakturovatelných hodin. Jakmile je známá doba trvání určitých úkolů, je možné měřit ziskovost projektu, nastavit konkurenceschopné sazby, zvýšit efektivitu pracovních postupů a zlepšit budoucí výdaje z rozpočtu klienta. (Timely, 2022a)

3.1.2 Fakturovatelný vs nefakturovatelný čas

Společnosti, podnikatelé, agentury nebo nezávislí pracovníci obvykle účtují za své služby hodinovou sazbu. Aby bylo možné takto účtovat podle odpracovaného času, je nutné umět sledovat množství času, které se každý den stráví na projektech daného klienta. (Timely, 2018)

Fakturovatelný čas

Fakturovatelný čas reprezentuje veškerou práci, která může být účtovaná klientovi. V závislosti na poskytovaných službách dané společnosti se mohou lišit úkoly představující fakturovatelný čas. Mezi tyto nejběžnější úkoly se zahrnuje:

- projektové plánování,
- provádění projektových prací,
- analýza a výzkum,
- projektové schůzky,

- pracovní e-maily,
- zpracování změnových požadavků klienta. (FreshBooks, 2019)

Nefakturovatelný čas

Nefakturovatelný čas zahrnuje všechny pracovní úkoly, které nejsou fakturovatelné klientovi. Mezi nejběžnější úkoly, které nejsou klientovi naúčtovány, patří:

- vypracování návrhů a nabídek pro nové klienty,
- obchodní strategie a výzkum,
- rozvoj a školení zaměstnanců,
- vývoj na interních projektech,
- oprava vlastních chyb,
- administrativní úkony, jako jsou výkazy nebo fakturace. (Timely, 2018)

Důležitost sledování nefakturovatelného času

Důležitost sledování nefakturovaného času nelze podceňovat. Zaměstnanec, který sleduje pouze fakturovatelný čas, zaznamenává pouze část své práce. Podnikání bez znalosti nefakturovaného času obvykle neběží hladce nebo se nerozvíjí. Na rozdíl od fakturovatelného času, jenž ukazuje okamžité příjmy, nefakturovatelný čas pomáhá týmům stávat se výkonnějšími, efektivnějšími a ziskovějšími. (Timely, 2018)

Holistický dohled nad skutečnou činností týmu dává možnost udržet veškeré úsilí jednotlivých členů týmu viditelné a zodpovědné, což je užitečné zejména pro menší týmy s omezenými zdroji. (Timely, 2022d)

Etika sledování času

Sledování času, a s tím související sledování zaměstnanců, obvykle bývá etické, ale také existují programy nebo přístupy zcela neetické.

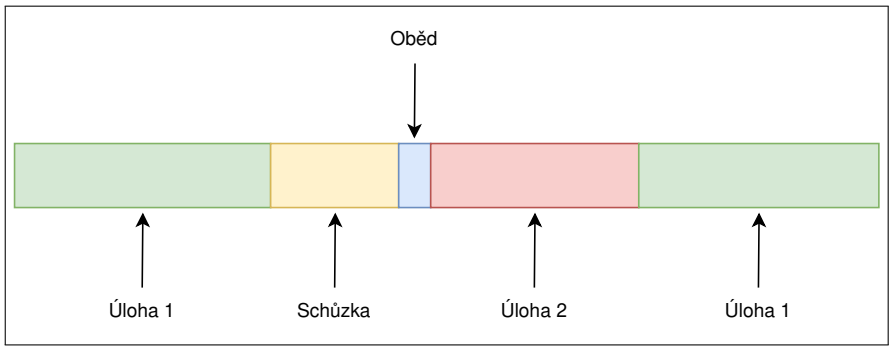
Základem etického přístupu je souhlas zaměstnanců se sledováním a sdílením svých dat, stejně tak jako přístup k těmto datům. Zaměstnanec má právo vědět, proč jsou tato data shromažďována a jak jsou nebo mohou být využita. Sledování času by mělo v zaměstnanci vzbuzovat pocit, že se tím podporuje jeho vlastní zájem a zájem zaměstnavatele. Využívané systémy, které sledují aktivitu, musí dbát na soukromí zaměstnanců i firmy. (Timely, 2022d)

Naopak neetický přístup nejčastěji zahrnuje funkci dalšího monitorování zaměstnanců, jelikož používá pro sledování času a aktivity také skryté metody. Příkladem je sledování pohybů myši, sledování veškeré internetové aktivity nebo vzdálený přístup k webové kameře počítače.

Výkonné týmy jsou založeny na důvěře a transparentnosti, proto by zaměstnanci měli mít plnou kontrolu nad svým soukromím a měli by mít možnost si vybrat, která data chtějí sdílet a která mají zůstat soukromá na úrovni uživatele. (Timely, 2017)

3.1.3 Rekonstrukce dne

Jednotlivé činnosti v průběhu pracovního dne je možné znázornit prostřednictvím specifických částí v rámci časové osy. Posloupnost jednotlivých činností v průběhu jednoho dne může být reflektována časovou osou. Závisí na pozorované skutečnosti, jelikož v daný čas je možné sledovat hned několik současných událostí. Na obrázku 1 je znázorněn jednoduchý příklad, jak je možné do časové osy události promítnout. Jedná se o detailní rozbor či analýzu událostí, které se odehrávají v průběhu dne.



Obrázek 1: Rekonstrukce pracovního dne - vlastní zpracování

3.2 Časové řady

Časová řada je soubor datových bodů indexovaných v časovém pořadí. Je to sekvence zachycující po sobě jdoucí rovnoměrně rozložené body v čase a lze ji použít ke sledování a analýze trendů v čase. (InfluxData, 2023) Časové řady se používají v mnoha oborech, jako je ekonomie, finance, věda a inženýrství, protože mohou poskytnout velké množství informací o daném systému. (Hayes, 2022)

Statistickou řadou je možné označit posloupnost znaků, jenž jsou určitým způsobem uspořádány. V případě, že je toto uspořádání realizováno na základě časového sledu hodnot znaku, jedná se o řadu časovou. (Kladivo, 2013) Srovnatelnost časových řad je zajištěna při dodržení srovnatelných základů. To znamená, že pro srovnání je zvolena stejná délka úseku, stejné organizační uspořádání či stejná volba okamžiku. (Burda, 2010)

Časové řady se nejčastěji dělí do dvou hlavních kategorií, a to intervalové a okamžikové řady. *Intervalové* jsou řady, které se vztahují k určitému období, naopak *okamžikové* se vztahují k určitému okamžiku nebo například datu. (Burda, 2010)

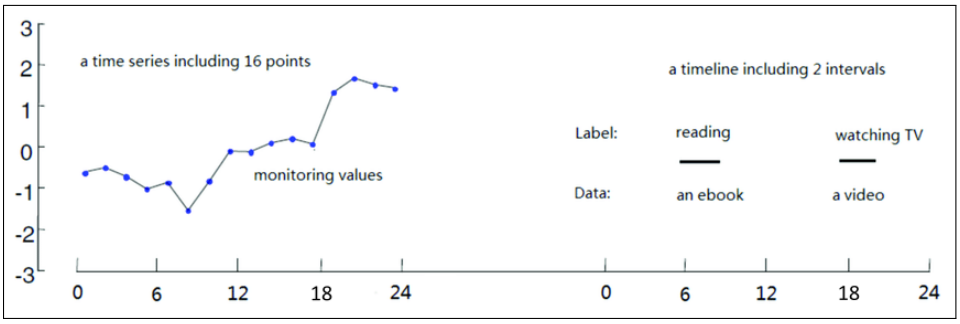
Data z časových řad mohou mít různou charakteristiku, jejíž znalost je důležitá při modelování dat časových řad. První z nich je *trend dat*, to znamená jisté snižování nebo zvyšování hodnoty v čase. Trend je nutno zohlednit v modelu při předpovědi časových řad. Druhou charakteristikou je *sezónní vliv*, což jsou vzorce, které se vyskytují v datech periodicky, například měsíčně či ročně. Na tyto vzorce mají vliv sezónní faktory, jako jsou svátky či změny počasí, proto je nutné i tuto charakteristiku zahrnout při přípravě prognostického modelu. V neposlední řadě existují nepravidelné či náhodně se vyskytující vlivy, což ovlivňuje přesnost předpovědi. Proto je nutné také s *nahodilými vlivy* počítat při předpovědi vývoje. (Shumway et al., 2017)

Časová osa

Časové osy, stejně tak jako časové řady na základě znázorněných datových bodů, poskytují vizuální obraz o událostech a vztazích v průběhu času. (IBM, 2023) Jsou mimořádně užitečné pro identifikaci vzorců, zjišťování trendů a případném vytváření předpovědí, protože mohou poskytnout přehled o průběhu událostí. Právě možnost identifikace klíčových událostí a sekvencí z časových os vytváří mocný nástroj umožňující zobrazit a analyzovat data související s časem. (BetterEvaluation, 2023)

Časové řady a časové osy jsou řazeny mezi základní nástroje pro analýzu dat a rozhodování. Využívají se k reprezentaci konkrétní události nebo procesu jako celku. Dohromady mohou být použity k pochopení minulosti, přítomnosti a plánování budoucnosti. (InfluxData, 2023)

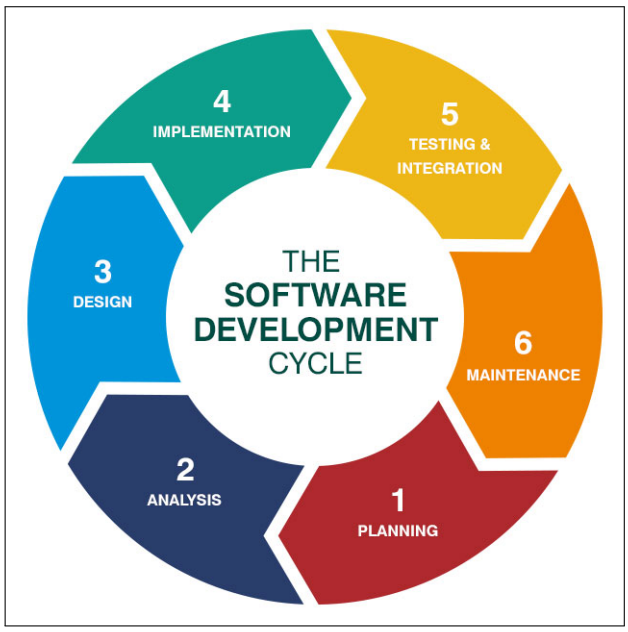
Rozdíl mezi časovou řadou a časovou osou spočívá v tom, že časová řada je číselná *sekvence datových bodů* shromážděných v pravidelných intervalech za určité časové období. Tyto hodnoty jsou spojitými kvantitativními údaji. Časová osa obvykle bývá diskrétní a je vizuální reprezentací *sady intervalových objektů*, tedy posloupností událostí. Časová osa tedy zobrazuje velikost nebo intenzitu určitého jevu, zatímco časová osa znázorňuje pořadí událostí. (He et al., 2018) Vzhledem k povaze diplomové práce a datům, které lze v souvislosti s time trackingem získat, je uvažována primárně časová osa.



Obrázek 2: Rozdíl mezi časovou řadou a časovou osou - upraveno (He et al., 2018)

3.3 Projektové systémy

Řešení pro sledování *issues* a projektů, tedy nástrojů pro projektové řízení, existuje celá řada. Základ mají společný, a to poskytovat manažerům, vývojářům a zákazníkům podporu pro organizaci práce, prioritizaci, sledování vývojových požadavků nebo chyb v průběhu celého životního (či vývojového) cyklu softwaru. Tento cyklus je znázorněn na obrázku 3.



Obrázek 3: Vývojový cyklus softwaru (BWC, 2019)

Jednotlivá řešení se liší zejména cenou, plány předplatného a služeb nebo nabízenými funkcemi. Různé internetové zdroje tak dohromady zmiňují rozmanitý přehled nástrojů. Například web forbes.com mezi nejlepší software pro řízení projektů u malých firem pro rok 2022 uvádí monday.com, Airtable, ClickUp, Notion nebo Confluence. (Rudder et al., 2022)

Web brainhub.eu pak zmiňuje jiný výčet šesti nejlepších systémů pro sledování *issues* v roce 2022, a to nástroj Jira, Backlog, Trac, Redmine, WebIssues a Asana. (Warcholinski, 2022)

3.3.1 JIRA

Nástroj pro řízení projektů Jira od Atlassianu nabízí rozsáhlé možnosti sledování *issues*. Jedná se o komplexní nástroj, který nejen umožňuje sledovat vývojové požadavky a chyby, ale současně zaujímá místo nástroje pro správu různých typů požadavků, testovacích případů až po agilní vývoj softwaru. (Atlassian, 2022b)

Jira uživatelům poskytuje například prostor pro:

- správu požadavků,
- přizpůsobení vlastních pracovních postupů (tzv. workflows),
- odhady a sledování stráveného času,
- sledování pokroku,
- využití Scrum nebo Kanban tabule,
- nastavení oprávnění na úrovni projektu,
- integrace CI/CD,
- backlog a roadmapy (časové osy) projektu.

Jiru je možné nakonfigurovat tak, aby vyhovovala jakémukoliv typu projektu a byla přizpůsobena na míru každému případu použití. Toho je možné dosáhnout nejen díky šablonám, ale také celou řadou doplňků, kterými je možné tento nástroj rozšířit. (Atlassian, 2022a)

3.3.2 Redmine

Redmine je webová aplikace pro správu projektů či projektové řízení. Je napsána pomocí frameworku Ruby on Rails, je multiplatformní a schopna pracovat s různými databázemi. Jedná se o open source aplikaci, která je vydaná za podmínek GNU General Public License v2 (GPL). Zdrojový kód aplikace je nejen volně dostupný, ale je možné jej upravovat a dále šířit nebo používat v programech pod touto licencí.

Mezi hlavní funkce, které Redmine poskytuje, patří:

- flexibilní systém sledování *issues*,
- flexibilní řízení přístupu založené na rolích a právech uživatelů,

- schopnost podpory pro více projektů,
- sledování stráveného času,
- Ganttův diagram a kalendář,
- wiki stránka či fórum pro každý projekt,
- definice vlastních polí pro *issues*, projekty a uživatele,
- podpora vícenásobné LDAP autentizace,
- integrace s nástroji, které umožňují správu a verzování zdrojových kódů (např. SVN, CVS, Git, ...),
- správa novinek, dokumentů a souborů. (Lang et al., 2022)

Pro Redmine existuje velké množství, v dnešní době již přes tisíc, existujících pluginů. Ty uživatelům poskytují podporu pro další rozšíření, a tím nabízejí možnost vytvořit z Redmine ještě komplexnější nástroj pro projektové řízení. (Redmine, 2022)

3.4 Přístupy ke sledování času

K měření času lze přistupovat několika způsoby. Na trhu dnes existuje mnoho dostupných řešení, které jsou poskytovány s open-source nebo komerční licencí.

Obecně lze přístupy k měření času rozdělit na dvě skupiny - nástroje měřící čas manuálně a nástroje měřící čas automaticky. Nástroje měřící čas manuálně si lze představit jako stopky, uživatel začíná a ukončuje měření obvykle pomocí tlačítka. Nástroje měřící čas automaticky v zásadě sledují aktivitu uživatele při práci na počítači, na základě toho dokáží určit čas strávený danou aktivitou. U všech nástrojů, které nenabízejí integraci s požadovanými systémy je nutné důkladně zhodnotit poskytované API a jeho možnosti pro vytvoření vhodné a udržitelné vlastní integrace. (Timely, 2020) Níže jsou blíže představeny vybrané metody, a to metoda tužka - papír, tabulky, přepínací kostky a softwarová řešení.

3.4.1 Metoda tužka - papír

Tato čistě manuální metoda vyžaduje pouze tužku, papír a stopky (nebo jiné zařízení, které umožňuje funkci stopek, například telefon). Postup je zcela jednoduchý, uživatel spouští časovač ve chvíli, kdy začne pracovat na úkolu a zastaví jej, když s úkolem skončí. Získané výsledky si poté zapíše na papír.

Problém s tímto přístupem je, že papír je možné jednoduše ztratit a poté se jedinec musí spoléhat pouze na svou paměť, kdy musí retrospektivně projít den a vzpomínat

na to, na čem během dne pracoval. Současně vzhledem k použité metodě nemá získané výsledky digitálně. Přesnost záznamu závisí na tom, zda nezapomene během práce na jiné úloze nebo během pauzy zastavit stopky nebo naopak je nezapomene znovu spustit, když se vrátí k dané úloze. (Timely, 2022d)

3.4.2 Metoda tabulky

Metoda tabulky je o úroveň digitálně pokročilejší manuální přístup na rozdíl od metody tužka - papír. Místo zapisování časů na papír jsou informace zapsány do tabulkového editoru, např. Excelu nebo Google tabulek, často se používají určité šablony tabulek. Metoda poskytuje větší jistotu před ztrátou dat, a pokud jsou používány k výpočtům vestavěné vzorce, existuje menší pravděpodobnost, že dojde k chybě. Digitálně zaznamenané výsledky je možné jednoduše dále sdílet s klienty či kolegy.

Jelikož se jedná stále o metodu manuálního sledování času, ani zde se uživatel nevyhne ručnímu spouštění a zastavování časovače. Do tabulky je nutné data zadávat, proto se jedná o metodu, jenž může být časově náročná a nespolehlivá. (Timely, 2022d)

3.4.3 Přepínací kostky

Dalším přístupem k manuálnímu měření času jsou přepínací kostky. Jedná se o fyzická zařízení, které lze na základě otáčení přepínat a sledovat tak čas u předdefinovaných aktivit. Přepínací kostku je nutné připojit k aplikaci, a to nejčastěji mobilní či webové. Ani tato metoda, stejně jako žádná z výše uvedených, není schopna bez vstupu uživatele zastavit či spustit záznam. Se získanými daty je možné dále pracovat, a to prostřednictvím exportu, analýzou či editací dat nebo vizualizací dat prostřednictvím různých grafů. (TimeFlip, 2022b)

3.4.4 Software

Softwarový přístup spadá do manuálního, ale současně do automatického měření času. Oba způsoby zahrnují záznam času nástrojem nebo aplikací pro sledování času. Manuální způsob znamená záznam ručním zadáním informace, na čem uživatel pracuje či spuštěním časovače. Automatický přístup zachycuje informace o tom, na čem uživatel pracuje. Automatické sledování času je rychlejší, nejpřesnější a pro uživatele nejjednodušší způsob sledování času.

Ne vždy je však sledovací software etický, někteří zaměstnavatelé narušují soukromí zaměstnanců pořizováním snímků obrazovky, které v některých případech mohou obsahovat také soukromé informace. Samozřejmě existují nástroje, které aktivně chrání soukromí zaměstnanců a současně udržují firemní kulturu. (Timely, 2022d)

3.5 Dostupné nástroje

Nástroje zabývající se tématem měření času jsou obvykle dvojího typu, a to buď nástroje pro manuální, nebo nástroje pro automatizované měření času. Oba tito zástupci jsou dále i s příklady specifikováni v samostatných podkapitolách.

3.5.1 Nástroje pro manuální měření času

Dnešní trh nabízí širokou škálu řešení pro manuální měření času. Jednotlivá řešení se nejčastěji rozlišují dle licence, pod kterou jsou poskytovány, tedy zda se jedná o nástroj komerční nebo je poskytován bezplatně.

Většina nástrojů je postavena na tzv. *start-stop* systému, kdy uživatel manuálně spustí nebo zastaví měřený interval. Současně může uživatel přidat další časový interval nebo poznámku o prováděné aktivitě. Takto vytvořené záznamy je možné kontrolovat a prohlížet v přehledu, časové ose či grafu. Mnoho nástrojů dále umožňuje export zaznamenaných dat v různých formátech či přímo nabízí integraci s mnoha dalšími systémy. (Replogle, 2023)

TIMEFLIP2

TIMEFLIP2 je bezplatný nástroj, jenž je poskytován ve formě mobilní a webové aplikace. Společnost TimeFlip nabízí přepínací kostku TIMEFLIP2 Tracker, která umožňuje nadefinovat až 12 aktivit. Uživatel následně může otáčením kostky přepínat definované aktivity. Webová aplikace nenabízí možnost spouštět aktivity, slouží zejména k vizualizaci naměřených dat, jejich editaci či exportu. Mobilní aplikace kromě funkcionalit webové aplikace umožňuje definovat a spouštět aktivity, slouží ale také k nastavení přepínací kostky a propojení kostky s účtem uživatele. (TimeFlip, 2022b) V současné době nástroj poskytuje integraci pouze s Google, Apple a Microsoft Outlook kalendáři. Vývojáři mohou využít připravené API k vývoji vlastní integrace. Společnost plánuje do budoucna integraci s aplikacemi Slack, Jira a IFTTT. (TimeFlip, 2022a)

Timeular

Timeular je placený nástroj pro manuální měření času a je poskytován ve formě desktopové, mobilní a webové aplikace. Desktopová aplikace oproti ostatním obsahuje funkcionalitu „QuickTrack“. Funkcionalitu je možné vyvolat pomocí klávesové zkratky, následně je zobrazeno modální okno a uživatel má možnost přidat poznámku, novou aktivitu či zastavit měření. (Timeular, 2022b)

Nástroj je nabízen ve dvou cenových úrovních předplatného. Základní předplatné poskytuje mimo jiné přístup k desktopové, mobilní a webové aplikaci, či REST API,

díky kterému uživatel může vytvořit vlastní integraci. Oproti základnímu předplatnému poskytuje vyšší předplatné práci v týmu, analýzu trendů, export dat a již připravené integrace aplikace pomocí aplikace Zapier. Zapier je nástroj k automatizaci na principu *If This Then That*. Zapier na základě nadefinovaného spouštěče vyvolá zvolenou akci, např. je-li naměřena více jak jedna hodina nekategorizované práce v Timeular, vytvoř kartu v Trello. (Zapier, 2023) Tímto způsobem lze také integrovat Redmine.

Mimo běžnou aplikaci společnost Timeular nabízí Timeular Tracker, což je osmistěnná přepínací kostka, pomocí které lze otáčením přepínat předvolené aktivity. Uživatel má možnost nadefinovat až osm vlastních aktivit. Kostka dle autorů pomáhá vybudovat si návyk sledování času pomocí svalové paměti, což má mít za následek silnou fyzickou připomínku. (Timeular, 2022a)

Slabou stránkou tohoto řešení je skutečnost, že data uživatele jsou uložena u poskytovatele služby. Základní předplatné neposkytuje export dat ani možnost použít již existující integrace. (Timeular, 2022c)

Toggl Track

Nástroj Toggl Track poskytuje možnost snadného a výkonného sledování stráveného času. Toggl Track spadá do kategorie manuálních nástrojů a je nabízen ve čtyřech cenových úrovních předplatného včetně bezplatné verze. Uživatel prostřednictvím jednoho kliknutí může spustit a následně pak vypnout časovač, kterým ovládá měření. (Toggl, 2022a)

Slabinou tohoto řešení je forma uložení dat, ta jsou uložena u poskytovatele služby a společnost nemá nad daty plnou kontrolu. Ačkoliv lze spustit trackování přímo u vybrané *issue* v Redmine, Toggl Track není s Redminem integrovaný tak, aby uživatelé mohli přiřadit *issue* v rozhraní nástroje. Nelze tedy vytvořit výkazy u *issues* v Redmine na základě měření v nástroji Toggl Track. (Toggl, 2022b)

Obecně ale Toggl Track poskytuje více než sto integrací. Umožňuje zobrazení souhrnného, detailního a týdenního reportu. (Toggl, 2022c)

TTRebel

TTRebel je nástroj pro manuální měření času vyvinutý společností Orchitech Solutions, s.r.o. sloužící jako rozšíření trackovací funkcionality projektového systému Redmine. Blíže je tento nástroj představen v kapitole 3.8 TTRebel.

3.5.2 Nástroje pro automatické měření času

Na rozdíl od nástrojů pro manuální měření času je nabízeno podstatně méně nástrojů pro automatické měření času. Tito obvykle desktopoví klienti jsou nástroje sledující

aktivitu a práci uživatele na počítači. Sledují momentálně používané aplikace a zaznamenávají strávený čas, například u webových prohlížečů dokáží obvykle sledovat aktuálně používanou záložku. Nástroje dokonce umí rozpoznat nepřítomost uživatele, kdy na základě nečinnosti pohybů myši či stisků klávesnice chování vyhodnotí a daný interval příslušně označí. (Timely, 2022d)

Nástroje sledovanou aktivitu zpravidla člení do kategorií a zároveň poskytují uživateli možnost definovat si kategorie vlastní, do nichž jsou následně automaticky přiřazovány časové intervaly na základě určité aktivity. Definice kategorií lze provést buď výběrem konkrétních aplikací nebo použitím regulárních výrazů. Uživatel takto určí, že karta prohlížeče s názvem *Facebook* spadá do kategorie *Nezaúčtovatelné* nebo že používání aplikace *Visual Studio Code* bude kategorizováno jako *Vývoj*. Nástroj uživateli poté obvykle poskytuje možnost prohlížení dat přehledně v grafech nebo časových osách, na kterých jsou procentuálně zastupeny kategorizované činnosti či aplikace. (Timely, 2020)

Některá řešení jsou poskytována jako open-source s možností a také nutností vlastního provozování. Tato varianta má tu výhodu, že uživatel má svá data plně pod kontrolou. Jelikož je v nástrojích pro automatické měření času shromažďována veškerá aktivita, je možné k získaným datům přistupovat jako k citlivým údajům, u kterých nechceme, aby opustily naše zařízení. Další řešení jsou poskytována jako SaaS, kdy uživatel sice nemusí řešit vlastní provozování a může mít data synchronizována napříč několika zařízeními, ale zároveň nemá plnou kontrolu nad získanými daty. (Timely, 2022d)

Mezi nástroje pro automatické měření času spadá např. ActivityWatch, jenž je poskytován jako open-source nebo RescueTime poskytovaný jako SaaS.

ActivityWatch

ActivityWatch je modulární open-source nástroj umožňující automaticky sledovat aktivitu při práci na počítači. Filozofií nástroje je, že každý uživatel jej může rozšířit, nastavit a používat dle vlastních potřeb, navíc plně bezplatně. ActivityWatch je multiplatformní aplikace, mezi podporované operační systémy patří Linux, Windows, macOS a Android. Jedním ze základních principů nástroje je, že veškerá získaná data vlastní pouze uživatel, nikoliv společnosti poskytující službu jako to obvykle bývá u jiných nástrojů. Data jsou ukládána lokálně a neopouštějí zařízení. ActivityWatch však nabízí možnost importu a exportu dat pomocí připraveného REST API, díky tomu je možné jej integrovat s jinými nástroji. V době psaní této práce není dostupná synchronizace mezi více zařízeními. (Bjäreholt et al., 2022)

Nástroj umožňuje rozdělit naměřená data do kategorií, které si uživatel nadefinuje, což poskytuje lepší vhled do naměřených dat. Dále ActivityWatch nabízí rozšíření pro prohlížeče Chrome a Firefox, které mimo jiné poskytují možnost sledovat aktivní kartu prohlížeče. Nástroj také nabízí rozšíření pro textové editory, uživatel tedy může například sledovat, jak tráví čas při programování. (Bjäreholt et al., 2022)

ActivityWatch sleduje aktivitu uživatele pomocí tzv. *watchers*, což jsou moduly pro sledování aktivity. (Bjäreholt et al., 2021d) Data mohou být importována do nástroje pomocí tzv. *importers*. (Bjäreholt et al., 2021b) Nástroj obsahuje předinstalované dva *watchers*, a to *aw-watcher-afk* sledující přítomnost či absenci uživatele pomocí aktivity klávesnice a myši, a *aw-watcher-window* sledující aktuálně aktivní aplikaci a název okna. ActivityWatch poskytuje široké množství rozšíření pomocí připravených *watchers* a *importers*, každý uživatel si tak může sestavit nástroj podle vlastních potřeb. Nedostatkem ActivityWatch je absence integrace s Redmine, nicméně díky připravenému REST API a skutečnosti, že všechna data má uživatel uložena lokálně, je možné snadno vytvořit vlastní integraci.

RescueTime

RescueTime je komerční nástroj pro automatické sledování aktivity. Veškerá zaznamenaná data jsou uložena u poskytovatele služby. Uživatel má možnost definovat neproduktivní aplikace či denní cíle produktivní práce. Následně je uživatel nástrojem upozorňován pomocí notifikací při dosažení stanoveného cíle, dlouhé neaktivity nebo sledováním neproduktivní aplikace, čímž se uživatele snaží motivovat k produktivnějšímu pracovnímu dni. (RescueTime, 2021c)

RescueTime nabízí integraci například s aplikacemi GitHub, Google Workspace nebo Visual Studio Code. Nástroj lze napojit na další aplikace pomocí aplikace Zapier, tímto způsobem lze také integrovat Redmine. (RescueTime, 2021a)

Ačkoliv lze integrovat Redmine pomocí aplikace Zapier, jedná se o další finanční náklad, jelikož Zapier je placený nástroj. Výše předplatného se odvíjí od počtu uživatelů a počtu spuštěných úkolů. Ve výsledku je tedy nutné platit předplatné jak pro RescueTime, tak i pro Zapier. Nevýhodou tohoto řešení také pro některé uživatele může být ukládání dat u poskytovatele služby. (RescueTime, 2021b)

Timely

Timely se řadí mezi multiplatformní placené automatické nástroje pro sledování aktivity. Timely dokáže na základě spuštěné Memory aplikace uživatelům zaznamenávat čas strávený na jakémkoliv webu nebo v jakékoliv aplikaci. Je možné sledovat nejen strávený čas, ale také sledovat vývoj projektů, výkon týmů a novinkou je sledování

nastavených plánů. Na základě těchto záznamů uživatel získá přehled, kolik času strávil vývojem, schůzkami nebo úpravou dokumentů. Aplikace uchovává data u poskytovatele služby, přičemž obsahuje přísnou politiku proti sledování, čímž je zajištěno 100% soukromí uživatele. (Timely, 2022c)

Trial verzi aplikace je možné vyzkoušet si zdarma na 14 dní. Startovací balíček, který poskytuje přístup pro sledování 50 projektů, při platbách za rok začíná na částce 8 USD/měsíc. Dále je možnost využít balíček Premium, Unlimited či Unlimited +. Preferovaný balíček se odvíjí od potřeb dané firmy. (Timely, 2022e) Timely je možné integrovat s aplikací Google Kalendář, Outlook, Zoom, Jira a další. (Timely, 2022b)

WakaTime

WakaTime je nástroj pro automatické sledování aktivity uživatele v editorech, které jsou nástrojem podporovány. Aktivita je měřena pouze při psaní. Nástroj je nabízen jak v bezplatné, tak i v placené verzi. WakaTime v bezplatné verzi nabízí integraci s aplikacemi jako je GitLab nebo GitHub, pomocí kterých je možné měřit čas dle commitů, úloh či pull requestů. Dále nabízí integraci s Google Kalendářem, z něhož umí měřit čas strávený na schůzkách. Nástroj nabízí další integrace, jako je Office 365, Slack nebo Zoom, avšak tyto integrace jsou dostupné pouze v placené verzi. Nástroj však nenabízí integraci s Redmine. (WakaTime, 2022)

3.6 Podpůrné prostředky pro sledování aktivity

Mezi podpůrné prostředky pro sledování aktivity lze zařadit senzory sledující přítomnost uživatele na bázi měření vzdálenosti. Pomocí senzoru měřící vzdálenost lze určit, zda-li je v definovaném rozsahu přítomný objekt. Na základě této informace můžeme následně určit, zda uživatel sedí u počítače či nikoliv. Tento prostředek však nedokáže určit, zda přítomný uživatel sleduje monitor a skutečně pracuje, nebo se věnuje jiné nepracovní aktivitě.

V dnešní době na trhu také existují řešení, která sledují aktivitu uživatele pomocí tzv. *eye-trackingu*. Tento nástroj dokáže určit přítomnost uživatele a zároveň dokáže určit, zda-li uživatel sleduje monitor nebo nikoliv. Danému uživateli však nemusí být příjemné neustálé sledování kamerou, a to nejen vzhledem ke ztrátě soukromí a určitého osobního prostoru, ale také například kvůli vědomí nedostatečného času, jenž věnuje pracovním činnostem.

3.6.1 Akustické technologie

Akustické technologie se zaměřují na oblast vibrací a zvuku, zejména technologiemi věnujícími se produkci, ovládání, přenosu, příjmu a účinkům zvuku. (Kleiner, 2011)

Zvukové signály, sestávající se z tlakových vln šířících se vzduchem, těží z toho, že zvuk se šíří mnohem pomaleji než elektromagnetické signály, a umožňuje tak mnohem snadněji měřit čas mezi vyzařováním a příchodem signálu. Do této skupiny spadá slyšitelný zvuk a ultrazvuk. (Brena et al., 2017)

Slyšitelný zvuk

Zvuk je forma energie přenášená vlnami, které je možné zachytit prostřednictvím čidel. Zvuk se nejčastěji dělí na dva typy, a to zvuk slyšitelný a neslyšitelný, podle toho, zda je lidské ucho schopné tyto zvuky vnímat. Rozdíl je ve frekvenci, kdy slyšitelný zvuk se nachází obvykle v rozmezí 20 HZ až 20 kHz, neslyšitelný zahrnuje frekvence pod 20 Hz a nad 20 kHz. Senzory zpravidla reagují právě na slyšitelný zvuk, který nemusí být uživatelům, zejména v delším časovém horizontu používání, příjemný. Jsou proto používána sofistikovanější schémata, která používají vodoznak zvukových signálů již dostupného zvuku v daném prostoru. Takto vysílané zvukové signály již uživatel nezpozoruje. (Brena et al., 2017) Obecně tyto senzory obsahují mikrofon, který detekuje intenzitu zvuku prostředí, kde citlivost se liší dle jednotlivých řešení.

Ultrazvuk

Ultrazvukové senzory používají k měření vzdálenosti mezi senzorem a objektem ultrazvukové vlny s frekvencí větší než 20 kHz. (Brena et al., 2017) Senzor vysílá ultrazvukové vlny a čeká na navrácení odražených vln. Fyzickou vzdálenost lze vypočítat jako rozdíl v čase mezi vysláním zvukové vlny a navrácením odražené vlny. Některé senzory mají integrované teplotní čidlo, které pomáhá přesněji určit vzdálenost, jelikož zvuk cestuje odlišnou rychlostí dle teploty prostředí. (Shawn, 2020) Tyto senzory typicky umí měřit vzdálenost od jednotek centimetrů do jednotek metrů. Výhodou těchto senzorů je nízká pořizovací cena a energetická náročnost. Oproti zvukovým signálům mají dále tu výhodu, že jsou pro lidské ucho neslyšitelné. Nevýhodou těchto senzorů je omezená vzdálenost, pro zamýšlené účely této práce je však dostatečná. (Brena et al., 2017)

3.6.2 Optické technologie

Optické technolgie zahrnují všechny technologie, které slouží ke generování, přenosu, zesílení, manipulaci, tvarování, měření nebo využití světla. Mimo viditelného světla, které je identifikováno jako elektromagnetické vlny s vlnovými délkami zhruba v rozsahu 380–780 nm, využívají technologie také sousední oblasti elektromagnetického světla, například infračervené světlo. (Holtmannspötter et al., 2009)

Infračervené záření

Senzory reagující na infračervené záření realizují záznam vzdálenosti prostřednictvím vyzařování infračervené vlny a výpočtu úhlu odrazu. (Brena et al., 2017) Dioda vyzařující infračervené světlo vysílá signál pomocí záblesků neviditelného světla. Tento světelný paprsek následně dopadá na objekt a odráží se pod určitým úhlem. Odražený paprsek dopadá na polohově citlivý detektor (PSD), tzv. fotodiodu. Senzor v PSD určí vzdálenost předmětu a na základě vzdálenosti je možné vyhodnotit přítomnost uživatele. Obvyklá oblast, ve které tato technologie detekuje přítomnost, je v rozsahu desítek centimetrů. (Shawn, 2020)

3.6.3 Rádiové technologie

Většina rádiových technologií, které se používají pro polohové systémy, využívá rádiové signály omezené na malý rozsah frekvencí (tzv. úzkopásmové signály). Existují také aplikace využívající velké části spektra, tedy signály s rozprostřeným spektrem. Mezi rádiové technologie se řadí například Wi-Fi, Bluetooth, RFID a ZigBee. Tyto technologie bývají využívány ve vnitřních polohových systémech. (Shawn, 2020)

3.6.4 Eye-tracking

Eye-tracking je metoda zaznamenávající pohled a pohyb očí v průběhu času. Eye-tracker měří, kam, jak a v jakém pořadí uživatel směřuje svůj pohled během konkrétní úlohy. V každém okamžiku je možné pozorovat pouze malou část zorného pole s vysokou ostrostí. Právě v této části zorného pole se v daný okamžik nachází podnět, o kterém uživatel přemýšlí nebo jej zpracovává. Eye-tracking může využívat nevědomé zpracování podnětů, jelikož pohyby očí jsou z velké části mimo vědomou kontrolu. Jedinec se může rozhodnout, na co se dívá, ale detaily tohoto pohybu jsou do značené míry reflexní, tedy mimovolné. (Carter et al., 2020)

Moderní eye-trackery nejčastěji fungují na pricipu, kdy zdroj světla (například infračervené) vytváří odraz na rohovce. Tento odraz je identifikován daným softwarem a na základě nastavení senzoru, při tzv. kalibraci, je možné s vysokou mírou přesnosti odhadnout polohu zornice. (Carter et al., 2020)

Cena těchto senzorů se liší dle rychlosti získávání dat, které závisí na vzorkovací frekvenci, během níž se za jednu sekundu zaznamenává poloha očí. Mezi nejvíce rozšířeného výrobce patří Tobii, produkt Tobii Eye Tracker 5 je dostupný v současné době za 259 €, tedy asi 6 200 Kč. Na českých webech se v lednu 2023 cena pohybuje kolem 7 500 Kč s DPH.

Po konzultaci s managementem společnosti Orchitech Solutions, s.r.o. nebude využita metoda eye-trackingu. Jelikož se jedná pouze o doplňující prostředek k přesnějším

podkladům, je cena pro pořízení tohoto senzoru vysoká natolik, že pro účely práce cena i z dlouhodobého hlediska převyšuje očekávaný přínos.

3.7 ActivityWatch

V kapitole bude představen nástroj ActivityWatch, a to jeho architektura a nástroje, jejichž znalost je klíčová pro účely této práce. Dále budou předloženy ukázky obsahující výchozí data a nastavení, tedy jednotlivé části aplikace, jako je dashboard, časová osa, stopky, kategorizace aktivit a ukázka dat v nezpracované formě.

3.7.1 Architektura

Architekturu nástroje ActivityWatch lze zjednodušeně charakterizovat jako server-klient. Server je spuštěný na pozadí systému uživatele a zajišťuje manipulaci s daty, zapouzdřuje logiku aplikace a komunikuje s klienty, nesleduje však žádnou aktivitu uživatele ani nesbírá žádná data. Klienty jsou v terminologii myšleni tzv. *watchers* a *importers*, kteří se starají o sběr dat.

Importer na vyžádání importuje data do ActivityWatch, typicky z jiného zdroje. Úlohou *importer* je získání požadovaných dat z jiného zdroje, zpracování dat do vhodné formy a následný import na server. Typickým případem užití může být importování dat z jiného nástroje, kde uživatel již má informace o aktivitě, např. Google Kalendář. (Bjäreholt et al., 2021c)

Watcher periodicky sleduje aktivitu uživatele a pravidelně odesílá data na server. *Watcher* bývá typicky zaměřen na jeden zdroj informací, např. přítomnost uživatele u počítače nebo sledování aktivní aplikace.

Datový model

Event je entita reprezentující měřenou událost. Obsahuje časové razítko vzniku události, její trvání a data. *Heartbeat* je metoda, která spojuje sousední *events* s identickými daty v rámci tzv. pulzního okna. Pulzní okno je časové rozmezí, ve kterém se s jistotou nacházejí dva *events*. Typicky se jedná o delší časový úsek, než je doba mezi opakovaným získáním nových dat. Tato metoda je užitečná pro úsporu místa v úložišti a vytížení disku a doporučuje se, aby *watchers* používali tuto metodu při odesílání *events* na server. Sloučení dvou *events* A a B se provede, pokud jsou jejich data identická a jejich časová razítka jsou v rámci pulzního okna. Výsledný *event* bude mít dřívější časové razítko a trvání takové, které odpovídá rozdílu mezi časovými razítky. *Bucket* je samostatný jmenný prostor, jehož obsahem je množina *events* stejného typu. Obvykle má každý klient právě jeden *bucket*. (Bjäreholt et al., 2021a)

3.7.2 Nástroje pro sběr dat

Jak již bylo uvedeno, ActivityWatch sleduje aktivitu uživatele pomocí *watchers*, kteří periodicky monitorují jednotlivé typy aktivity. *Watcher* může sledovat v podstatě jakýkoliv typ aktivity, obvykle však bývá zaměřen právě na jednu.

ActivityWatch v základní verzi poskytuje předinstalovaný *watcher aw-watcher-afk*. Analýzou zdrojového kódu bylo zjištěno, že každých pět sekund provede kontrolu, zda bylo aktivováno nějaké vstupní zařízení a pokud po dobu tří minut nebyla zaznamenána žádná aktivita, je aktivita uživatele označena jako *AFK*. Tento *watcher* může být použit při rozhodování, zda naměřený časový interval může být účtovaný zákazníkovi.

V základní verzi ActivityWatch také poskytuje předinstalovaný *watcher aw-watcher-window*, který sleduje aktivní okno aplikace. Tento *watcher* pouze sleduje název okna aplikace, nikoliv jeho obsah. Uvedený *watcher* je užitečný ve sledování práce uživatele, kdy s vhodně nastavenou kategorizací lze určit, zda se uživatel věnoval pracovní či mimopracovní činnosti. S vhodnou kategorizací lze také určit, o jaký druh pracovní aktivity se jednalo, např. vývoj (development) či code review.

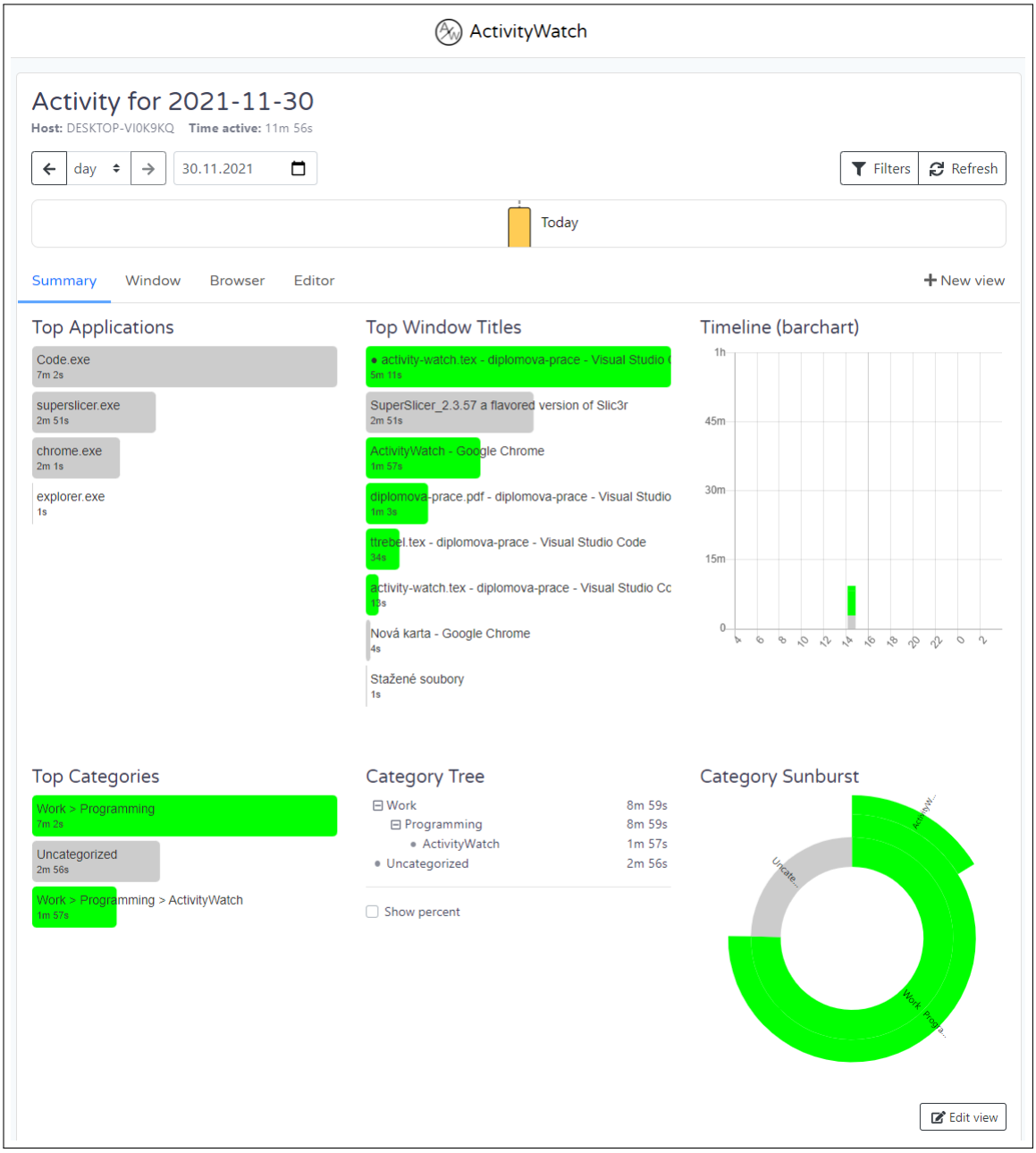
Jak již bylo uvedeno, *aw-watcher-window* sleduje pouze aktivní okno aplikace, v případě webového prohlížeče však uživatel navštěvuje různé stránky a pro tento případ je vhodné použít *watcher aw-watcher-browser*. Zmíněný *watcher* lze nainstalovat do prohlížeče jako rozšíření, které sleduje aktivní záložku prohlížeče, v době psaní této práce byly podporovány prohlížeče Google Chrome a Mozilla Firefox. (Bjäreholt et al., 2021d)

Dalším zdrojem informací může být *watcher aw-watcher-vscode*, což je rozšíření IDE Visual Studio Code, které sleduje aktivitu uživatele při práci s Visual Studio Code. Tento *watcher* poskytuje informace o názvu projektu, programovacím jazyce a názvu souboru, se kterým uživatel pracoval. Významnou informací, kterou lze z IDE Visual Studio Code získat, je název větve ve verzovacím nástroji Git. Vývojáři obvykle pojmenovávají větev podle identifikátoru *issue* v Redmine, např. *rm40123*, tudíž by tato informace mohla zásadně usnadnit určení *issue*, na které uživatel pracuje. Tuto informaci však zmíněný *watcher* implicitně neposkytuje. Implementace této funkcionality bude řešena v rámci kapitoly 4.7.1 ActivityWatch.

3.7.3 Ukázka aplikace

Dashboard

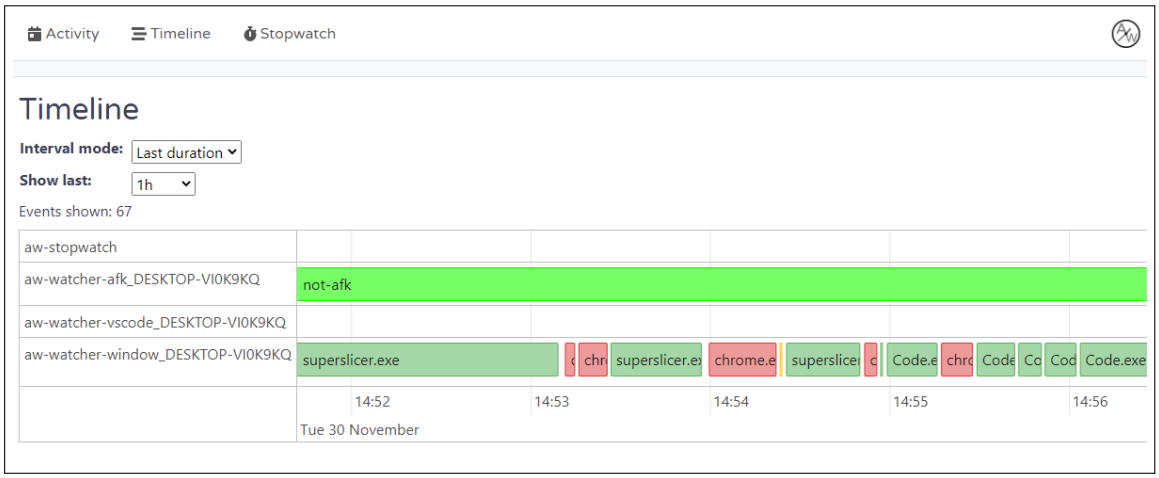
Po přihlášení může uživatel zobrazit *dashboard*, tedy přehled o jeho naměřených datech, viz obrázek 4. Uživatel má možnost filtrovat data podle data, má také na výběr ze zobrazení konkrétního dne, týdne a měsíce. Na této stránce se nachází seznam nejpoužívanějších aplikací, u kterých je zobrazen čas strávený jejich používáním, dále se zde nacházejí nejčastěji používaná okna a časová osa. Ve spodní části obrazovky se nachází nejčastější kategorie a výsečový graf, který je rozdělen dle kategorizované aktivity.



Obrázek 4: Celkový přehled naměřených dat - vlastní zpracování

Časová osa

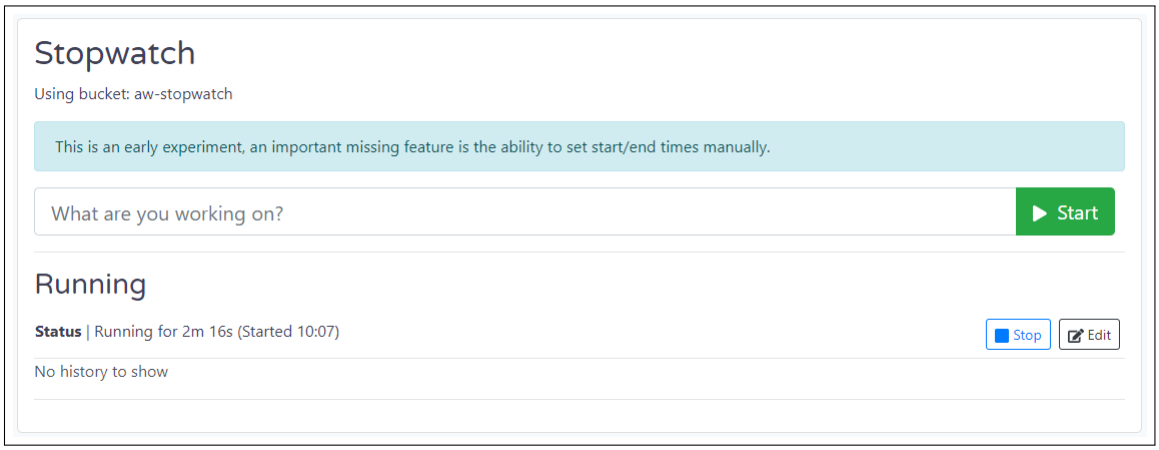
Časová osa znázorněná na obrázku 5 obsahuje přehled jednotlivých *watchers* a *importers*. Uživatel má možnost filtrovat události podle data nebo intervalu. Na časové ose jsou vyobrazeny časové řady jednotlivých událostí, po najetím kurzorem jsou u dané události zobrazeny dodatečné informace.



Obrázek 5: Ukázka časové osy - vlastní zpracování

Stopky

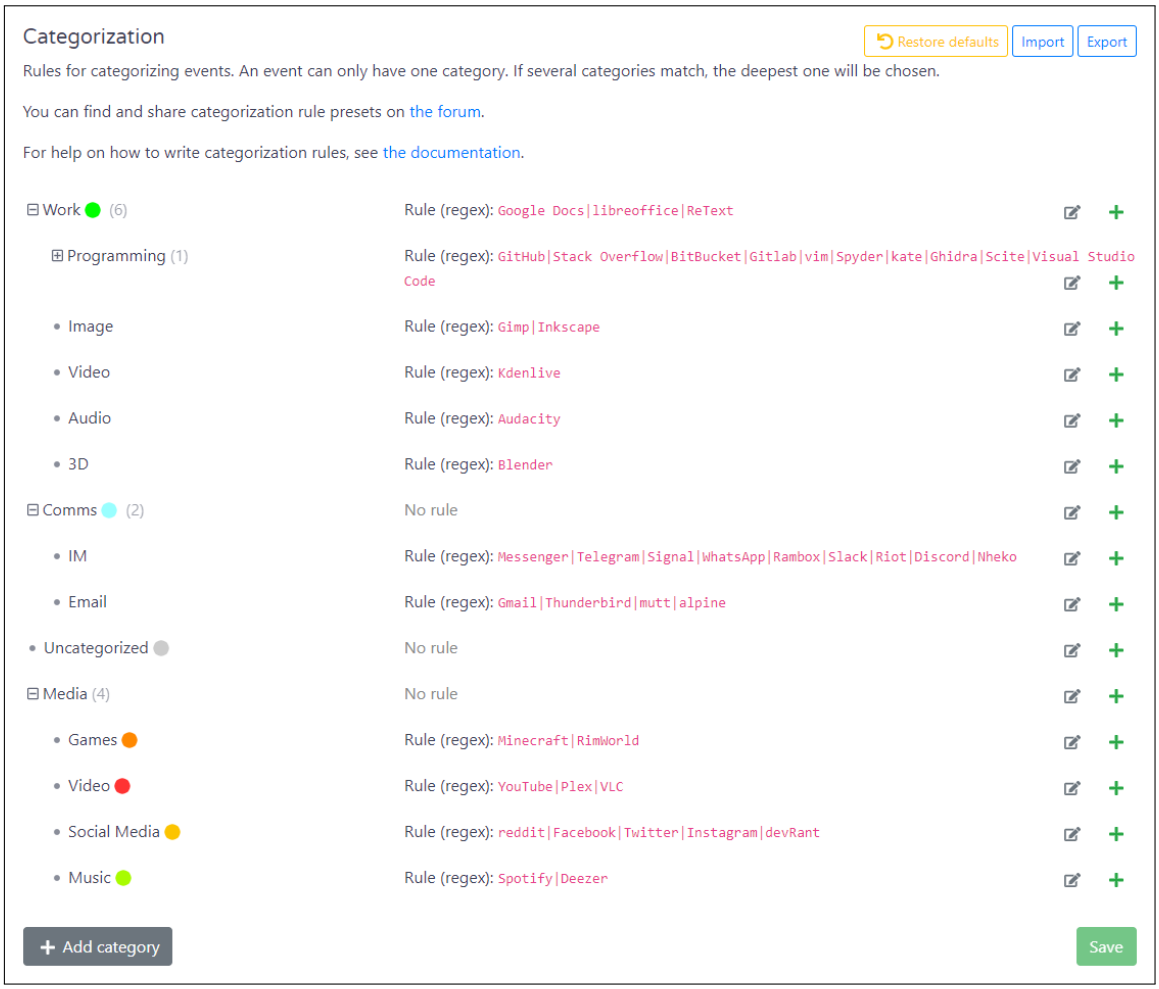
ActivityWatch poskytuje stopky pro manuální měření času, avšak tato funkcionalita je experimentální a obsahuje chyby. Uživatel má možnost upravovat jednotlivé záznamy, včetně ruční úpravy intervalů. Tato funkcionalita je do jisté míry podobná funkcionalitě aplikace TTRebel, avšak neposkytuje takové možnosti, jako TTRebel. Ukázka této funkcionality je znázorněna na obrázku 6.



Obrázek 6: Ukázka zabudovaných stopek - vlastní zpracování

Kategorizace aktivit

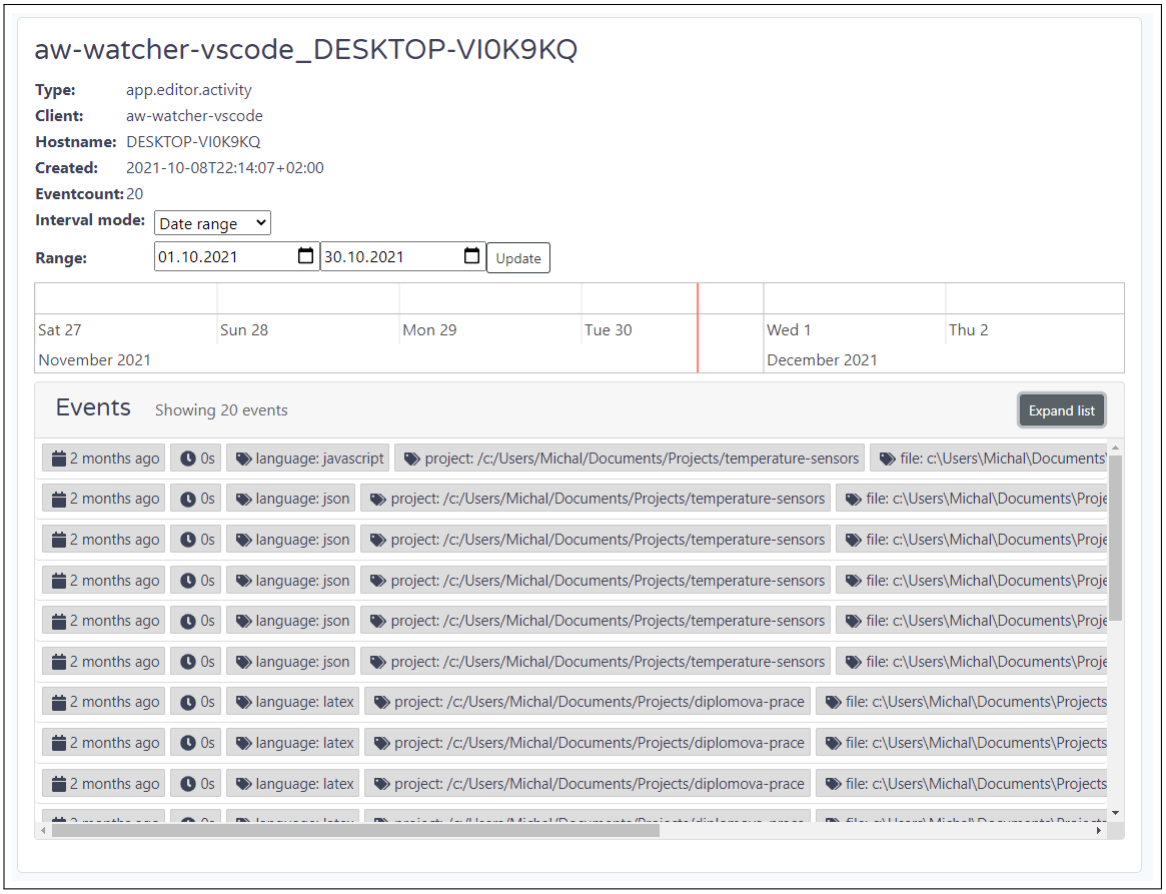
Kategorizace aktivit je v aplikaci ActivityWatch řešena pomocí regulárních výrazů. Na obrázku 7 jsou zobrazeny výchozí kategorie, uživatel je však může libovolně upravit či odstranit. V pravé horní části obrazovky se nacházejí tlačítka pro import a export kategorií.



Obrázek 7: Ukázka kategorizace aktivit - vlastní zpracování

Nezpracovaná data

ActivityWatch poskytuje uživateli data také v nezpracované formě, např. pro externí zpracování. Data lze exportovat jak všechna, tak i informace pro jednotlivé *watchers* a *importers*. Na obrázku 8 jsou zobrazena data *watcher aw-watcher-vscode*. Obrazovka dále obsahuje časovou osu, kde jsou po najetí kurzoru zobrazena stejná data pro danou událost jako na přehledu ve spodní části obrazovky.



Obrázek 8: Ukázka nezpracovaných dat - vlastní zpracování

3.8 TTRebel

Filozofií nástroje TTRebel, který je vyvíjen firmou Orchitech Solutions, s.r.o. je co nejvíce uživatelům zpříjemnit měření a vykazování odpracovaného času. Důvodem vzniku nástroje byla absence vhodného řešení, které by splňovalo požadavky a potřeby firmy a zaměstnanců. Většina existujících nástrojů pro sledování času integrovaných s Redmine nenabízela možnost exportovat naměřený čas do Redmine v podobě výkazů a ty, které to nabízely, byly closed-source či placené, což neumožňovalo jejich rozšíření o další požadované funkcionality. Nástroj je provozován na interních serverech a společnost Orchitech Solutions, s.r.o. tak má plnou kontrolu nad daty.

V průběhu let se tento nástroj vyvíjel dle potřeb zaměstnanců tak, aby zaznamenávání a následné vykazování stráveného času bylo co nejpřívětivější. Současně tak, aby byla aplikace co nejpřehlednější, ale zároveň obsahovala všechny možnosti, které uživatel využívá a aby zaznamenaný strávený čas obsahoval potřebné informace. Na základě podnětů vznikla také mobilní aplikace, díky které je vykazování ve firmě opět o něco jednodušší. Nástroj v současné době nabízí integraci pouze se systémem Redmine.

3.8.1 Architektura aplikace

Po technické stránce je aplikace navržena jako běžná moderní webová aplikace - klient a server. Frontend (klient) je napsán v JavaScriptovém frameworku VueJS, backend (server) je napsán v jazyce Java s frameworkem Spring Boot. Komunikace mezi frontendem a backendem i mezi backendem a Redmine probíhá pomocí REST API. Jako úložiště dat používá aplikace databázi PostgreSQL.

Základním prvkem aplikace je *tracking session*, což reprezentuje jednu *issue* v Redmine, v rámci které si uživatelé měří čas strávený řešením *issue*. Uživatelé mohou v aplikaci přidávat, měřit a upravovat jednotlivé *tracking sessions*, přiřadit jim *issue* z Redmine, přidat detailnější informaci týkající se pracovní činnosti a nastavit jednu z předdefinovaných aktivit. Samotnou informaci o naměřeném čase obsahuje *tracking interval*, který je ve vztahu N:1 k *tracking session*.

Aplikace obsahuje, kromě přihlašovacího formuláře, dvě stránky - hlavní stránku sloužící pro správu *tracking sessions*, které jsou shlukovány po dnech a exportní stránku sloužící pro export naměřených *tracking sessions* do Redmine.

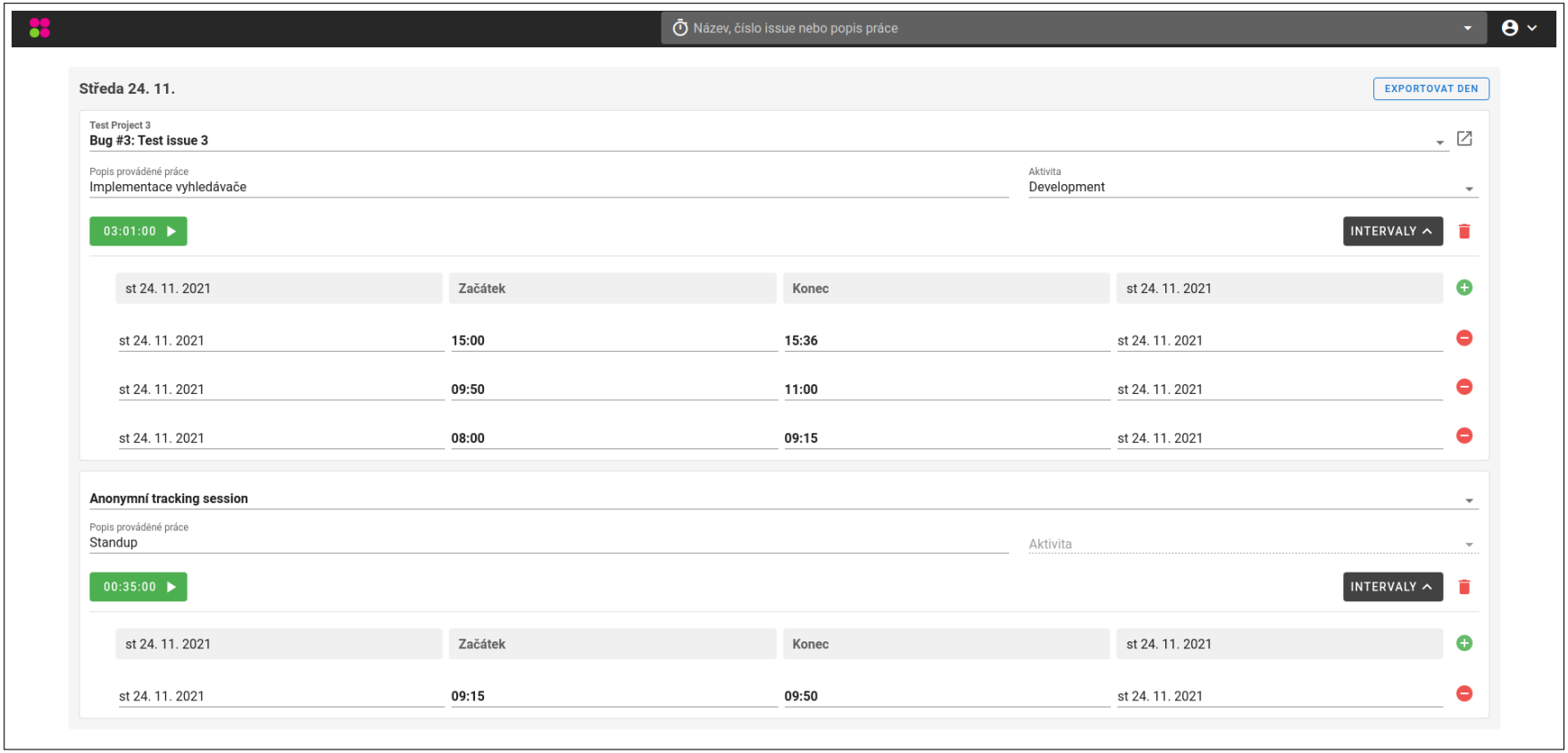
3.8.2 Ukázka aplikace

Po přihlášení do aplikace se uživateli zobrazí hlavní stránka (viz obrázek 9), na které má možnost vytvářet a spravovat jednotlivé *tracking sessions*. Pomocí pole v pravé horní části obrazovky může přidávat nové *tracking sessions*, v rozbalovacím seznamu může buď vybrat anonymní *tracking session*, nebo přímo pro konkrétní *issue* v Redmine, kterou má na sobě přiřazenou. Pomocí vyhledávače může také vybrat jakoukoliv *issue*, ke které má přístup. Pro vyhledání *issue* je možné použít jak její identifikátor, tak i název.

Anonymní *tracking session* znamená, že není přiřazená k žádné úloze v Redmine, bez nastavení *issue* však není možné takový záznam vyexportovat do Redmine. Pokud uživatel vybere ze seznamu některou *issue* v Redmine, dojde k vytvoření *tracking session*, která má nastavenou příslušnou *issue*. Implicitně mají všechny vytvořené trackin sessions nastavenou aktivitu *Uncategorized*, a to na základě požadavku uživatelů.

Vedle zmíněného pole se nachází tlačítko, které v současné době slouží pouze k odhlášení uživatele. V hlavní části obrazovky se nacházejí jednotlivé *tracking sessions*, které jsou seskupovány dle data vytvoření.

Úprava naměřeného času je možná buď pomocí úpravy časových intervalů na detailu *tracking session*, či úpravou hodnoty celkového naměřeného času na exportní stránce.



Obrázek 9: Hlavní stránka aplikace TTRebel - vlastní zpracování

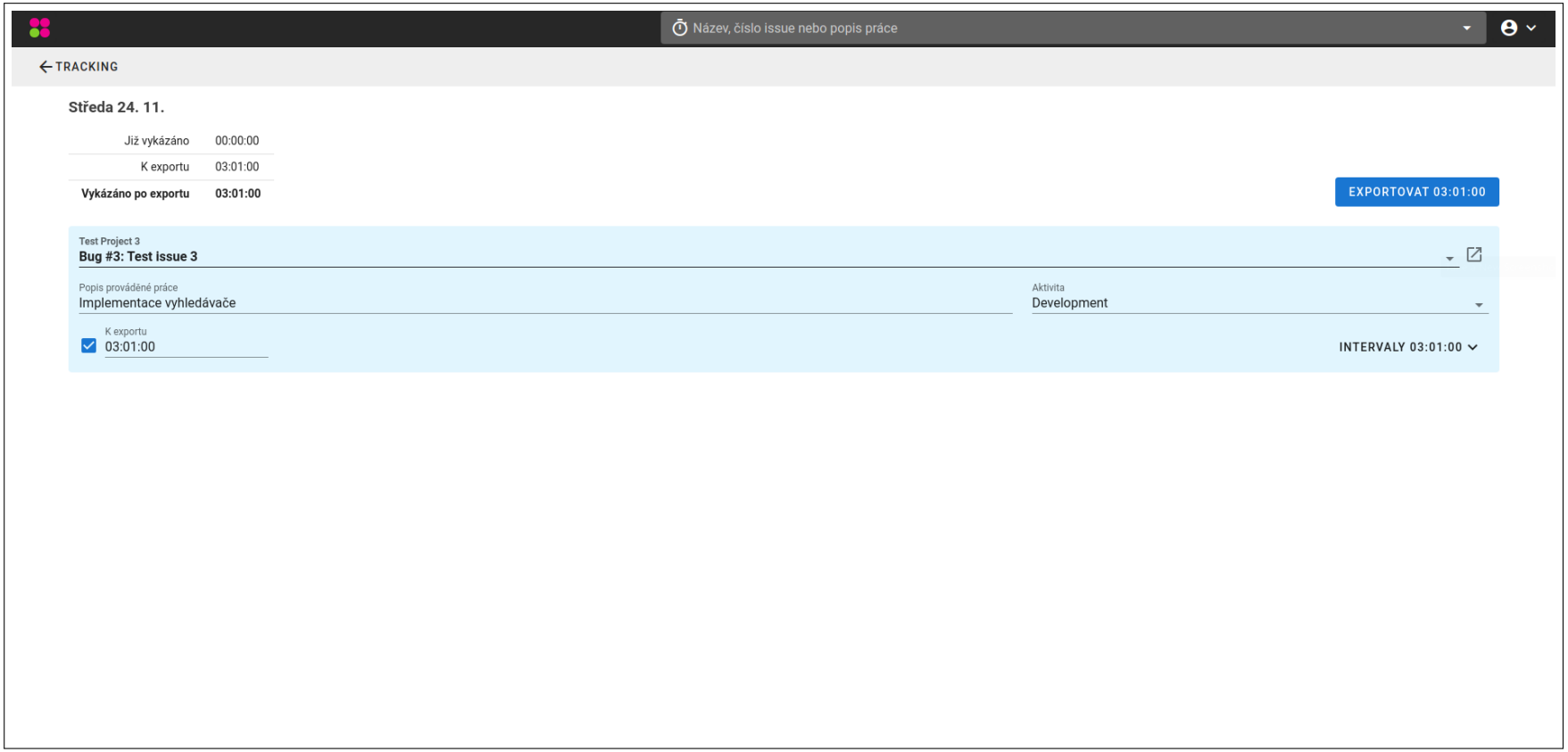
Po stisknutí tlačítka *Exportovat den* je uživatel přesměrován na exportní stránku (viz obrázek 10), kde se nacházejí *tracking sessions* pro vybrané datum. Ve vrchní části obrazovky se nachází přehled, pomocí kterého uživatel zjistí již vykázaný čas pro dané datum v Redmine, čas k exportu a celkový čas po exportu. Pomocí zaškrtávacího políčka na levé straně obrazovky lze vybrat konkrétní *tracking sessions*, které se mají exportovat do Redmine, implicitně jsou vybrány všechny. Nemá-li některá *tracking session* přiřazenou *issue* v Redmine nebo jí chybí komentář, není možné takový záznam exportovat do Redmine. Na této stránce lze také upravovat naměřený čas konkrétní *tracking session* pomocí přidání či upravení stávajících intervalů. Dále lze přidávat či měnit *issue* v Redmine, komentář a aktivitu vybrané *tracking session*, stejně jako na hlavní stránce aplikace.

Po stisknutí tlačítka *Exportovat* dochází k odeslání vybraných dat do systému Redmine pomocí rozhraní REST, a to ve formátu JSON - viz ukázka 1.

```
{
  "time_entry": {
    "comments": "Status".
    "issue_id": "3",
    "spent_on": "2021-11-24",
    "activity_id": "1",
    "hours": "3.02"
  }
}
```

Kód 1: Ukázka dat exportovaného záznamu do Redmine

V budoucnu je plánováno přidání automatického dopočítání času do osmi hodin, tedy plného úvazku, jelikož ne vždy se uživateli podaří naměřit veškerý pracovní čas, ačkoliv pracoval. Tato funkcionalita již byla obsažena v původní aplikaci TTRebel než došlo k přechodu na nový framework a kompletnímu refaktoringu aplikace.



Obrázek 10: Exportní stránka aplikace TTRebel - vlastní zpracování

4 Vlastní práce

4.1 Současný stav

V rámci této kapitoly bude představena společnost Orchitech Solutions, s.r.o., pro níž je realizována a na míru přizpůsobována implementace automatického time trackingu, která je tématem této práce.

4.1.1 Charakteristika firmy

Firma Orchitech Solutions, s.r.o. je česká technologická firma figurující v odvětví vývoje software, fungující na trhu od roku 2006. Firma pro své klienty připravuje softwarová řešení na míru, a to zejména v oblasti Identity Management. Jedná se o vývoj a nasazování systémů informační bezpečnosti. Dále také spravuje platformu pro studentské ISIC karty nebo benefitní program. Zároveň zaměstnanci pracují na několika interních projektech, jedním z nich je aplikace pro sledování času - TTRebel.

Menší počet zaměstnanců, jenž se dlouhodobě pohybuje do 20 pracovníků, umožňuje jednomu zaměstnanci zastávat různé role na různých projektech. Zaměstnanec, který primárně pracuje jako vývojář, se může současně věnovat požadavkům z odvětví DevOps nebo být projektovým manažerem některého projektu. Automatizace a zjednodušení vykazování času zaměstnanců tak pro firmu bude představovat značné vylepšení tohoto procesu.

4.1.2 Používané nástroje

Firma Orchitech Solutions, s.r.o. pro projektové řízení používá primárně nástroj Redmine, open-source software pro řízení projektů, který je využívaný pro evidenci stráveného času u jednotlivých *issues* a k řízení projektů. Z projektového hlediska zahrnuje informace jako jsou definice požadavků, rozsah, odhady nebo například datum předání, které jsou nedílnou součástí *issue* v Redmine. K řízení projektů je u některých zákazníků navíc používán systém Jira, ze kterého probíhá pravidelná synchronizace relevantních *issues* do systému Redmine. Redmine byl upraven tak, aby odpovídal účelům firmy a také potřebám společnosti a zaměstnanců.

Měření a vykazování stráveného času zaměstnanců je zaznamenáváno interním nástrojem TTRebel, jenž je integrovaný se systémem Redmine. Uživatelsky přívětivé prostředí slouží k příjemnějšímu a jednoduššímu vykazování. Tento nástroj je blíže představen v kapitole 3.8 TTRebel.

Firma taktéž využívá nástroje Google Workspace, z nich pak nejčastěji Gmail, Meet či Kalendář. Prostřednictvím nástroje Slack je zajištěna okamžitá komunikace mezi

zaměstnanci. Slack je v některých případech používán i pro rychlou komunikaci se zákazníky, přičemž vždy záleží na preferovaném komunikačním kanálu využívaném na straně zákazníka. Dále jsou pro tyto účely využívány aplikace Microsoft Teams, Skype nebo Element. Gitlab, tedy webový Git repozitář, je využit pro správu zdrojových kódů.

4.1.3 Využití výkazů

Výkazy ve firmě slouží jako podklad pro efektivní alokace zaměstnanců a současně se na jejich základě připravují reporty dodávané zákazníkům. Čas zaměstnanců je vyplňován prací na různých projektech, přičemž velká část firmy stráví pracovní dobu během týdne, mnohdy i během dne, na projektech pro různé zákazníky.

Vykazování stráveného času

Každý den je jiný, a tak potenciálně může vzniknout jediný záznam za celý den. Obvykle ale během dne vzniká takových záznamů více, z nichž si zaměstnanci každý samostatně měří. Může se jednat o různé požadavky na jednom projektu či na více projektech. Na základě výkazů zaměstnanec a firma získávají přehled o tom, kde je tráven čas.

Zaměstnanci si každý den zaznamenávají svou práci, která se skládá z:

- projektu,
- čísla *issue*,
- délky stráveného času,
- aktivity,
- komentáře.

V rámci aktivity je dostupná nabídka, ve které si zaměstnanec zvolí, zda pracoval na vývoji, analýze, managementu nebo například testoval. Výjimkou nejsou ani *issues* týkající se schůzek k jednotlivým projektům. Ty zahrnují zejména pravidelné schůzky vzhledem k agilnímu řízení projektů, jedná se o daily scrum, planning, rektrospektivu nebo refinement.

Každý výkaz v Redmine lze označit hodnotou *tech-learners*, kterou využívají především nově nastupující zaměstnanci. Tato hodnota poukazuje na skutečnost zaučování daného zaměstnance na projektu nebo v určité technologii.

Samotný záznam a vykazování patří obecně k nejméně oblíbeným aktivitám, které ale mají své místo v procesu projektů a fungování firmy. Na začátku každého měsíce vždy přichází povinnost zkontrolovat a dokončit výkazy za měsíc předcházející.

Řízení projektů

Výkazy jsou velice důležité pro řízení projektů. Na jejich základě projektoví manažeři získávají potřebné informace, ze kterých dále připravují přehledy pro alokaci členů týmu na daném projektu.

Podle stráveného času je možné průběžně kontrolovat čerpání a zavčas zjistit rozdíly v původně odhadovaném čase a skutečně stráveném čase pro danou objednávku. Znalost této informace je z pohledu řízení projektu podstatná, jelikož by mohlo být ohroženo datum předání a nasazení dodávky.

V případě objednávky na skutečně strávený čas jsou tyto reporty podkladem pro kontrolu a akceptaci předávané klientovi. Pokud se jedná o předem domluvený fixní čas, je nutné jej mít na paměti a průběžně jej kontrolovat se záměrem nepřekročit zaplacený čas.

Řízení firmy

Výkazy jednotlivých zaměstnanců vedení firmy používá pro plánování a řízení firmy z dlouhodobého hlediska. Při zjištění neefektivní alokace zaměstnanců v důsledku nedostatku práce je možné zavčas hledat další strategické klienty. Naopak v případě většího objemu plánované práce je možnost včas začít shánět nové zaměstnance.

Účetnictví

Informace o stráveném čase, tedy reporty, jsou klientům předkládány na základě dat vykázaných zaměstnanci. Tato data pak firma použije při fakturaci klientům. Fakturovaný čas zahrnuje zejména vývoj, testování nebo podporu projektu klienta.

Pracovní nasazení

Znalost výkazů, a tím stráveného času v práci, je důležitá také pro zjištění či předcházení nedostatečného zapojení nebo naopak přepracování zaměstnance. Dlouhodobé a citelné vyšší pracovní nasazení může vyvrcholit v nežádoucí události, například ve vyčerpanost, nechuť či syndrom vyhoření. Na základě stráveného času lze tento trend sledovat a podle potřeb na něj reagovat.

Pro zaměstnance je strávený čas důležitý také v kontextu s výplatou a s tím souvisejícími situacemi, například čerpanou dovolenou nebo přesčasy daného zaměstnance.

4.2 Kritéria pro výběr komponent

Tato kapitola obsahuje rozbor zvolených kritérií, pomocí kterých se bude řídit výběr vhodných nástrojů a technologií. Na základě stanovených kritérií budou vybrána taková

řešení, která budou nejlépe splňovat uvedené požadavky. Kritéria byla stanovena na základě konzultace s managementem zvolené firmy.

Podporované platformy

Jelikož se předpokládá sběr dat z operačních systémů Windows a Linux, musí být možné provozovat nástroj na těchto systémech. Další operační systémy budou pouze bonusem.

Open source

Preferovaným řešením na základě konzultace s firmou bude skutečnost, že je nástroj dostupný jako open source. Tyto nástroje poskytují vhled do zdrojového kódu aplikace a lze tak zjistit, zda je nástroj napsán přijatelným způsobem, např. dle *best-practices* daného jazyka či neodporuje návrhovému vzoru. Open source nástroje také poskytují možnost upravit si nástroj dle vlastních potřeb.

Některé projekty mohou trpět na nedostatečnou velikost komunity o to víc, pokud jejich vývoj závisí na přispěvatelích. Zde vyvstává otázka, zda se vyplatí investovat zdroje do takového produktu, nebo zda bude vhodnější poohlédnout se po alternativě.

Aktivnost projektu

U méně aktivních projektů roste riziko, že nástroj nebude dále vyvíjen, v horším případě se potýkáme dokonce s problémem, že nebudou opravovány bezpečnostní chyby. Zřídka aktivní projekty s sebou nesou také vyšší pravděpodobnost, že za nimi nebude stát příliš velká komunita a tím pádem může být obtížnější řešit případné problémy nebo specifické požadavky.

U komerčních produktů lze metriku měřit pomocí hodnocení nástroje a společnosti vyvíjející nástroj, auditů společnosti a aktivity komunity. U open source nástrojů se mnohdy setkáváme s nedostupnými audity, a proto je možné metriku měřit prostřednictvím aktivity komunity, rychlosti vydávání nových verzí či oprav a velikosti vývojářského týmu.

API

Jedním ze sledovaných kritérií je API. Dostupné, kvalitně zpracované a zdokumentované API je základem pro snadnější vývoj integrovaného systému. Pokud by zvolený nástroj v základu neposkytoval integraci s jinými nástroji, je tento ukazatel o to podstatnější. V případě absence integrace s požadovanými nástroji je nutné, aby daný produkt měl přístupné API pro import a export dat.

Import a export dat

Vybraný nástroj musí umožňovat import a export dat, jelikož s daty je potřeba dále pracovat. Očekává se požadavek na další zpracování či agregaci dat v dalších nástrojích. Zkoumáno bude nejen zda nástroj umí importovat a exportovat data, ale také jakým způsobem je možné tyto operace provést a jaké formáty poskytuje.

Import dat bude sledován zejména u nástrojů sloužící k analýze dat.

Export dat bude sledován především u nástrojů sloužících k měření a zaznamenávání času. Bude zkoumáno, zda získaná zdrojová data lze rovnou exportovat a jestli je možné exportovat všechna dostupná data. Některá řešení poskytují omezený export dat, a to jak z hlediska zvoleného technického řešení, tak s ohledem na vybrané předplatné daného nástroje.

Získaná data však nemusí být v uživatelsky přívětivé podobě, jelikož budou zpracována strojově.

Integrace

Zvolený nástroj by měl poskytovat integraci se systémem Redmine. Jako výhoda při výběru nástroje bude vnímána integrace s dalšími systémy, díky čemuž by nebylo potřeba, při případném požadavku na rozšíření, vyvíjet další vlastní integrace. Každá poskytovaná integrace zvyšuje znovupoužitelnost řešení. Množství integrací mnohdy může korelovat s popularitou nástroje. Především u komerčních produktů je v zájmu firmy podporovat co nejvíce systémů, neboť to usnadňuje zapojení nástroje do existujícího ekosystému.

Cena a financování

Kritérium ceny a financování bude sledováno především u komerčních produktů. Některé produkty bývají dostupné zdarma, jiné mohou být poskytovány s bezplatnou, ale časově omezenou verzí na vyzkoušení. Placené služby typicky mají formu předplatného, jenž se odvíjí od velikosti společnosti, resp. počtu lidí využívajících službu, či od funkcí, které dané předplatné poskytuje. Ceny jednotlivých řešení se mohou výrazně lišit a při sestavování praktického řešení může jít o důležitý faktor ovlivňující výběr nástrojů.

4.3 Výběr IoT senzoru

Na základě průzkumu v teoretické části v kapitole 3.6 Podpůrné prostředky pro sledování aktivity a v souvislosti s použitím bylo pro většinu typů senzorů vybráno několik zástupců. Vybraní zástupci podle typu senzoru, cenová kalkulace a finálně vybraný senzor jsou tématem této kapitoly.

4.3.1 Dostupné senzory

S ohledem na cenu a účel použití budou v práci uvažovány ultrazvukové, zvukové a infračervené senzory. Vzhledem k povaze diplomové práce není na přesnost kladen vysoký důraz, i tak je u většiny níže zmíněných senzorů přesnost ± 1 %.

Zvukové senzory

Mezi zvukové senzory jsou pro příklad a ukázku vybráni dva zástupci, a to zvukový senzor s LM393 a LM386. Vzdálenost, ve které jsou schopny detekovat objekt a frekvence jsou u obou uváděny identicky. Vlastnosti jsou zapsány v tabulce 1. Tyto informace jsou čerpány z webu rpishop.cz a botland.cz.

Název	**Vzdálenost**	**Frekvence**
Zvukový senzor s LM393	až do 50 cm	16 - 20 kHz
Zvukový senzor s LM386	až do 50 cm	16 - 20 kHz

Tabulka 1: Vlastnosti vybraných zvukových senzorů

Ultrazvukové senzory

Pro ultrazvukové senzory se mezi klíčové vlastnosti řadí vzdálenost, tedy v jakém dosahu se musí senzor od pozorovaného objektu nacházet. Tyto hodnoty jsou u pozorovaných a níže uvedených senzorů velice podobné. Další vlastnost, která je u senzorů níže taktéž téměř identická, je přesnost.

Název	**Vzdálenost**	**Přesnost**
HC-SR04	2 - 400 cm	± 0,3 cm
HY-SRF05	2 - 450 cm	až do 0,3 cm
US-100	2 - 450 cm	0,3 cm + 1 %

Tabulka 2: Vlastnosti vybraných ultrazvukových senzorů

Infračervené senzory

Pro ukázku infračervených senzorů byli opět vybráni dva zástupci, oba od výrobce Sharp Corporation. Senzory detekují předměty v určité vzdálenosti a jejich výstupem je analogový signál. Tato hodnota poté závisí na vzdálenosti mezi senzorem a detekovaným objektem.

Název	Rozsah měření	Vlnová délka	Přesnost
Sharp GP2Y0A41SK0F	4 - 30 cm	870 nm	± 1 %
Sharp GP2Y0A21YK0F	10 - 80 cm	850 nm	± 1 %

Tabulka 3: Vlastnosti vybraných infračervených senzorů

Eye-tracking

Jak již bylo zmíněno v teoretické části v kapitole 3.6.4 Eye-tracking, tento typ senzoru nebude v rámci této diplomové práce uvažován. Příklad pro tento typ senzoru byl již i s aktuální cenou uveden v teoretické části práce.

4.3.2 Cenová kalkulace

Uváděné senzory jsou dostupné na různých webech. Cenová dostupnost vybraných senzorů, jenž jsou uvedené v kapitole 4.3.1 Dostupné senzory, je předmětem tabulky 4. Každý web má pro vybraný senzor uvedenou svou cenu, rozdíly jsou mnohdy znatelné. Všechny ceny uvedené níže a také ceny v tabulce 4 a 5 jsou včetně DPH.

Jako příklad může posloužit ultrazvukový senzor *HC-SR04*. Ten lze na webu rpishop.cz zakoupit za 39 Kč, u tohoto senzoru je uveden maximální dosah 400 cm. Web botland.cz nabízí dokonce dva senzory označené jako *HC-SR04*. První má uveden maximální dosah 400 cm, tedy s největší pravděpodobností odpovídá tomu z rpishop.cz, ale je za více než trojnásobnou cenu, a to 129 Kč. U druhého je uveden maximální dosah 200 cm, ten je na webu za cenu 46 Kč, tedy cenově blíže senzoru z rpishop.cz.

Název	Typ senzoru	Cena	Obchod
Zvukový senzor s LM393	zvukový	49 Kč	rpishop.cz
Zvukový senzor s LM386	zvukový	153 Kč	botland.cz
HC-SR04	UV	39 Kč	rpishop.cz
HY-SRF05	UV	49 Kč	rpishop.cz
US-100	UV	139 Kč	rpishop.cz
Sharp GP2Y0A41SK0F	IR	199 Kč	dratek.cz
Sharp GP2Y0A21YK0F	IR	668 Kč	botland.cz

Tabulka 4: Cenová kalkulace vybraných senzorů

Kromě samotného senzoru je nutné zajistit také vývojovou desku, na kterou se bude daný senzor připojovat. Pro tyto účely je potřeba vývojová deska obsahující Wi-Fi modul. Na základě zmíněných požadavků připadá v úvahu vývojová deska založená na čipu ESP8266. Výrazné cenové rozdíly, jako tomu bylo u senzorů v případě vývojové desky, nejsou pozorovány, i když se ceny na různých webech opětovně liší. Vývojové desky určené pro ukázku jsou zahrnuty v tabulce 5.

Název	**Cena**	**Obchod**
NodeMCU ESP8266 WiFi vývojová deska	179 Kč	rpishop.cz
NodeMcu CP2102 Lua WI-fi ESP8266	147 Kč	dratek.cz

Tabulka 5: Cenová kalkulace vybraných vývojových desek

Senzor musí být umístěn do krytu, jenž je možné vytisknout na 3D tiskárně. Jelikož firma 3D tiskárnou disponuje, bylo uvažováno toto řešení. Model krytu pro vybraný senzor byl připraven v aplikaci Autodesk Fusion 360.

4.3.3 Závěr

Jelikož se jedná pouze o doplňkový prostředek k ještě detailnějšímu záznamu aktivity, bude dostačovat senzor z nižší cenové relace. Je očekáván případný nákup přibližně dvaceti takových senzorů, proto byl brán ohled také na poměr ekonomické stránky a přínosu tohoto zařízení. Také dostupnost daného senzoru je v současné době, tedy v lednu 2023, bez výpadků, proto by s případným pořízením neměl být žádný problém. Pro účely diplomové práce a zároveň pro naplnění požadavků zadavatelem, firmou Orchitech Solutions, s.r.o., byl vybrán senzor *HC-SR04*. Využití a účel senzoru byl konzultován s technikem z obchodu rpishop.cz, kterým byl vybraný senzor taktéž doporučen.

Souhrnně by senzory *HC-SR04* z webu rpishop.cz pro 20 uživatelů vyšly na 780 Kč s DPH. Pro každý senzor je nutné zakoupit taktéž vývojovou desku, jenž by byla přidána k objednávce senzorů. Za vývojové desky pro 20 uživatelů by v současné době bylo účtováno 3 580 Kč s DPH. Ceny jsou platné ke dni 14.01.2023.

Dále je senzory nutné umístit do krytu, který při tisku na již vlastněné 3D tiskárně vyjde přibližně na 10 Kč včetně započtené spotřeby energií.

Celková náklady na IoT senzory při započtení produktů výše zmíněných a současně sepsaných v tabulce 6 pro firmu s 20 zaměstnanci vychází na 4 560 Kč.

Produkt	Cena za jednotku	Cena za 20 ks	Obchod
Senzor HC-SR04	39 Kč	780 Kč	rpishop.cz
Vývojová deska	179 Kč	3 580 Kč	rpishop.cz
Kryt	10 Kč	200 Kč	vlastní 3D tisk

Tabulka 6: Cenová kalkulace všech komponent

4.4 Výběr nástroje

V teoretické části práce byli představeni zástupci nástrojů jak pro manuální, tak i pro automatické sledování aktivity. U všech představených nástrojů byla zvážena kritéria uvedená v kapitole 4.2 Kritéria pro výběr komponent. Všechny prozkoumané nástroje umožňují měřit čas strávený aktivitami uživatele. Některé přistupují k problematice formou manuálního měření, jiné pomocí automatického měření a následné kategorizace. U některých nástrojů je zřejmé, jak se se získanými daty nakládájí, jiné nástroje mají tuto problematiku méně transparentní.

4.4.1 Závěr

Po zvážení kritérií a výsledků analýzy dostupných nástrojů byl jako vhodný nástroj vybrán ActivityWatch. Ačkoliv v základu neposkytuje integraci s Redmine nebo aplikací TTRebel, je snadno rozšiřitelný, přizpůsobitelný potřebám uživatele a má přístupné REST API, které umožní rozšířit nástroj o vlastní integraci.

Dále byl vybrán nástroj TTRebel, jelikož se jedná o již existující řešení vyvinuté potřebám zaměstnanců firmy Orchitech Solutions, s.r.o., je ho možné dle potřeb pro automatický time tracking dále upravit. Výhodou tohoto nástroje je také fakt, že zaměstnanci jsou zvyklí jej používat a po vzhledové i funkční stránce jim vyhovuje.

4.5 Požadavky na navrhované řešení

Kritéria, která by navrhované řešení mělo splňovat, byla stanovena v kapitole 4.2 Kritéria pro výběr komponent. V této kapitole budou uvedeny doplňující požadavky na navrhované řešení. Požadavky byly stanoveny na základě diskuse s managementem společnosti Orchitech Solutions, s.r.o. a s ohledem na získané závěry z analýzy v kapitole 3.5 Dostupné nástroje.

Doplňující požadavky na navrhované řešení jsou následující:

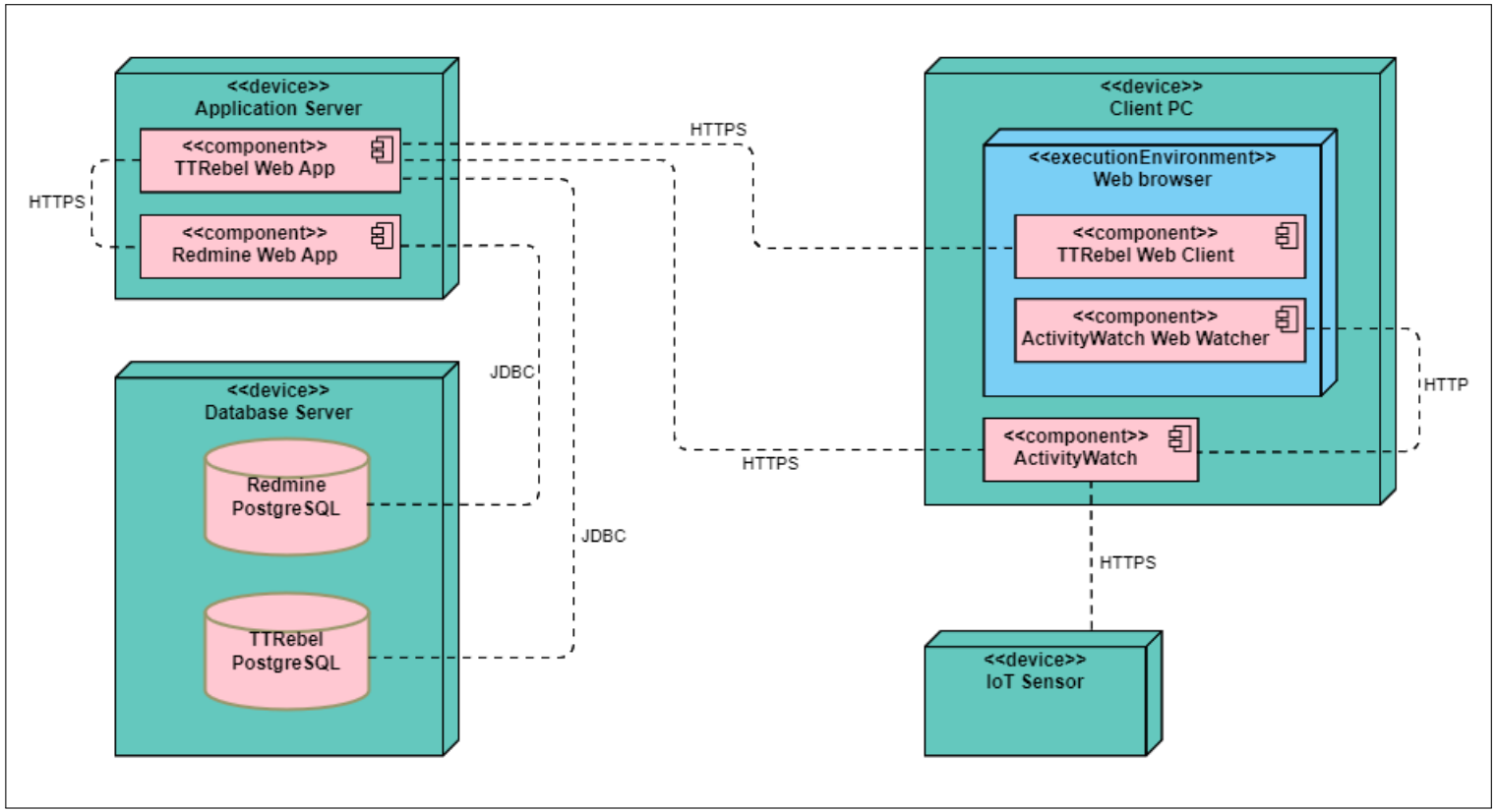
- Řešení bude poskytovat synchronizaci mezi aplikacemi TTRebel a ActivityWatch jak *real-time*, tak zpětně.
- Řešení bude poskytovat možnost sledovat absenci uživatele pomocí vhodného IoT senzoru.
- Řešení pomůže uživateli s přiřazováním aktivity ke konkrétním Redmine *issues*.
- Řešení pomůže uživateli s přiřazením konkrétní Redmine *issue* k relevantnímu úseku odpracovaného času.
- Řešení dokáže automaticky nebo s drobnými manuálními zásahy kategorizovat aktivitu do definovaných kategorií.
- Řešení bude automaticky monitorovat aktivitu uživatele a následně dokáže alespoň hrubě zrekonstruovat pracovní den na úrovni času stráveného na konkrétních Redmine *issues*, nebo alespoň na úrovni času stráveného kategorizovanými činnostmi.

4.6 Návrh architektury řešení

Architektura navrženého řešení se skládá z těchto pěti komponent:

- nástroj ActivityWatch,
- rozšíření webového prohlížeče,
- senzor pro měření přítomnosti,
- webová aplikace TTRebel,
- projektový systém Redmine.

Sběr dat o aktivitě zaměstnance bude probíhat na lokálním zařízení daného uživatele pomocí ActivityWatch. Senzor pro měření přítomnosti bude sloužit jako suplement k softwarovému měření přítomnosti nabízeného v rámci ActivityWatch. Po konzultaci s managementem společnosti Orchitech Solutions, s.r.o. nebude využita metoda eye-trackingu. *Tracking sessions* z webové aplikace TTRebel budou synchronizovány do ActivityWatch. Řešení poskytuje uživateli plnou kontrolu nad získanými daty, které neopustí jeho zařízení.



Obrázek 11: Diagram nasazení - vlastní zpracování

4.6.1 ActivityWatch

Nástroj ActivityWatch bude sloužit jako agregátor dat z integrovaných systémů. Provoz tohoto nástroje je zamýšlen zcela na bázi lokálního spouštění na zařízení daného uživatele. Tímto bude zajištěna plná kontrola nad daty. Uživatel má možnost upravit si nástroj dle svých potřeb a preferencí.

Nástroj bude monitorovat několik událostí:

- aktivní okno,
- aktivní kartu webového prohlížeče,
- Google Kalendář uživatele,
- přítomnost uživatele,
- nástroj Visual Studio Code,
- nástroje IntelliJ IDEA a WebStorm,
- nástroj TTRebel.

Aktivní okno

Monitorování aktivního okna bude zajištěno pomocí *watcher aw-watcher-window*. Monitorování bude realizováno kontinuálně po celou pracovní dobu.

Aktivní karta webového prohlížeče

Kontrola aktivní karty webového prohlížeče bude zajištěna pluginem, který bude blíže specifikován v kapitole 4.6.2 Rozšíření webového prohlížeče. Pro tento účel slouží *watcher aw-watcher-web*.

Google Kalendář

Monitorování Google Kalendáře uživatele bude zajištěno pomocí *importer aw-import-ical*. Import událostí uživatele bude realizován jednou denně.

Přítomnost uživatele

Sběr informací pro stranovení přítomnosti uživatele bude monitorován dvěma způsoby. Prvním způsobem bude *watcher aw-watcher-afk*, který bude monitorovat přítomnost uživatele pomocí aktivity vstupních zařízení. Druhý způsob, IoT senzor, je blíže specifikován v kapitole 4.6.3 Senzor pro měření přítomnosti.

Visual Studio Code

Nástroj Visual Studio Code, jakožto nejčastěji využívané vývojové prostředí ve firmě, bude v rámci zamýšleného řešení monitorován pomocí vlastního *watcher aw-watcher-vscode*. Tento *watcher* bude poskytovat data o názvu projektu, upravovaném souboru a jeho jazyce.

Pro účely práce bude také nutné rozšířit funkcionalitu o získání názvu větve z verzovacího nástroje Git. Tato informace povede k usnadnění zjištění, na které úloze vývojář pracuje, jelikož součástí nastavených a vyžadovaných konvencí ve společnosti Orchitech Solutions, s.r.o. je pojmenování větve podle identifikátoru *issue* v Redmine. Tato funkcionalita může být klíčová, jelikož pomůže uživateli v případné korekci či nalezení příslušné *tracking session*.

IntelliJ IDEA a WebStorm

Mezi další využívaná vývojová prostředí ve firmě patří nástroje společnosti JetBrains, a to IntelliJ IDEA a WebStorm. Tyto nástroje budou monitorovány pomocí *watcher aw-watcher-jetbrains*, jenž poskytuje data především o názvu projektu, upravovaném souboru a názvu větve z verzovacího nástroje Git.

TTRebel

V plánovaném řešení budou synchronizována data z nástroje TTRebel do ActivityWatch. Synchronizace dat bude probíhat jak v reálném čase, tak i zpětně. Synchronizace v reálném čase bude sledovat aktivní *tracking session*. Zpětná synchronizace umožní uživateli importovat všechna známá data pro zvolené časové období do nástroje ActivityWatch.

V současné době neexistuje nástroj, který by požadované funkcionality poskytoval. Během implementace proto vznikne *watcher aw-watcher-ttrebel*, který umožní synchronizovat data v reálném čase. Dále vznikne *importer aw-importer-ttrebel*, jenž zajistí informace pro zpětnou synchronizaci dat.

4.6.2 Rozšíření webového prohlížeče

Webový prohlížeč bude rozšířen o plugin *ActivityWatch Web Watcher*, který je dostupný pro prohlížeče Google Chrome a Mozilla Firefox. Toto rozšíření bude monitorovat aktivní kartu webového prohlížeče.

4.6.3 Senzor pro měření přítomnosti

Další metodou pro měření přítomnosti uživatele bude vybraný IoT senzor, který bude monitorovat přítomnost uživatele pomocí zvolené technologie. Monitorování bude zajištěno pomocí *watcher aw-watcher-table*, který bude upraven dle potřeb. Skutečnost vyplývající z průniku dat této množiny a množiny z kapitoly 4.6.1 ActivityWatch sekce Přítomnost uživatele bude indikovat, zda-li je uživatel přítomen u zařízení či nikoliv.

4.6.4 TTRebel

Nástroj TTRebel bude nadále sloužit k vytváření a správě *tracking sessions*, jak jsou uživatelé zvyklí. V rámci smluveného řešení je uvažován pouze jednosměrný export dat z nástroje TTRebel do nástroje ActivityWatch.

4.6.5 Redmine

Projektový systém Redmine bude nadále určen pro správu projektů a jako cílový systém pro výkazy. Veškerá potřebná funkcionalita pro práci s výkazy je již obsažena v aplikaci TTRebel. V souvislosti se zamýšleným řešením nebude nutné implementačně upravovat systém Redmine.

4.7 Implementace řešení

Představení implementace řešení s ukázkami a příklady u jednotlivých komponent, a to ActivityWatch a TTRebel a současně změny v souvislosti s vybraným senzorem jsou předmětem této kapitoly.

4.7.1 ActivityWatch

V rámci této kapitoly budou specifikovány implementační detaily jednotlivých modulů nástroje ActivityWatch. Dále bude rozebrána kategorizace aktivit a sestavení finálního řešení nástroje ActivityWatch.

Moduly sloužící k získání dat z aplikace TTRebel jsou implementovány v programovacím jazyce Python. Uvedený jazyk byl zvolen z toho důvodu, že jak samotný ActivityWatch, tak i téměř všechny oficiální moduly, jsou implementovány právě v uvedeném jazyce. Dalším důvodem byla skutečnost, že klient implementovaný v jazyce TypeScript (*aw-client-js*) je oproti klientovi implementováném v jazyce Python (*aw-client*) téměř neudržovaný a není dostupná téměř žádná dokumentace.

aw-ttrebel-core

V průběhu vývoje modulů *aw-watcher-ttrebel* a *aw-importer-ttrebel* bylo zjištěno, že sdílejí nezanedbatelnou část zdrojového kódu, a proto byl tento kód vyčleněn do samostatné knihovny, která je následně importována do jednotlivých modulů.

Za transformaci objektů `TrackingSession` a `TrackingInterval` z aplikace TTRebel na `Event`, jenž ActivityWatch používá jako entitu události, je zodpovědná metoda `map` třídy `SessionMapper`. Způsob mapování hodnot atributů byl zvolen tak, aby objekt obsahoval pouze potřebné informace pro automatickou kategorizaci.

```
@staticmethod
def map(
    session: TrackingSession, interval: TrackingInterval) -> Dict[str, str]:
  return {
    "tracking_session_id": session.id,
    "issue_id": session.issue_id,
    "system": session.system,
    "issue_name": session.issue_name,
    "description": session.description,
    "tracking_interval_id": (interval.id if interval is not None else None),
    "title": "[{}] {} #{}: {}".format(
        session.project_name,
        session.tracker_name,
        session.issue_id,
        session.issue_name),
  }
```

Kód 2: Implementace metody *map*

Sdílenou logiku modulů poskytuje třída `TTRebelClient`, která obsahuje především metody pro autentizaci a pro získávání *tracking sessions* nebo *tracking intervals*.

Metoda `get_sessions` vrací všechny *tracking sessions* pro zvolené období, lze pro ní nastavit parametry `created_after` a `created_before`. Metoda `get_intervals` vrací všechny *tracking intervals* pro zvolenou *tracking session*, kterou lze nastavit parametrem `tracking_session_id` a jehož hodnota reflektuje unikátní identifikátor *tracking session* v aplikaci TTRebel. Obě uvedené metody pro případ velkého objemu dat podporují stránkování pomocí parametrů `page_size` a `page_offset`. Třída dále obsahuje metodu `get_active_session`, jenž vrací aktuální aktivní *tracking session* v aplikaci TTRebel.

aw-watcher-ttrebel

Tento modul poskytuje synchronizaci dat z aplikace TTRebel do ActivityWatch v reálném čase. *Watcher* využívá API třídy `TTRebelClient` pro vyhledání aktivní *tracking session* a API třídy `ActivityWatchClient` pro persistenci dat. *Watcher* v cyklu vyhledává aktivní *tracking session*, je-li navrácen výsledek, dochází k namapování dat na *event*, který je následně persistován pomocí *heartbeat* do *bucket* ActivityWatch. Ve výchozí konfiguraci dochází k vyhledání aktivní *tracking session* každých deset sekund.

aw-importer-ttrebel

Modul *aw-importer-ttrebel* slouží k synchronizaci dat z aplikace TTRebel do ActivityWatch pro zvolené časové období. ActivityWatch v současné verzi neposkytuje API pro editaci *events*, která by byla ideálním řešením pro řešení kolize při importu záznamů, které již *importer* v datech obsahuje. Z tohoto důvodu bylo přistoupeno k řešení, kdy se pro zvolený časový úsek data *importer* vymažou a až poté proběhne samotný import dat. Ukázky 3 a 4 poskytují vhled do navrženého řešení.

Importer očekává na vstupních parametrech buď datumový interval počátku a konce importovaných dat, nebo počáteční datum. V případě využití druhé varianty dojde k importu dat od poskytnutého data až po současný den.

Za odstranění již existujících dat *importer* je zodpovědná metoda `delete_events` třídy `TTRebelImporter`. Metoda za pomoci třídy `ActivityWatchClient` poskytující API pro práci s persistovanými daty *importer*, vyhledává záznamy, které odpovídají poskytnutým identifikátorům `TrackingInterval` a danému časovému úseku. Pro optimalizaci výkonu se vyhledávají záznamy pro jednotlivé dny, dokud není dokončený průchod všemi dny daného časového úseku. ActivityWatch neposkytuje API pro hromadné odstranění *events* s možností definovat omezující parametry, z tohoto důvodu jsou záznamy mazány sekvenčně. Implementace metody `delete_events` je zobrazena v ukázce 3.

```
def delete_events(self, created_after: datetime,
    intervals: List[TrackingInterval]) -> None:
  start_date = created_after
  end_date = start_date + timedelta(days=1)
  count = 0

  interval_ids = []
  for interval in intervals:
    interval_ids.append(interval.id)
```

```
  while start_date <= datetime.now().replace(
      tzinfo=timezone(offset=timedelta())):
    # fetch all events for given interval IDs
    result = self.aw_client.query(
        query="events = query_bucket(find_bucket('{a}'));\n
        RETURN=filter_keyvals(events, 'tracking_interval_id',
        [{b}]);".format(
            a=self.bucket_name, b=','.join(map(str, interval_ids))),
        timeperiods=[(start_date, end_date)])[0]
    logger.info("Found {} events between {} and {}.".format(
        len(result), start_date, end_date))
    events = [Event(**event) for event in result]

    for event in events:
      logger.info("Deleting event {}.".format(event.id))
      self.aw_client.delete_event(self.bucket_name, cast(int, event.id))
      count += 1

    start_date = end_date
    end_date = start_date + timedelta(days=1)
  logger.info("Deleted {} events between {} and {}.".format(
      count, created_after, datetime.now()))
```

Kód 3: Implementace metody *delete_events*

Za import dat je zodpovědná metoda `import_intervals` třídy `TTRebelImporter`. Metoda využívá API třídy `TTRebelClient` pro vyhledání *tracking sessions* a *tracking intervals* a API třídy `ActivityWatchClient` pro persistenci dat.

V úvodu metody je vytvořen *bucket*, do kterého budou persistována data a následně dojde k vyhledání *tracking sessions*. Pro optimalizaci výkonu je nastaveno omezení počtu vrácených výsledků na třicet. Metoda prochází sekvenčně všechny vrácené *tracking sessions* a pro každou hodnotu vyhledá *tracking intervals*. Následně dochází k odstranění již existujících *events* pomocí metody `delete_events` uvedenou v předchozím odstavci, poté jsou všechny *tracking intervals* transformovány na *events*.

Pokud je vrácené množství *tracking sessions* menší než velikost stránky, znamená to, že byly navráceny všechny existující záznamy pro zvolená kritéria a import je ukončen, v opačném případě jsou vyhledány další *tracking sessions* a proces se opakuje. Implementace metody `import_intervals` je zobrazena v ukázce 4.

```
def import_intervals(self, created_after: datetime) -> None:
  assert created_after is not None
```

```
event_type = "ttrebel_importer"
self.aw_client.create_bucket(self.bucket_name, event_type)
sleep(1) # wait for server to start
page_size = 30
page_offset = 0
session_count = 0
interval_count = 0
# fetch the first batch of tracking sessions from TTRebel
sessions = self.ttrebel_client.get_sessions(
    page_size=page_size,
    page_offset=page_offset,
    created_after=created_after)

while True:
  events:List[Event] = []
  session_count += len(sessions)
  for session in sessions:
    intervals = self.ttrebel_client.get_intervals(
        page_size=100, tracking_session_id=session.id)
    self.delete_events(created_after, intervals)

    for interval in intervals:
      events.append(self.map_interval(session, interval))

    interval_count += len(intervals)
  self.aw_client.insert_events(self.bucket_name, events)
  logger.debug("{} events imported to ActivityWatch.".format(
      len(events)))

  page_offset += 1
  if len(sessions) <= page_size:
    break
  else:
    sessions = self.ttrebel_client.get_sessions(
        page_size=page_size,
        page_offset=page_offset,
        created_after=created_after)
logger.info("{} sessions and {} intervals were imported to
    ActivityWatch.".format(session_count, interval_count))
```

Kód 4: Implementace metody *import_intervals*

aw-watcher-afk

Stávající implementace obsahuje všechny potřebné funkce. V rámci implementace nebylo nutné tento *watcher* upravovat.

aw-watcher-vscode

V původní implementaci se při inicializaci, otevření editovatelného souboru či po vypršení časového limitu získává název projektu, jazyk a název daného souboru. Jelikož je IDE Visual Studio Code převážně používáno vývojáři, kteří pracují s verzovacími nástroji jako je například Git, bylo by vhodné, kdyby *watcher* uměl také získat název větve z Gitu.

Watcher aw-watcher-vscode je napsaný v jazyce TypeScript, ke komunikaci se serverem využívá klienta *aw-client-js*, jenž je psaný v jazyce JavaScript. Obě knihovny běží v Node.js. V rámci předimplementační analýzy bylo určeno, že nástroj pro získání doplňujících informací z verzovacího nástroje Git musí být kompatibilní s Node.js. V průběhu předimplementační analýzy bylo dále zjištěno, že pro získání doplňujících informací z verzovacího nástroje Git nabízí IDE Visual Studio Code integrovanou integrovanou podporu. Současně bylo zjištěno, že doplňující informace lze také získat za pomoci knihovny *simple-git*. Jelikož lze vyřešit danou problematiku již integrovanou funkcionalitou, bylo přistoupeno k tomuto řešení. Vybrané řešení bude také vhodnější pro akceptaci změn autorem *aw-watcher-vscode*, kterému byly poskytnuty provedené implementační úpravy v rámci pull requestu *Send Git branch name. #26.* (Bartoš, 2021)

Původní implementace byla rozšířena tak, že při inicializaci, změně větve, otevření editovatelného souboru nebo po vypršení časového limitu se získává také název větve v Gitu společně se stávajícími atributy.

aw-watcher-jetbrains

V případě, že uživatel nebude používat aplikaci Visual Studio Code, jenž má svůj vlastní *watcher* zmíněný výše, ale bude používat jiné vývojové prostředí, a to například IntelliJ IDEA nebo WebStorm, je připraven *watcher* starající se o získávání informací z aplikací JetBrains. Tento *watcher* již v základu obsahuje veškerou implementaci potřebných funkcí, nebylo proto nutné *watcher aw-watcher-jetbrains* upravovat.

aw-importer-ical

Tento *importer*, který získává informace z Google Kalendáře, nebylo potřeba dále rozšiřovat či upravovat, jelikož ve stávající implementaci obsahuje veškeré funkce, které budou pro účely této práce potřeba.

aw-watcher-web

Všechny funkce, které jsou pro *watcher* sledující okno, tedy *aw-watcher-web*, potřebné, jsou již v současné implementaci obsaeny, nebylo proto nutné tento *watcher* upravovat.

aw-watcher-window

Watcher zodpovědný za informace z aktivního okna, byl v dostatečném rozsahu funkcí připraven již před zpracováním této práce. V souvislosti s tímto *watcher* proto nebyly nutné žádné dalš úpravy.

Kategorizace aktivit

ActivityWatch ve výchozí konfiguraci poskytuje čtyři kategorie. Každá kategorie může obsahovat pravidlo ve formě regulárního výrazu, pomocí kterého je aktivita zařazena do příslušné kategorie. Zároveň ActivityWatch podporuje vytváření podkategorií, které blíže specifikují danou kategorii.

Následující přehled zobrazuje výčet výchozích kategorií ve formátu *Název kategorie - regulární výraz*.

- Work - `Google Docs|libreoffice|ReText`
  - Programming - `GitHub|Stack Overflow|BitBucket|Gitlab|vim|Spyder|kate|Ghidra|Scite`
    - ActivityWatch - `ActivityWatch|aw-`
  - Image - `Gimp|Inkscape`
  - Video - `Kdenlive`
  - Audio - `Audacity`
  - 3D - `Blender`
- Comms
  - IM - `Messenger|Telegram|Signal|WhatsApp|Rambox|Slack|Riot|Discord|Nheko`
  - Email - `Gmail|Thunderbird|mutt|alpine`

- Media
  - Games - `Minecraft|RimWorld`
  - Video - `YouTube|Plex|VLC`
  - Social Media - `reddit|Facebook|Twitter|Instagram|devRant`
  - Music - `Spotify|Deezer`
- Uncategorized

Ačkoliv výchozí kategorie poskytují hrubý přehled o tom, zda se uživatel věnoval pracovním činnostem, případně kterým, nebo se pracovním činnostem nevěnoval, není dostatečně konkrétní. Pro tento účel ActivityWatch umožňuje importovat vlastní sadu kategorií. Import kategorií do ActivityWatch je možné realizovat pomocí webového rozhraní, jenž akceptuje soubory ve formátu JSON, viz ukázka 5.

```
{
  "categories": [
    {
      "name": [
        "Work"
      ],
      "rule": {
        "type": "regex",
        "regex": "Google Docs|libreoffice|ReText"
      },
      "data": {
        "color": "#0F0"
      },
      "id": 0
    }
  ]
}
```

Kód 5: Ukázka požadovaného formátu kategorie

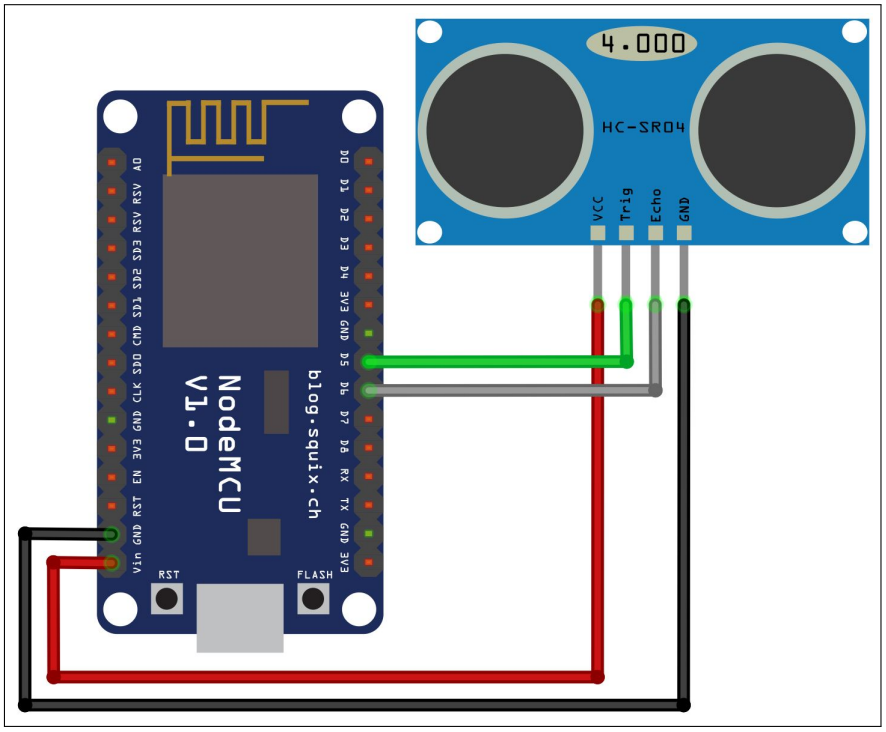
Současná implementace nástroje však importované kategorie persistuje pouze do lokální paměti prohlížeče, tudíž při každém smazání mezipaměti prohlížeče dochází ke ztrátě naimportovaných kategorií. Sestavení vlastních kategorií je blíže specifikováno v kapitole 4.10 Automatická kategorizace.

4.7.2 Senzor pro měření přítomnosti

Řešení pro měření přítomnosti se skládá z hardware části, jenž je sestaveno pomocí komponent dle výstupů kapitoly 4.3.3 Závěr, a software části v podobě *watcher aw-watcher-table*. V závěru této kapitoly bude představena praktická ukázka sestaveného řešení pro měření přítomnosti uživatele.

Diagram zapojení

Diagram zapojení na obrázku 12 zobrazuje zapojení vývojové desky *NodeMCU ESP8266* a ultrazvukového senzoru *HC-SR04*. Zapojené vstupně-výstupní piny *D5* a *D6* jsou zohledněny v implementaci logiky potřebné k fungování vybraného hardware.



Obrázek 12: Diagram zapojení - vlastní zpracování

Kryt

V souvislosti s implementací řešení byl také připraven kryt s držákem pro senzor a vývojovou desku ten byl pro účely práce následně vytištěn na FDM 3D tiskárně Voron V1.8. Pro vývoj krytu byl použit software Autodesk Fusion 360 s bezplatnou licencí pro nekomerční použití. Tiskové soubory jsou součástí této práce jako Příloha A.

aw-watcher-table

Modul *aw-watcher-table* se skládá ze dvou částí. První část obsahuje logiku potřebnou k provozu externího zařízení, v případě této práce jde o vývojovou desku *NodeMCU ESP8266* a ultrazvukový senzor *HC-SR04*. Druhá část obsahuje samotný *watcher*, který je stejně jako ostatní ActivityWatch moduly spuštěn na lokálním zařízení uživatele. Řešení vyžaduje internetové připojení obou součástí.

Původním záměrem autora tohoto *watcher* bylo použití pro monitorování polohy polohovatelného stolu, zda u počítače uživatel stojí či nikoliv. Díky koncepci, se kterou byl *watcher* implementován, stačí pouze upravit konfigurační soubor a poté lze použít toto řešení k monitorování přítomnosti uživatele. Pro účely práce tedy nebylo nutné zasahovat do logiky nástroje.

Konfigurace hodnot pro rozpoznání specifického stavu a jeho rozmezí vzdálenosti je realizována prostřednictvím atributu `height_levels`. Ve výchozím stavu je hodnota uvedeného atributu `"ball:0-70;sitting:70-90;standing:90-200"`, kde lze pozorovat, že stavy jsou definovány ve formátu *stav:rozmezí vzdálenosti* a jednotlivé stavy jsou odděleny středníkem. Na základě tohoto zjištění byla upravena konfigurace, kde byly definovány stavy `notAfk` a `afk` s výchozími hodnotami, které jsou zobrazeny v ukázce 6. Výchozí doporučené hodnoty může uživatel upravit dle potřeby pro optimální snímání jeho přítomnosti.

```
default_settings = {
  "poll_time": "5",  # seconds
  "height_levels": "notAfk:0-100;afk:101-400",  # cm
  "esp_ip": "192.168.2.4"
}
```

Kód 6: Upravená konfigurace *aw-watcher-table*

Ukázka sestaveného řešení

Kryt umožňuje měnit úhel náklonu dle potřeb uživatele. Úhel nasměrování je vhodné nastavit tak, aby senzor snímal pouze uživatele, nikoliv objekty za ním, například přisunutou pracovní židli ke stolu v případě absence uživatele. Vzdálenost a úhel pro získání adekvátního záznamu je potřeba otestovat a optimalizovat pro každého jedince.

Obrázek 13: Praktická ukázka sestaveného řešení - vlastní zpracování

4.7.3 TTRebel

V rámci implementace úprav aplikace TTRebel byly realizovány dvě hlavní změny. Aplikaci chyběl endpoint pro získání aktivní *tracking session* a *tracking sessions* neobsahovaly data o názvu *issue*, projektu a typu *issue*.

Aktivní *tracking session*

Pro získání aktivní *tracking session* bylo rozšířeno REST API o endpoint `/rest/2.0/trackingSessions/active` využívající metodu *GET* protokolu *HTTP*. Třída `TrackingSessionService` je zodpovědná za práci s *tracking sessions* na servisní vrstvě aplikace, zde byla implementována metoda `getActiveTrackingSession`. Metoda prostřednictvím datové vrstvy vyhledá takové *tracking sessions*, které jsou ve stavu `ACTIVE` a atribut `userId` odpovídá poskytnutému identifikátoru uživatele. Stav *tracking session* může nabývat stavu `NEW`, `ACTIVE`, `PAUSED` nebo `EXPORTED`. Aplikace umožňuje existenci nejvýše jedné *tracking session* ve stavu `ACTIVE` v daném okamžiku. Uvedené stavy jsou vypočítávány dynamicky při mapování dat z databáze. *Tracking session* nabývá stavu `ACTIVE` právě tehdy, když není exportovaná do Redmine (atribut `exportedOn` odpovídá hodnotě `NULL`) a některý z *tracking intervals* dané *tracking session* není ukončený (atribut `endTime` odpovídá hodnotě `NULL`).

Název a typ *issue*

Název *issue* není v databázi aplikace TTRebel persistován záměrně. V případě, že by se změnil název *issue* v Redmine a zároveň by byl název *issue* persistován v databázi TTRebel, mohlo by dojít k nekonzistenci dat. Implementace servisní třídy `TrackingSessionService` zodpovědné za práci s *tracking sessions* byla rozšířena o již existující servisní třídu `IssueService`, jenž je zodpovědná za práci s *issues* z projektového systému Redmine. Metody `getTrackingSession`, `getTrackingSessions` a `getActiveTrackingSession` třídy `TrackingSessionService`, které využívá modul *aw-ttrebel-core* prostřednictvím volání API, byly rozšířeny o volání metody `getIssueById` třídy `IssueService` sloužící k získání *issue*. Návratová hodnota obsahuje mimo jiné atributy `projectName` reprezentující název projektu, `trackerName` reprezentující typ *issue* a `subject` reprezentující název *issue*. Zmíněné atributy jsou následně namapovány na stejnojmenné atributy třídy `TrackingSession`.

4.8 Sběr dat

Pro sběr dat bylo využito jedno zařízení, a to zaměstnanecký notebook HP-ProBook-440-G5 s operačním systémem Kubuntu. Na tomto zařízení pracuje vývojář, jehož náplň práce je rozložena do několika projektů zároveň. Získaná data by měla pokrývat situace a činnosti, ze kterých je v určitém rozsahu možné vyvozovat závěry. Tato pravidla nejčastěji pokrývají požadavky pro zaměstnance na pozici vývojář. Bližší specifikace, například přizpůsobení kategorií dle pracovní pozice, nejsou součástí této práce a jsou zmíněny v kapitole 5.1 Limity. Sběr dat analyzovaných a níže interpretovaných byl realizován v období od 01.06.2022 do 31.08.2022.

4.9 Analýza dat

Cílem analýzy naměřených dat bylo zjistit, zda-li je možné a případně jakým způsobem rozčlenit získaná data. Zkoumáním naměřených dat bylo zjištěno, že je možné sestavit několik kategorií, a to:

- číslo *issue* v Redmine,
- zákazník,
- předdefinovaná aktivita.

Po konzultaci s managementem společnosti Orchitech Solutions, s.r.o. byly vzhledem k citlivosti dat informace týkající se projektů, názvů *issues*, názvů schůzek, zákazníků a zaměstnanců anonymizovány.

4.9.1 Číslo *issue* v Redmine

Číslo aktuálně zpracovávané *issue* je možné získat více způsoby. Zdrojem jsou za *watchers* *aw-watcher-ttrebel*, *aw-watcher-vscode*, *aw-watcher-jetbrains*, *aw-watcher-web* a *aw-watcher-window*. Za *importers* se jedná o *aw-importer-ttrebel* a *aw-importer-ical*. Bližší informace jsou uvedeny níže.

aw-watcher-ttrebel a aw-importer-ttrebel

Prvním způsobem pro získání čísla *issue* jsou data z nástroje TTRebel, avšak pouze za předpokladu, že *tracking session* má přiřazenou *issue*. Tato informace se vyskytuje v datech jak *aw-watcher-ttrebel*, tak i *aw-importer-ttrebel*. *Watcher* *aw-watcher-ttrebel* poskytuje informace ve formátu *[Zákazník] #39024: Testovací úloha*. Informaci o čísle *issue* také poskytuje *aw-importer-ttrebel*, který díky použité sdílené logice s *aw-watcher-ttrebel* obsahuje hodnotu ve stejném formátu. Číslo *issue* poskytují atributy *issue_id* a *title*, příklad této informace je znázorněn v ukázce 7.


```
"data": {
  "tracking_session_id": 101904,
  "issue_id": "43841",
  "system": null,
  "issue_name": "Politika pro minimalni delku hesla se nechova spravne",
  "description": "Analyza enduser, custom a vychozich politik, lokalni
    simulace, testovani s upravenou implementaci",
  "tracking_interval_id": 101906,
  "title": "[Zakaznik 5] #43841: Politika pro minimalni delku hesla
```

```
    se nechova spravne"
}
```

Kód 7: Ukázka dat *aw-importer-ttrebel* obsahující číslo *issue*

aw-watcher-vscode

Druhým způsobem je *watcher aw-watcher-vscode*, jenž přebírá informaci o čísle *issue* z názvu větve verzovacího systému Git, a to díky vyžadované konvenci v pojmenování větve podle čísla *issue* zmíněné v kapitole 4.6.1 ActivityWatch. Atribut, který v tomto případě poskytuje potřebnou informaci je *branch*, přičemž detailnější ukázka je znázorněna níže.

```
"data": {
  "language": "typescript",
  "project": "/Users/michal.bartos/development/fe-idm-core",
  "file": "/Users/michal.bartos/development/fe-idm-core/src/app/dashboard/
    service/dashboard.service.ts",
  "branch": "rm33189-momentjs-replacement"
}
```

Kód 8: Ukázka dat *aw-watcher-vscode* obsahující číslo *issue*

aw-watcher-jetbrains

Alternativou k výše zmíněnému *watcher* pro aplikaci Visual Studio Code je *watcher* pro aplikace JetBrains. Tento *watcher* získává informace o čísle *issue* z atributu *branch*, kdy v názvu větve ve verzovacím sytému Git je uložena potřebná informace. V ukázce níže je znázorněn příklad získaných dat z editoru WebStorm.

```
"data": {
    "file": "user-detail.component.ts",
    "fileFullPath": "/Users/michal.bartos/development/fe-idm-core/src/app/
      user/component/user-detail.component.ts",
    "project": "fe-idm-core",
    "projectPath": "/Users/michal.bartos/development/fe-idm-core",
    "language": "TypeScript",
    "editor": "webstorm",
    "editorVersion": "2022.3.1",
    "branch": "rm33189-momentjs-replacement",
    "commit": "3fnf08e02de595b325d40570a7555999c8db61b5",
}
```

Kód 9: Ukázka dat *aw-watcher-jetbrains* obsahující číslo *issue*

aw-importer-ical

V některých situacích se informace o čísle *issue* může vyskytovat také v datech *importer aw-importer-ical*. Typicky se jedná o případ, kdy událost v kalendáři je zaměřená na diskuzi ohledně konkrétní *issue*. Tato konvence ovšem ve firmě není pravidlem, proto ne na všechny události z Google Kalendáře je tento *importer* možné použít. V případě, že je vytvořená událost, jenž obsahuje číslo *issue*, je možné ji získat z atributu *title*, u kterého je očekávaný formát obsahující číslo *issue* a název schůzky.

```
"data": {
  "title": "#33189 - kickoff meeting",
  "attendees": [
    "user@orchitech.cz",
    ...
  ]
}
```

Kód 10: Ukázka dat *aw-importer-ical* obsahující číslo *issue*

aw-watcher-web

Dalším zdrojem obsahující informaci o čísle *issue* je *aw-watcher-web*, kde je možné hodnotu získat dvěma způsoby. První zdroj informace je možné nalézt v názvu karty, tedy atributu *title*, ve kterém se zobrazuje celý název dané *issue*. Druhým zdrojem je informace v URL, ve kterém se taktéž nachází číslo *issue*. Oba tyto případy pokrývá ukázka níže.

```
"data": {
  "url": "https://www.redmine.orchi.tech/issues/38226",
  "title": "Feature #33189: Nahrazeni knihovny momentjs - Redmine",
  "audible": false,
  "incognito": false,
  "tabCount": 2
}
```

Kód 11: Ukázka dat *aw-watcher-web* obsahující číslo *issue*

aw-watcher-window

V neposlední řadě se jedná o *watcher aw-watcher-window*, který zaznamenává název aktuálně aktivního okna. V případě, že se jedná o prohlížeč, poskytuje stejnou informaci jako *watcher aw-watcher-web*.

```
"data": {
  "app": "chrome.exe",
  "title": "Feature #33189: Nahrazeni knihovny
    momentjs - Redmine - Google Chrome"
}
```

Kód 12: Ukázka dat *aw-watcher-window* obsahující číslo *issue*

4.9.2 Zákazník

Pro správné určení určení toho, pro kterého zákazníka, respektive na kterém projektu, byl čas stráven, je vhodné znát právě zákazníka. Na základě těchto informací je zákazníkům předkládán a fakturován vykázaný čas. Taktéž tuto informaci je možné z aplikace ActivityWatch získat, a to díky *watcher aw-watcher-ttrebel* a *importers aw-importer-ttrebel* a *aw-importer-ical*.

aw-watcher-ttrebel a aw-importer-ttrebel

Zákazník, respektive projekt, je obsažen v datech z *aw-watcher-ttrebel* a *aw-importer-ttrebel*. Oba tyto zdroje zahrnují informaci o zákazníkovi v atributu *title*, který je ve formátu *[Zákazník] #39024: Testovací úloha*.

```
"data": {
  "tracking_session_id": 101904,
  "issue_id": "43841",
  "system": null,
  "issue_name": "Politika pro minimalni delku hesla se nechova spravne",
  "description": "Analyza enduser, custom a vychozich politik, lokalni
    simulace, testovani s upravenou implementaci",
  "tracking_interval_id": 101906,
  "title": "[Zakaznik 3] #43841: Politika pro minimalni delku hesla
    se nechova spravne"
}
```

Kód 13: Ukázka dat *aw-importer-ttrebel* obsahující zákazníka

aw-importer-ical

Dalším zdrojem informace o zákazníkovi je *importer* obsahující informace z Google Kalendáře, kterým je *aw-importer-ical*. Pro tento případ je nutné, aby byla dodržena konvence, kdy v názvu vytvářené události bude jméno zákazníka (projektu). Tato konvence je v současné době ve firmě poměrně využívána na rozdíl od méně používaného čísla *issue* v názvu události. Zákazníka je možné nalézt v atributu *title*.

```
"data": {
  "title": "Zakaznik 5 - stand up",
  "attendees": [
    "user@orchitech.cz",
    ...
  ]
}
```

Kód 14: Ukázka dat *aw-importer-ical* obsahující zákazníka

4.9.3 Přiřazení aktivity

Po diskusi s managementem společnosti Orchitech Solutions, s.r.o. byly vybrány základní a nejčastěji používané aktivity, které firma ve výkazech používá. Pro každou z nich je níže definováno jejich použití a přehled pro jejich zvolení v automatické kategorizaci. Výčet kategorií je následující:

- Analysis & Design
- Consultation
- Development
- Code review
- Management

Aktivitu lze s jistou mírou přesnosti určit podle používané aplikace, tuto informaci lze vyčíst z dat *watchers aw-watcher-window* a *aw-watcher-web*.

Analysis & Design

Aktivita vztahující se ke kategorii *Analysis & Design* zahrnuje činnosti spojené s analýzou a designem v souvislosti s daným projektem či požadavkem. Zahrnuje veškerou přípravu podkladů, PoC, vytváření a rozpad dílčích úkolů daného požadavku, respektive

dané objednávky (SOW). Dále sem spadají přípravy diagramů, jakožto další součást prezentace daného řešení znázorňujícího využívané procesy. Do této kategorie spadají aplikace jako je *draw.io* pro tvorbu diagramů. Současně jsou to textové a tabulkové editory, tedy Google Docs, Google Sheets nebo Google Slides a také aplikace z balíku LibreOffice.

Consultation

Kategorie *Consultation* představuje různorodé schůzky se zákazníkem týkající se nových a/nebo stávajících požadavků. V průběhu příprav požadavku a během samotného zpracování často vyvstávají nejrůznější nejen implementační otázky. Některé je možné rozhodnout interně, do jiných je nutné zapojit také další strany, aby požadavek dopadl k co největší spokojenosti zákazníka. Agilní přístup umožňuje průběžnou prioritizaci a dodefinování požadavků. Pro všechny tyto účely slouží kategorie Consultation, do které jsou řazeny například aplikace *Google Meet, Skype* či *Slack*.

Development

Pravděpodobně nejčastěji je očekáváno trackování vývoje, tedy kategorie *Development*. Samotný vývoj aplikací je nedílnou součástí technologické firmy vyvíjející systémy a aplikace na míru. Mimo samotného programování zahrnuje tato kategorie taktéž související aktivity. Filtr pro zařazení měřené aktivity do této kategorie zahrnuje vývojová prostředí, která v rámci své práce zaměstnanci nejčastěji používají. Jedná se o aplikace *Visual Studio Code, Eclipse, WebStorm* a *IntelliJ IDEA*.

Code review

Pro zajištění kvality dodávaného produktu je nedílnou součástí procesu implementace kontrola kódu tzv. *Code review*. V tomto případě je specifikace nástroje jednoznačná, firma pro tyto účely využívá nástroj *GitLab*.

Management

Nejrůznější projektové schůzky, jako standup, planning, refinement nebo retrospektiva, spadají do aktivity *Management*. Vedle těchto schůzek se do kategorie dále řadí práce s výkazy na projektu a další projektová operativa zahrnující nejčastěji fakturace a akceptace v rámci jednotlivých projektů. Tyto činnosti již bývají specifické pro projekt či pracovní pozici zaměstnance. Každý projekt má pro projektové schůzky zpravidla specifickou adresu, která je opětovně využívána. Na základě toho je možné blíže rozlišit různorodé schůzky se zákazníkem od projektových a pravidelných.

Současně tato kategorie nejvíce odpovídá potřebám a skutečnostem řešených v rámci interních schůzek a konzultací, pro které je vyčleněna *issue* #18911. Také zde platí, že někteří zaměstnanci tuto kategorii využívají častěji než ostatní.

4.10 Automatická kategorizace

Aby bylo možné automaticky zařadit naměřená data do kategorií, bylo nutné sestavit regulární výrazy pro jednotlivé kategorie. Výstupem jednotlivých podkapitol budou navržené kategorie v podobě odrážkového seznamu, jenž budou obsahovat název kategorie a sestavený regulární výraz ve formátu *Název kategorie - regulární výraz*. Následně tyto kategorie budou spojeny do jednoho celku, jenž bude transformován do požadovaného formátu nástroje ActivityWatch, viz ukázka 5. Výsledný soubor kategorií je součástí této práce jako Příloha B.

4.10.1 Číslo *issue* v Redmine

Na základě výstupů analýzy naměřených dat v kapitole 4.9.1 Číslo *issue* v Redmine byl pro každý uvedený *watcher* a *importer* sestaven regulární výraz, pomocí kterého bude z dat získáno číslo *issue* v Redmine.

aw-watcher-ttrebel a aw-importer-ttrebel

Díky sdílenému mapování *events* modulů *aw-watcher-ttrebel* a *aw-importer-ttrebel* bylo možné sestavit regulární výraz pokrývající oba zmíněné moduly.

```
#([0-9]{5}):
```

Kód 15: Sestavený regulární výraz pro *aw-watcher-ttrebel* a *aw-importer-ttrebel*

aw-watcher-vscode

Název větve ve verzovacím systému Git může začínat jako *rm39024*, *RM39024*, *39024*, *rm39024-test*. Pro účely *watcher aw-watcher-vscode* byl sestaven regulární výraz pokrývající všechny zmíněné formáty názvů větve.

```
^(?:rm|RM)?([0-9]{5})(?:(?:-|_|)[a-zA-Z]+)?$
```

Kód 16: Sestavený regulární výraz pro *aw-watcher-vscode*

aw-watcher-jetbrains

Regulární výraz, jenž byl sestaven pro tento *watcher*, je identický s regulárním výrazem zmíněným výše. Tato skutečnost je možná proto, že stejně tak jako u *watcher aw-watcher-vscode*, obsahuje *aw-watcher-jetbrains* na začátku názvu větve název *issue*, ke které se dané úpravy váží. Připravený regulární výraz pro tento případ lze vidět v ukázce níže.

```
^(?:rm|RM)?([0-9]{5})(?:(?:-|_|)[a-zA-Z]+)?$
```

Kód 17: Sestavený regulární výraz pro *aw-watcher-jetbrains*

aw-watcher-web

Watcher aw-watcher-web obsahuje data o dané úloze v URL a názvu karty. URL může nabývat hodnoty *https://www.redmine.orchi.tech/issues/38226*, název karty může nabývat hodnoty *Feature #33189: Nahrazeni knihovny momentjs - Redmine*. Pro tento *watcher* byl vzhledem k formátu hodnot v URL a názvu karty vydefinován regulární výraz uvedený v ukázce 18, jenž pokrývá oba případy.

```
#([0-9]{5}):
```

Kód 18: Sestavený regulární výraz pro *aw-watcher-web*

aw-watcher-window

Watcher aw-watcher-window obsahuje informaci o aktivním okně, je tedy dle typu a názvu aplikace možné odhadnout informaci o zpracovávané úloze. Naměřená data obsahují informaci o čísle *issue* ve formátu *Story #34009: Implementace provisioneru*. Pro účely tohoto byl sestaven regulární výraz uvedený v ukázce 19.

```
#([0-9]{5}):
```

Kód 19: Sestavený regulární výraz pro *aw-watcher-window*

Sestavené kategorie

Výše byly definovány regulární výrazy pro jednotlivé *importers* a *watchers*. V sestavených regulárních výrazech lze spatřit opakující se vlastnosti. Z tohoto důvodu bylo přistoupeno k vydefinování jednoho regulárního výrazu pokrývajícího všechny případy k získání čísla *issue*. Mezi výrazy byl použit znak `|`, jenž umožňuje kontrolu shody s první nebo druhou částí výrazu.

- Issue - `#([0-9]{5}):|^(?:rm|RM)?([0-9]{5})(?:(?:-|_|)[a-zA-Z]+)?$`

4.10.2 Zákazník

Informaci o zákazníkovi obsahují data *importers aw-importer-ical*, *aw-importer-ttrebel* a *watcher aw-watcher-ttrebel*. Tyto zdroje obsahují informaci o zákazníkovi na začátku dat v titulku dané *tracking session*. Na rozdíl od čísla *issue*, kde byla data z analýzy dat vstupem pro regulární výraz, tak u zákazníka byl seznam zákazníků sestaven z reálného seznamu zákazníků firmy. Pro tento seznam byly následně sestaveny regulární výrazy.

Sestavené kategorie

Pro zachování anonymity dat byla jména všech zákazníků zaměněna za univerzální název ve formátu Zákazník *číslo*. Sestavené kategorie jsou obsaženy v zápisu níže, ten odpovídá formátu *Jméno zákazníka - regulární výraz*.

- Zákazník
  - Zákazník 1 - `Zákazník 1 (?:IAM|Support)`
  - Zákazník 2 - `Zákazník 2 (?:Development|Security)`
  - Zákazník 3 - `Zákazník 3 (?:IdM|Support)`
  - Zákazník 4 - `Zákazník 4 (?:IdM|Support)`
  - Zákazník 5 - `Zákazník 5`
  - Zákazník 6 - `Zákazník 6`
  - Zákazník 7 - `Zákazník 7`

4.10.3 Přiřazení aktivity

Na základě analýzy dat v kapitole 4.9.3 Přiřazení aktivity, jenž poskytla přehled používaných aplikací ve společnosti Orchitech Solutions, s.r.o., je možné sestavit regulární výrazy pro přiřazení trackované události k dané aktivitě uživatele. Některé navržené kategorie byly po konzultaci s managementem společnosti doplněny o další aplikace či upřesňující podkategorie. Podobně jako jsou sestaveny výchozí kategorie nástroje ActivityWatch, které kategorizují činnost dle názvu aplikace, byly sestaveny regulární výrazy pro navržené kategorie.

Sestavené kategorie

Na základě analýzy byl sestaven následující přehled aplikací, přičemž každá aplikace byla zařazena do některé z používaných aktivit. Jelikož ani tento seznam nemusí být konečný, tak v případě, že daná aplikace nebude odpovídat žádné aktivitě, bude umístěna do aktivity *Uncategorized*. V souvislosti s anonymizací dat byly názvy schůzek upraveny.

- Analysis & Design
  - Dokumentace - `Google Drive|Google Docs|Google Sheets|Google Slides|libreoffice`
  - Diagramy - `draw\.io`
- Consultation
  - Přímá komunikace - `Google Meet|Microsoft Teams|Slack|Element|Skype`
  - E-mail - `Gmail`
- Development
  - Implementace - `Visual Studio Code|Eclipse|WebStorm|IntelliJ IDEA|Jira|Redmine|Stack Overflow|GitHub`
  - Dokumentace - `Wiki|KB|Knowledge Base`
  - Testování - `localhost|127\.0\.0\.1|dev`
- Code review - `GitLab`
- Management
  - Projektové schůzky - `nzq-spma-cnb|nxp-masv-bat`
  - Interní schůzky - `del-atuc-jup|18911`
- Uncategorized

4.11 ActivityWatch

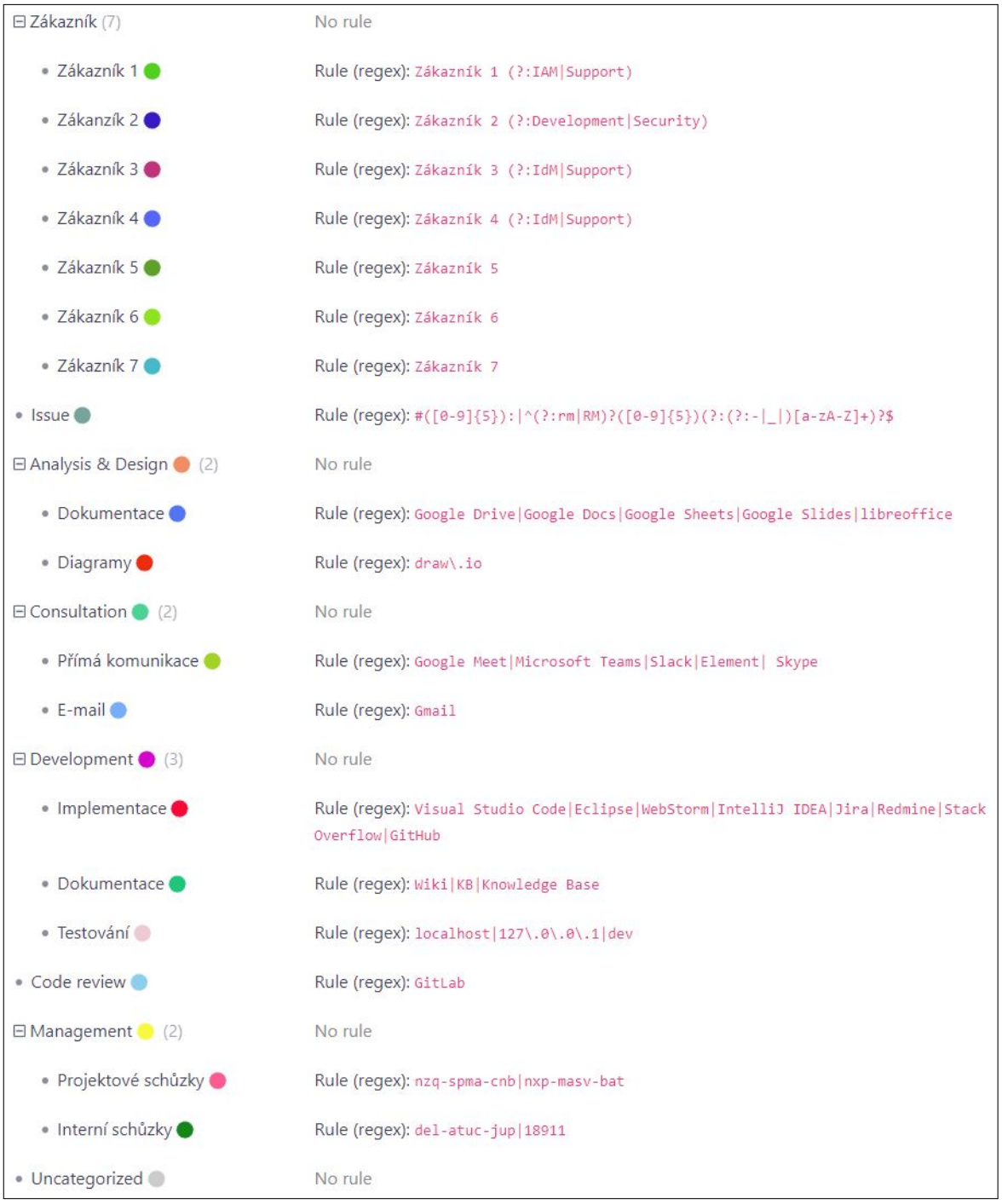
Aplikace ActivityWatch, jakožto jedna z hlavních komponent výsledného řešení, v průběhu implementace prošla několika více či méně uživatelsky patrnými změnami. Nejen, že probíhaly změny vedoucí k výslednému řešení určenému pro firmu Orchitech Solutions, s.r.o., ale taktéž byly publikovány další verze a změny přímo od tvůrců aplikace. Verze ActivityWatch, se kterou byly spjaty poslední změny v době zpracovávání této práce, je *v0.12.2b1*. Zde jsou obsaženy nové, případně již existující konfigurační možnosti. Mezi nové se řadí nástroj *Category builder* nebo možnost vytvoření regulárního výrazu pro lepší vymezení *AFK* stavu.

4.11.1 Konfigurace nástroje

Proto, aby byly výstupy z aplikace co nejvíce přínosné, je potřeba před používáním nastavit uživatelsky nejdůležitější části. Z již výše zmíněných informací se jedná o přidání či aktualizaci regulárních výrazů pro různé kategorie. Další nastavení se týká *Category builder* a dodatečné konfigurace rozlišení *AFK* stavu. Všechna tato nastavení i s komentářem jsou uvedena níže.

Sestavené kategorie

Finální nastavení v aplikaci ActivityWatch, a to rozlišení zákazníka, čísla *issue* a jednotlivé aktivity je zobrazeno níže na obrázku 14. V levém sloupci luze spatřit seznam a barevné rozlišení jednotlivých dat, která se následně vyskytují v aplikaci. V pravém sloupci jsou specifikovány jednotlivé regulární výrazy, jenž odpovídají výstupům získaným během analýzy dat.



Obrázek 14: Nastavená kategorizace - vlastní zpracování

Category builder

Tento nástroj pomáhá vytvářet kategorie z nekategorizovaného času uživatele. Funguje tak, že načte veškerý nekategorizovaný čas za zvolené období a poté najde nejběžnější slova podle celkového času, nikoliv podle počtu výskytů. Každé z nalezených slov může být buď ignorováno, je-li příliš obecné či irelevantní, nebo může být použito k vytvoření nové kategorie nebo k připojení slova k již existujícímu pravidlu.

Na obrázku 15 je ukázka několika pojmů, jenž jsou řazeny do kategorie *Uncategorized*. Je zde vidět, že uživatel používá aplikaci Microsoft Edge, tedy prohlížeč jenž není součástí žádného regulárního výrazu na obrázku 14. Současně pro tento prohlížeč není samostatný *watcher* tak, jak tomu je u prohlížeče Google Chrome. Na základě tohoto výstupu se může uživatel rozhodnout, zda aplikaci zahrne do některé již existující kategorie nebo tento výstup i nadále bude patřit mezi nekategorizované aplikace. Dále se může rozhodnout pro aplikaci vytvořit vlastní pravidlo a kategorii, přidat aplikaci do již existujícího pravidla.

Obrázek 15: Nastavená kategorizace - vlastní zpracování

Dodatečná konfigurace rozlišení *AFK* stavu

ActivityWatch poskytuje dodatečnou konfiguraci pro rozlišení *AFK* stavu pomocí regulárního výrazu. Tato konfigurační volba umožňuje uživateli definovat aplikace, při kterých *watcher aw-watcher-afk* nemá časový úsek odpovídající časovému úseku dané aplikace označovat stavem *AFK*. Typickým případem použití jsou komunikační aplikace jako Google Meet či Microsoft Teams. Uživatel při používání zmíněných aplikací nemusí aktivně ovládat zařízení pomocí myši či klávesnice, díky kterým by *watcher aw-watcher-afk* rozpoznal přítomnost, nicméně lze předpokládat, že je u zařízení přítomen a věnuje se pracovní aktivitě. Ačkoliv uvedený případ použití je v této práci pokrytý pomocí *watcher aw-watcher-table*, jedná se o další upřesnění rozpoznání přítomnosti uživatele. Pro účely této práce byl definován regulární výraz pro aplikace Google Meet, Microsoft Teams a Skype uvedený v ukázce 20.

```
Google Meet|Microsoft Teams|Skype
```

Kód 20: Regulární výraz pro dodatečné rozlišení *AFK* stavu

4.11.2 Rekonstrukce pracovního dne

Pracovní den je složen z několika po sobě jdoucích úloh nebo aktivit. Tyto aktivity lze zahrnout a zobrazit prostřednictvím časové osy. Časová osa obsahuje různé časové úseky, jenž mohou být více či méně detailní. Také je možné zobrazit kategorizovaná data z vybraného dne pomocí výsečového grafu.

Časová osa

Jedním z hlavních výstupů práce je časová osa, která znázorňuje komplexní získané informace z *watchers* a *importers*. Zaměstnanec má možnost zobrazit si časovou osu pro stanovený čas zpětně od aktuální chvíle nebo pro jednotlivé dny, viz obrázek 16. Následně si může v časové ose ještě více přiblížit úsek, který jej zajímá a který potřebuje vykázat. Detailnější záznam událostí je zobrazen na obrázku 17. Uživatelsky jednoduchým a efektivním způsobem tedy může nalézt prázdný časový úsek i s informacemi, které mu poskytují podklad pro vykázání času.

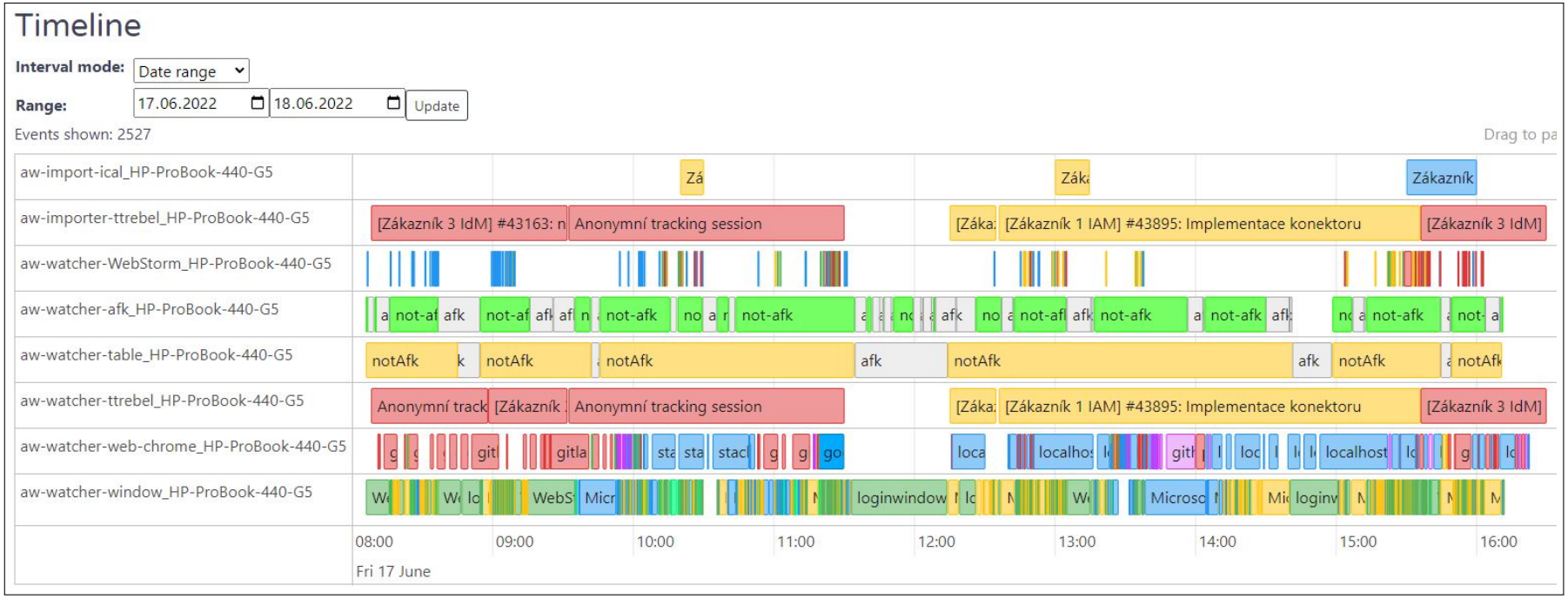
Importer z Google Kalendáře je zobrazen na prvním řádku časové osy a je jej možné sledovat na obrázku 16 a 17.

Druhý znázorněný *importer*, kterým je *aw-importer-ttrebel* souvisí s *watcher* aplikace TTRebel. Na řádku s *importer* je první časový blok již pojmenován jako *Zákazník 3 IdM #43163*, přičemž u *watcher* ve stejném čase obsahuje *Anonymní tracking session*. Tato nekonzistence je způsobena tím, že uživatel nejprve vytvořil *Anonymní tracking session* a poté kolem deváté hodiny k této *tracking session* přidal číslo *issue*. *Importer* vykresluje jeden měřený úsek, zatímco pro *watcher*, který průběžně zaznamenává informaci o *issue*, vznikly časové úseky dva. Navazující časové úseky obsahující *Anonymní tracking session* v obou těchto zdrojích jsou detailněji znázorněny na obrázku 17. Právě taková situace, nebo velice podobná, je jednou z očekávaných vstupů uživatele pro zpětné dohledávání stráveného času. V souvislosti s *watcher* prohlížeče Google Chrome lze odvodit, že se uživatel věnoval *Code Review*, *Development* nebo *Consultation*, a to z toho důvodu, že značnou část času měl jako poslední otevřenou záložku webové stránky gitlab.com a stackblitz.com. Jelikož se ve *watcher* nachází informace, že současně měl spuštěnou aplikaci Microsoft Teams, lze očekávat, že se jednalo o *Consultation* a uživatel současně komunikoval se zákazníkem ohledně určité specifikace. Po najetí kurzorem myši na detail časového úseku jsou zobrazeny další informace, které mohou obsahovat mimo jiné i číslo *issue*, což lze typicky sledovat u *watchers* aplikací JetBrains nebo Visual Studio Code. Pokud by nebyla vytvořena jedna souvislá *tracking session*, ale bylo by jich více či by zcela chyběla, bylo by nutné, aby se uživatel ještě blíže zaměřil na informace ve *watchers*.

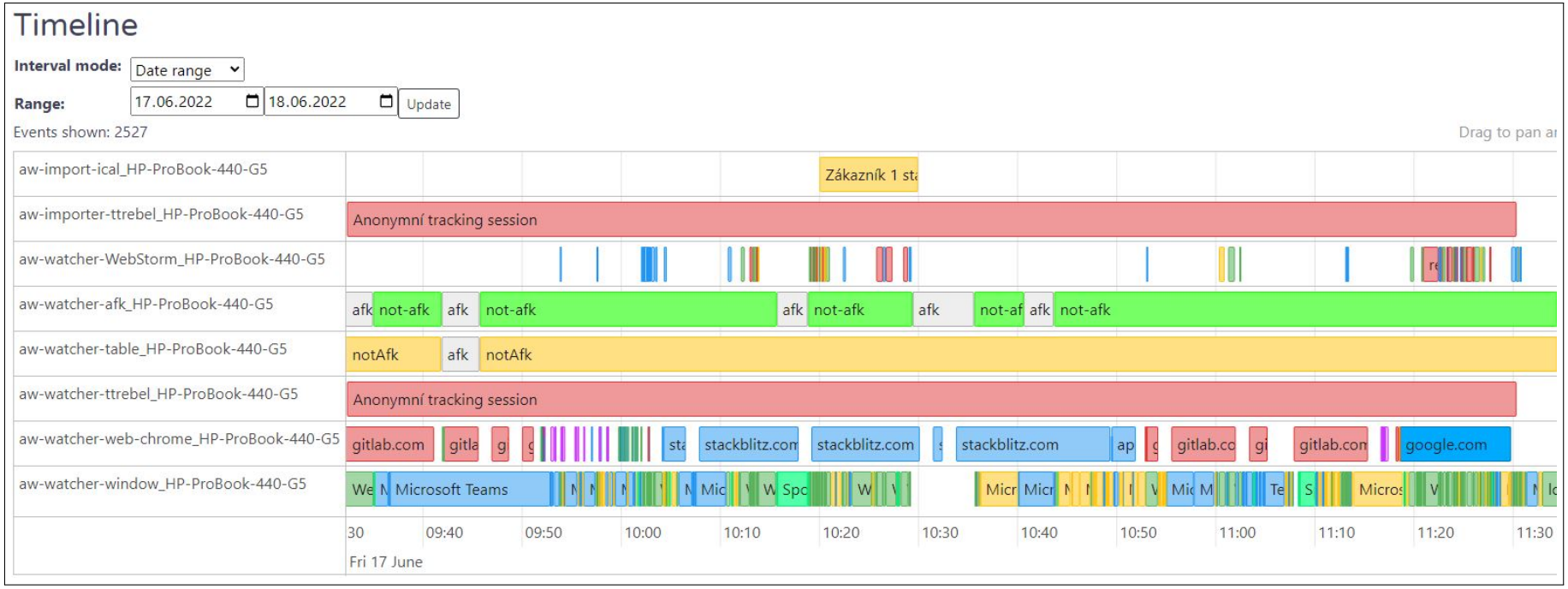
Na časové ose jsou zřetelné dva zdroje pro informaci o přítomnosti uživatele, přičemž se tyto zdroje v některých časových úsecích rozcházejí. To je způsobeno tím, že *watcher aw-watcher-table* sleduje přítomnost uživatele na základě zapojeného senzoru, zatímco výstupní informace z *aw-watcher-afk* je založena pouze na aktivitě klávesnice a myši. První jmenovaný *watcher* je tedy přesnějším zdrojem této informace. Pokud si zaměstnanec například čte dokumentaci a nehýbe myší či nepíše na klávesnici, tak pro druhý jmenovaný zdroj je nepřítomen u počítače a je zaznamenán stav *AFK*.

Časová osa dále obsahuje informaci o aktuální, respektive poslední otevřené záložce v aplikaci Google Chrome. Velice často si zaměstnanec při implementaci *issue* nejprve otevře danou *issue* v Redmine a následně ji zpracovává. Průběžně si také dohledává informace nebo konkrétnější požadavky pro danou *issue*.

Nejvíce událostí, respektive nejkratší časové úseky, obsahuje poslední řádek na obrázku 16, tedy *watcher* aktivního okna. Ten může dokreslit další souvislosti s aktivitou zaměstnance. Jakmile se zaměstnanec přepne z jedné aplikace do jiné, je v tomto řádku vytvořen nový úsek.



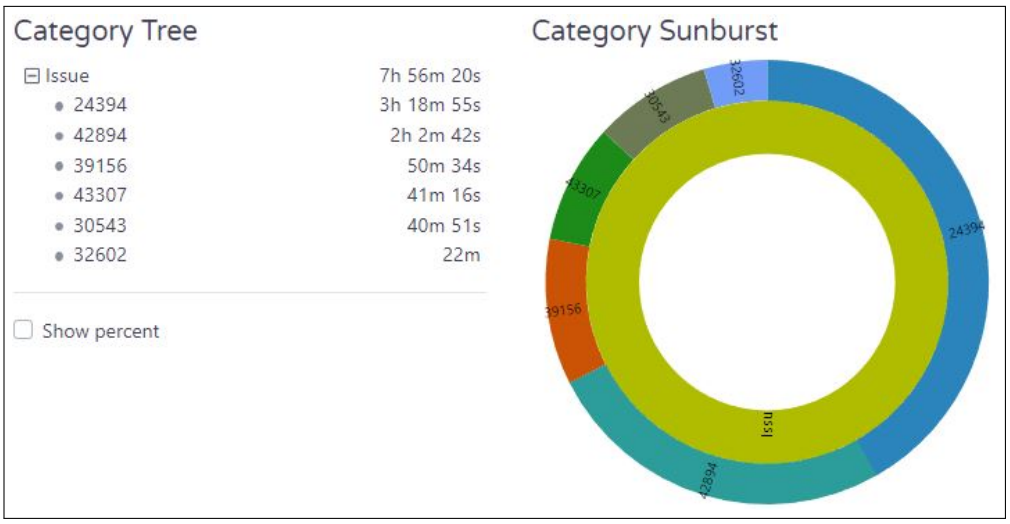
Obrázek 16: Pracovní den znázorněný na časové ose - vlastní zpracování



Obrázek 17: Detail časového úseku anonymní *tracking session* - vlastní zpracování

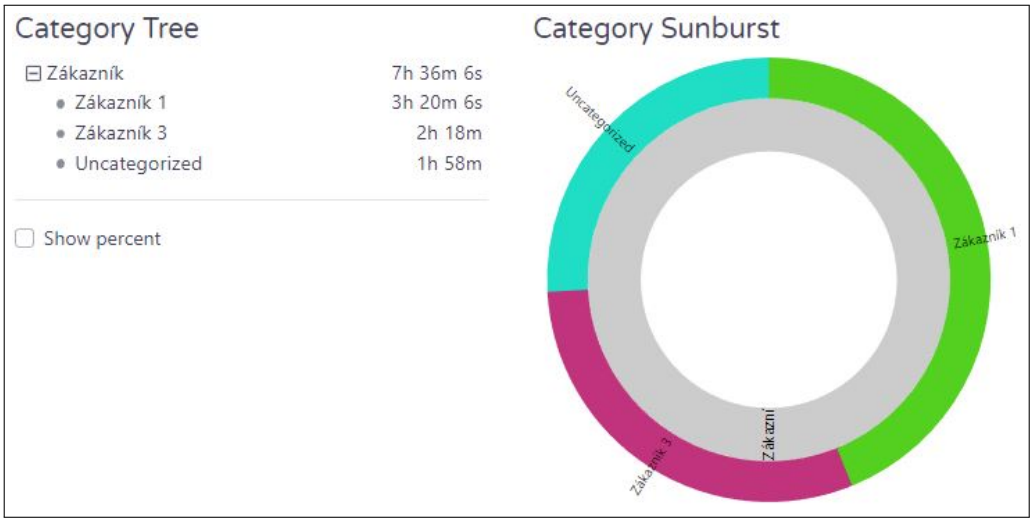
Kategorizovaná data

Uživatel má možnost zobrazit si z daného dne časové úseky, jenž byly věnovány jednotlivým *issues*. Tyto úseky znázorňují celkový čas, během kterého byla spuštěna aplikace ActivityWatch. Zatímco v levé části jsou pro číslo dané *issue* vypsány přesné hodnoty, tak na výsečovém grafu vpravo jsou jednotlivé úseky znázorněny graficky. Pokud by uživatel potřeboval, může hodnoty časových úseků zobrazit v procentech.



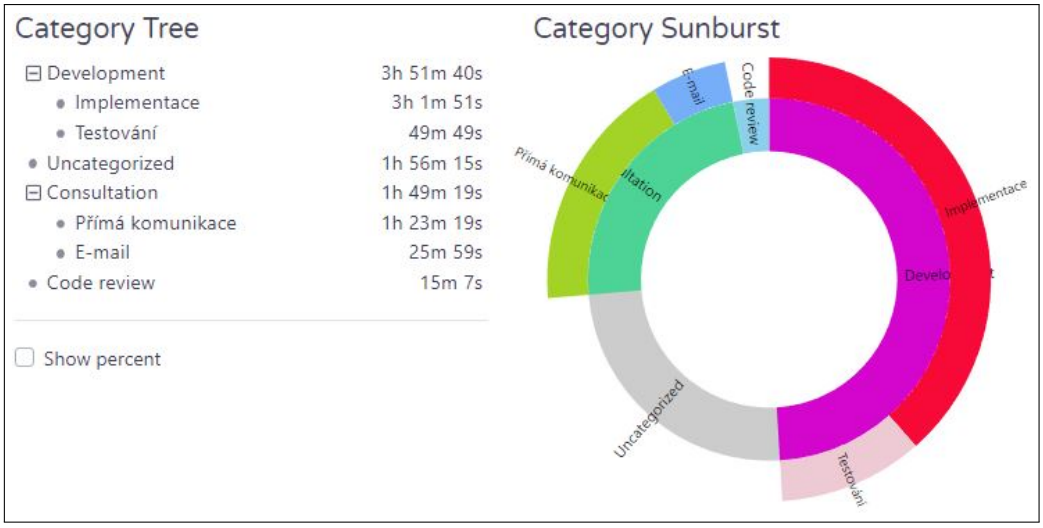
Obrázek 18: Kategorizovaná data podle čísla *issue* - vlastní zpracování

Na obrázku 19 lze pozorovat data kategorizovaná podle zákazníka. Téměř polovinu pracovního dne se zaměstnanec věnoval práci pro *Zákazníka 1*. Další čas, a to právě 2 hodiny a 18 minut, byl stráven pro *Zákazníka 3*. Zbývající část je označena jako *Uncategorized*, protože zaměstnanec má v tomto časovém úseku vytvořenou *Anonymní tracking session*.



Obrázek 19: Kategorizovaná data podle zákazníka - vlastní zpracování

Poslední výsečový graf znázorňuje různé aktivity v rámci jednoho sledovaného dne. Na zobrazeném grafu mohou být aktivity hůře čitelné, ale při znalosti nastavených kategorií se jedná spíše o kosmetický nedostatek. V grafu je například zaznamenaná aktivita *Consultation*, ta je dále dělena na *Přímou komunikaci* a *E-mail*, a to na základě připravených regulárních výrazů. Také na obrázku 20 je možné zobrazit informace z procentuálního rozložení naměřeného času.



Obrázek 20: Kategorizovaná data podle aktivity - vlastní zpracování

5 Zhodnocení

Stanovený cíl práce, kterým byl návrh a následné ověření řešení pro automatické sledování času aktivit pracovníků v oblasti profesionálních služeb, se podařilo naplnit. V průběhu implementace a naplňování tohoto cíle taktéž proběhl sběr dat a původní návrh proběhl finálními úpravami umožňujícími získaná data snadněji přiradit na zákazníka, respektive projekt a projektovou úlohu.

Taktéž byly naplněny specifické požadavky firmy, které byly sepsány v kapitole 4.5 Požadavky na navrhované řešení. Výsledné řešení v současné podobě automaticky monitoruje aktivitu uživatele, na základě čehož je možné vytvořit rekonstrukci dne. Současně poskytuje synchronizaci mezi aplikacemi TTRebel a ActivityWatch. Dále generuje podklady napomáhající uživateli přiřadit aktivitu a relevantní časový úsek ke konkrétní Redmine úloze. Řešení taktéž na základě připravených regulárních výrazů umožňuje kategorizaci aktivity do definovaných kategorií. Díky senzoru poskytuje informaci o přítomnosti uživatele u zařízení.

Nad rámec definovaných požadavků aplikace funguje nejen na operačním systému Windows a Linux, ale také je možné řešení provozovat na zařízeních s macOS. Realizované implementační úpravy týkající se získávání názvu větve z IDE Visual Studio Code byly poskytnuty open source komunitě.

5.1 Limity

V průběhu zpracování vyvstalo několik limitů týkající se výsledného řešení. Tyto limity pro funkčnost a splnění zadání nejsou zásadní, naopak slouží jako podklad pro možné vylepšení v případě, že o to bude mít firma zájem.

Univerzálnost řešení

Předkládané řešení pokrývá nejčastější úkony a aktivity pro pozici vývojáře. Pokud by firma měla potřebu mít ActivityWatch přizpůsoben pro různé pracovní pozice, bylo by nutné tyto požadavky dále zpracovat. Příkladem může být vývojář, který bude používat Google Sheets a jeho aktivita bude zařazena v kategorii *Analysis & Design*, zatímco projektový manažer tu samou aplikaci použije a s největší pravděpodobností očekává, že čas bude zaznamenán v kategorii *Management*.

Jelikož ale zaměstnanci firmy velice často působí na více pozicích současně, jak již bylo zmíněno v úvodu praktické části, v kapitole 4.1.1 Charakteristika firmy, bylo by pravděpodobně nutné připravit unikátní sadu kategorizačních filtrů pro jednotlivé zaměstnance.

Nedefinované aplikace

Je pravděpodobné, že některé aplikace, které zaměstnanci využívají méně často, nejsou v současném filtrování pokryty. Může proto nastat situace, kdy bude daná aktivita umístěna do nesprávné kategorie. Tento limit by bylo možné eliminovat současně s přípravou sady kategorizačních filtrů pro jednotlivé zaměstnance, což bylo zmíněno výše.

Zobrazení kategorií aktivit na časové ose

Kategorie aktivit, které jsou na základě regulárních výrazů přiřazeny, se nezobrazují na detailu časové osy. Uživatel ale má možnost je zobrazit ve výsečovém grafu.

6 Závěr

Odvedenou práci je možné zaznamenávat různými způsoby, od počtu vytvořených součástek v továrnách, přes počet vyřízených hovorů či e-mailů až například po počet řádků kódu. Právě ale v oblasti softwaru nemusí být kvantita důkazem kvality. Vývoj jednoho požadavku může obnášet a obvykle také obnáší celou řadu úkonů, které nezahrnují pouze samotnou implementaci. Práce softwarového vývojáře se tak tříští na mnoho činností a přesný záznam každé z nich si nemusí zaměstnanec vždy poctivě zaznamenat. Vytvoření návyku je klíčové, ačkoliv ne vždy bývá všespásné. Jedna chyba, vyrušení kolegou a do záznamu činností se dostává prázdné místo. Potřeba zpětného dohledávání informací o tom, na čem se daný den pracovalo, tak může způsobit nemalé problémy. Přesně toto byl důvod vzniku tématu této diplomové práce, která by měla zaměstnancům firmy poskytnout podklady k dohledání chybějících výkazů.

Diplomová práce je věnována návrhu, implementaci a ověření řešení pro automatické sledování času. Práce je rozdělena na dvě hlavní kapitoly, a to teoretická východiska a na vlastní práci. Pro potřeby této práce byla vytvořena rešerše zaměřující se na sledování a vykazování času, smysluplnosti a přínosech v této činnosti. Ve spojitosti s časovými řadami byl definován pojem časová osa, jelikož tato forma grafického znázornění je relevantní v souvislosti se sledováním času. Do časové osy je možné zanést jednotlivé činnosti z daného dne.

Přiblíženy byly projektové systémy, podrobnější informace byly představeny pro systém JIRA a Redmine, jakožto systémy využívané ve firmě. Jelikož existuje více přístupů ke sledování času, například metodou tužka - papír, tabulky, přepínacích kostek či software, byly také tyto krátce představeny. Dostupné nástroje v současné době poskytují možnost manuálního či automatického měření času. Vybraní zástupci byli stručně specifikováni, zejména s přihlédnutím na představená očekávání managementu firmy.

Vybrané nástroje, mezi které se řadí ActivityWatch a TTRebel, jsou v práci podrobněji představeny a blíže specifikovány včetně představení architektury a ukázky dané aplikace.

V úvodu vlastní práce byla představena firma, pro kterou je výstup této práce primárně určen. Současně byly představeny využívané nástroje ve firmě a také způsoby, kterými firma výkazy využívá. Získané teoretické znalosti následně byly využity pro vydefinování kritérií výběru senzoru a nástrojů, se kterými se dále pracuje. Výběr nástrojů je založen na již existujícím řešení využívaném ve firmě, kterým je TTRebel a druhým zvoleným nástrojem je ActivityWatch, jenž nejvíce odpovídá zadaným kritériím.

Dále byly sepsány detailnější a specifické požadavky firmy na navrhované řešení. Na základě získaných znalostí a požadavků byl připraven návrh řešení pro vybrané nástroje a komponenty. Tento návrh byl poté uveden díky implementačním změnám do provozu. Takto získaná data byla analyzována, díky čemuž je výsledné automatické sledování času vylepšeno tak, aby odpovídalo potřebám pracovníků firmy.

V souvislosti s rekonstrukcí pracovního dne umožňuje navržené řešení uživateli zobrazit přehledně, nejčastěji pak v časové ose, podrobnější informace o tomto dni. Toto řešení znázorňuje, jakým způsobem jsou jednotlivé aktivity rozloženy v průběhu pracovního dne. Zejména u zaměstnanců, kteří si vykazují s delší časovou prodlevou tento výstup zkracuje dobu hledání a zjednodušuje zpětné dohledávání nevykázaného času.

7 Seznam použitých zdrojů

ATLASSIAN, 2022a. *Jira Software - Integrations | Atlassian* [online] [cit. 2022-10-04]. Dostupné z: https://www.atlassian.com/software/jira/features/integrations.

ATLASSIAN, 2022b. *What is Jira Software used for? | Atlassian* [online] [cit. 2022-10-04]. Dostupné z: https://www.atlassian.com/software/jira/guides/use-cases/what-is-jira-used-for#jira-for-agile-teams.

BARTOŠ, Michal, 2021. *Send Git branch name. #26* [online] [cit. 2022-04-02]. Dostupné z: https://github.com/ActivityWatch/aw-watcher-vscode/pull/26.

BETTEREVALUATION, 2023. *Timelines and time-ordered matrices* [online] [cit. 2023-02-06]. Dostupné z: https://www.betterevaluation.org/methods-approaches/methods/timelines-time-ordered-matrices.

BJÄREHOLT, Erik; BJÄREHOLT, Johan, 2021a. *Architecture - ActivityWatch v0.10.0 documentation - Server* [online] [cit. 2021-07-26]. Dostupné z: https://docs.activitywatch.net/en/latest/architecture.html#server.

BJÄREHOLT, Erik; BJÄREHOLT, Johan, 2021b. *Importers - ActivityWatch v0.10.0 documentation* [online] [cit. 2021-07-26]. Dostupné z: https://docs.activitywatch.net/en/latest/importers.html.

BJÄREHOLT, Erik; BJÄREHOLT, Johan, 2021c. *Importers - ActivityWatch v0.10.0 documentation* [online] [cit. 2021-07-26]. Dostupné z: https://docs.activitywatch.net/en/latest/importers.html.

BJÄREHOLT, Erik; BJÄREHOLT, Johan, 2021d. *Watchers - ActivityWatch v0.10.0 documentation* [online] [cit. 2021-07-26]. Dostupné z: https://docs.activitywatch.net/en/latest/watchers.html.

BJÄREHOLT, Erik; BJÄREHOLT, Johan, 2022. *ActivityWatch - Open-source time tracker* [online] [cit. 2022-12-11]. Dostupné z: https://activitywatch.net/.

BRENA, Ramon F.; GARCÍA-VÁSQUEZ, Juan Pablo; GALVÁN-TEJADA, Carlos E.; MUÑOZ-RODRIGUEZ, David; VARGAS-ROSALES, Cesar; FANGMEYER, James, 2017. Evolution of Indoor Positioning Technologies: A Survey. *Journal of Sensors*, s. 1–21. ISSN 1687-725X. Dostupné z DOI: doi:10.1155/2017/2630413.

BURDA, Zdeněk, 2010. *Statistika pro obchodní akademie*. 5. vyd. Praha: Fortuna. ISBN 978-80-244-3841-2.

BWC, Admin, 2019. *Software Development Life Cycle (SDLC)* [online] [cit. 2022-10-04]. Dostupné z: https://bigwater.consulting/2019/04/08/software-development-life-cycle-sdlc/.

CARTER, Benjamin T.; LUKE, Steven G., 2020. Best practices in eye tracking research. *International Journal of Psychophysiology*, s. 49–62. ISSN 01678760. Dostupné z DOI: doi:10.1016/j.ijpsycho.2020.05.010.

FRESHBOOKS, 2019. *What Are Billable Hours? Time Tracking Tips To Get You Paid* [online] [cit. 2022-10-15]. Dostupné z: https://timelyapp.com/guides/time-tracking.

HAYES, Adam, 2022. *What Is a Time Series and How Is It Used to Analyze Data?* [Online] [cit. 2023-01-11]. Dostupné z: https://www.investopedia.com/terms/t/timeseries.asp.

HE, Zhenwen; MA, Xiaogang, 2018. A Distributed Indexing Method for Timeline Similarity Query. *Algorithms*. Roč. 11, č. 4, s. 19. ISSN 1999-4893. Dostupné z DOI: doi:10.3390/a11040041.

HOLTMANNSPÖTTER, Dirk; REUSCHER, Günter, 2009. *Optical technologies. Technology Guide*. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-540-88545-0. Dostupné z DOI: doi:10.1007/978-3-540-88546-7_19.

IBM, 2023. *Time series chart overview* [online] [cit. 2023-02-06]. Dostupné z: https://www.ibm.com/docs/en/qradar-on-cloud?topic=management-time-series-chart-overview.

INFLUXDATA, 2023. *What is Time Series Data? | Definition, Examples, Types & Uses.* [Online] [cit. 2023-01-11]. Dostupné z: https://www.influxdata.com/what-is-time-series-data/.

KLADIVO, Petr, 2013. *Základy statistiky*. Olomouc: Univerzita Palackého v Olomouci. ISBN 978-80-244-3841-2.

KLEINER, Mendel, 2011. *Acoustics and Audio Technology*. 3. vyd. USA: J. Ross Publishing. ISBN 978-1-60427-052-5.

LANG, Jean-Philippe; ROHLOFF, Bernhard, 2022. *Overview - Redmine* [online] [cit. 2022-10-05]. Dostupné z: https://www.redmine.org/projects/redmine/wiki.

REDMINE, 2022. *Plugins - Redmine* [online] [cit. 2022-10-05]. Dostupné z: https://www.redmine.org/plugins.

REPLOGLE, Nicole, 2023. *The 5 best time tracking apps in 2023* [online] [cit. 2023-02-22]. Dostupné z: https://zapier.com/blog/best-time-tracking-apps/.

RESCUETIME, 2021a. *Integrations - RescueTime Classic* [online] [cit. 2021-07-29]. Dostupné z: https://help.rescuetime.com/category/209-integrations.

RESCUETIME, 2021b. *RescueTime - Connect your RescueTime account to Zapier* [online] [cit. 2021-07-29]. Dostupné z: https://www.rescuetime.com/integrations/zapier.

RESCUETIME, 2021c. *RescueTime - Fully Automated Time Tracking Software* [online] [cit. 2021-07-29]. Dostupné z: https://www.rescuetime.com/.

RUDDER, Alana; MAIN, Kelly, 2022. *Best Project Management Software Of 2022* [online] [cit. 2022-10-03]. Dostupné z: https://www.forbes.com/advisor/business/software/best-project-management-software/.

SHAWN, 2020. *Types of Distance Sensors and How to Select One?* [Online] [cit. 2022-09-06]. Dostupné z: https://www.seeedstudio.com/blog/2019/12/23/distance-sensors-types-and-selection-guide/.

SHUMWAY, Robert H.; STOFFER, David S., 2017. *Time series analysis and its applications: with R examples*. 4. vyd. Cham: Springer International Publishing. ISBN 9783319524511.

TIMEFLIP, 2022a. *Connect TIMEFLIP2 with popular calendar apps* [online] [cit. 2022-09-19]. Dostupné z: https://timeflip.io/api.

TIMEFLIP, 2022b. *TIMEFLIP2: Quick Start Guide* [online] [cit. 2022-09-19]. Dostupné z: https://timeflip.io/quickstartguide.

TIMELY, 2017. *Is it ethical to track employees?* [Online] [cit. 2022-10-15]. Dostupné z: https://timelyapp.com/blog/is-it-ethical-to-track-employees.

TIMELY, 2018. *Why everyone should track non-billable hours* [online] [cit. 2022-10-11]. Dostupné z: https://timelyapp.com/blog/why-you-should-track-non-billable-hours.

TIMELY, 2020. *What is automatic time tracking?* [Online] [cit. 2022-09-20]. Dostupné z: https://timelyapp.com/blog/what-is-automatic-time-tracking.

TIMELY, 2022a. *Do you really know what your team is working on?* [Online] [cit. 2022-10-11]. Dostupné z: https://timelyapp.com/blog/what-your-team-is-working-on.

TIMELY, 2022b. *Itegrations - Timely* [online] [cit. 2022-09-20]. Dostupné z: https://timelyapp.com/integrations.

TIMELY, 2022c. *Our privacy promise* [online] [cit. 2022-09-20]. Dostupné z: https://timelyapp.com/our-privacy-promise.

TIMELY, 2022d. *Time tracking: The Ultimate Time Tracker Guide* [online] [cit. 2022-10-11]. Dostupné z: https://timelyapp.com/guides/time-tracking.

TIMELY, 2022e. *Timely Pricing & Plans* [online] [cit. 2022-09-20]. Dostupné z: https://timelyapp.com/pricing.

TIMEULAR, 2022a. *Time Tracking Cube: The 8-Sided Time Tracker Dice - Timeular* [online] [cit. 2022-09-10]. Dostupné z: https://timeular.com/tracker/.

TIMEULAR, 2022b. *Timeular keyboard shortcuts - accurate time tracking software with no effort* [online] [cit. 2022-09-10]. Dostupné z: https://timeular.com/quicktrack/.

TIMEULAR, 2022c. *Timeular Pricing* [online] [cit. 2022-09-10]. Dostupné z: https://timeular.com/pricing/.

TOGGL, 2022a. *Features | Toggl Track* [online] [cit. 2022-09-24]. Dostupné z: https://toggl.com/track/features/.

TOGGL, 2022b. *Redmine Time Tracking | Toggl Track* [online] [cit. 2022-09-24]. Dostupné z: https://toggl.com/track/redmine-time-tracking/.

TOGGL, 2022c. *Time Tracking Integrations with 100+ Tools | Toggl Track* [online] [cit. 2022-09-24]. Dostupné z: https://toggl.com/track/integrations/.

WAKATIME, 2022. *WakaTime API Docs* [online] [cit. 2022-10-21]. Dostupné z: https://wakatime.com/developers.

WARCHOLINSKI, Matt, 2022. *Top 6 Best Issue Tracking Systems in 2022* [online] [cit. 2022-10-03]. Dostupné z: https://brainhub.eu/library/best-issue-tracking-systems.

ZAPIER, 2023. *How it works* [online] [cit. 2023-02-22]. Dostupné z: https://zapier.com/how-it-works.

8 Seznam obrázků, tabulek, grafů a zkratek

8.1 Seznam obrázků

8.2 Seznam tabulek

8.3 Seznam kódů

8.4 Seznam použitých zkratek

AFK Away from keyboard

API Application Programming Interface

IDE Integrated Development Environment

JSON JavaScript Object Notation

REST Representational State Transfer

SaaS Software as a Service

8.5 Slovník pojmů

bucket Samostatný jmenný prostor, jehož obsahem je množina *events* stejného typu.

event Základní datová entita aplikace ActivityWatch reprezentující měřenou událost.

heartbeat Metoda aplikace ActivityWatch, která spojuje sousední *events* s identickými daty v rámci tzv. pulzního okna.

importer Obecné pojmenování modulu aplikace ActivityWatch sloužící k importu dat z jiné aplikace.

issue Identifikovaný problém, který se vyskytuje během projektu a vyžaduje řešení.

tracking interval Datová entita aplikace TTRebel, jenž reprezentuje měřené časové období.

tracking session Základní datová entita aplikace TTRebel, jenž reprezentuje jednu *issue* v Redmine, v rámci které si uživatelé měří čas strávený řešením *issue*.

watcher Obecné pojmenování modulu aplikace ActivityWatch sloužící k periodickému sledování aktivity uživatele v reálném čase.

Přílohy

Příloha A - Obsah přiloženého média

- `ActivityWatch` ..... zdrojové kódy implementovaných modulů
  - `aw-importer-ttrebel`
  - `aw-ttrebel-core`
  - `aw-watcher-ttrebel`
  - `kategorie.json`
- `STL` ..... tiskové soubory ve formátu STL
  - `table_watcher_base.stl`
  - `table_watcher_body.stl`
  - `table_watcher_cover.stl`
- `zaverecna_prace.pdf` ..... text práce ve formátu PDF

Příloha B - Přehled sestavených kategorií

- Issue - `#([0-9]{5}):|^(?:rm|RM)?([0-9]{5})(?:(?:-|_|)[a-zA-Z]+)?$`
- Zákazník
  - Zákazník 1 - `Zákazník 1 (?:IAM|Support)`
  - Zákazník 2 - `Zákazník 2 (?:Development|Security)`
  - Zákazník 3 - `Zákazník 3 (?:IdM|Support)`
  - Zákazník 4 - `Zákazník 4 (?:IdM|Support)`
  - Zákazník 5 - `Zákazník 5`
  - Zákazník 6 - `Zákazník 6`
  - Zákazník 7 - `Zákazník 7`
- Analysis & Design
  - Dokumentace - `Google Drive|Google Docs|Google Sheets|Google Slides|libreoffice`
  - Diagramy - `draw\.io`
- Consultation
  - Přímá komunikace - `Google Meet|Microsoft Teams|Slack|Element|Skype`
  - E-mail - `Gmail`
- Development
  - Implementace - `Visual Studio Code|Eclipse|WebStorm|IntelliJ IDEA|Jira|Redmine|Stack Overflow|GitHub`
  - Dokumentace - `Wiki|KB|Knowledge Base`
  - Testování - `localhost|127\.0\.0\.1|dev`
- Code review - `GitLab`
- Management
  - Projektové schůzky - `nzq-spma-cnb|nxp-masv-bat`
  - Interní schůzky - `del-atuc-jup|18911`
- Uncategorized