TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Webová aplikace pro administraci projektů

Diplomová práce

Studijní program: N2612 – Elektrotechnika a informatika
Studijní obor: 1802T007 – Informační technologie

Autor práce: **Bc. Petr Žďárský**
Vedoucí práce: Ing. Igor Kopetschke

Liberec 2018

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics
and Interdisciplinary Studies

Web application for project administration

Master thesis

Study programme: N2612 – Electrical Engineering and Informatics
Study branch: 1802T007 – Information Technology

Author: **Bc. Petr Žďárský**
Supervisor: Ing. Igor Kopetschke

Liberec 2018

Prohlášení

Byl jsem seznámen s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu TUL.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Diplomovou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé diplomové práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum: 6.9.2018

Podpis:

Technická univerzita v Liberci
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: **2017/2018**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Petr Žďárský**
Osobní číslo:	**M15000193**
Studijní program:	**N2612 Elektrotechnika a informatika**
Studijní obor:	**Informační technologie**
Název tématu:	**Webová aplikace pro administraci projektů**
Zadávající katedra:	**Ústav nových technologií a aplikované informatiky**

Zásady pro vypracování:

1. Seznamte se s potřebami softwarových firem ohledně technického řešení administrace projektů, které používají, a proveďte průzkum trhu se zaměřením na existující řešení a cenovou politiku.
2. Navrhněte SW řešení, které bude splňovat požadavky vyplývající z průzkumu.
3. Implementuje navržené řešení v podobě webové aplikace na platformě .NET
4. Uveďte vyhodnocení a zpětnou vazbu získanou na základě zkušebního provozu u klientské firmy

Rozsah grafických prací: **dle potřeby**

Rozsah pracovní zprávy: **40 - 60 stran**

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

[1] RESIG, John. JavaScript a Ajax: moderní programování webových aplikací. Brno: Computer Press, 2007. ISBN 9788025118245.
[2] VIRIUS, Miroslav. Programování pro .NET. V Praze: České vysoké učení technické, 2011. ISBN 978-80-01-4-864-1.

Vedoucí diplomové práce: **Ing. Igor Kopetschke**
Ústav nových technologií a aplikované informatiky

Datum zadání diplomové práce: **19. října 2017**

Termín odevzdání diplomové práce: **14. května 2018**

L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

Ing. Josef Novák, Ph.D.
vedoucí ústavu

V Liberci dne 19. října 2017

Poděkování

Na tomto místě bych rád poděkoval Ing. Igoru Kopetschkemu za jeho odborné vedení mé diplomové práce. A také bych chtěl poděkovat panu Paulu Tesarovi BSc. zastupujícímu firmu Prague Labs s.r.o. za poskytování konzultací, databázového frameworku a HTML šablony aplikace.

Abstrakt

Tato diplomová práce se věnuje vývoji systému, který má usnadnit administrativní agendu malých firem a agentur působících v IT odvětví a oborech jim přidruženým. První část práce se zabývá průzkumem toho, co za aplikace firmy používají a jak je hodnotí. Tato část také obsahuje srovnání nejčastěji zmiňovaných aplikací a jejich analýzu. Druhá část se zabývá návrhem systému řešícího funkce vzešlé z průzkumu a požadavků klientské firmy. Práce zde seznamuje s obecnými postupy při vytváření aplikací na platformě ASP .NET MVC, zejména s implementací výsledného řešení, architektury a procesů. V další části práce je uvedeno, jak probíhala integrace nového systému do prostředí klientské firmy. Výsledkem práce je nová webová aplikace s aplikačním rozhraním, jejíž funkce odrážejí potřeby firem spojené s administrativními úkony vázané na projekty, jejich účtování a reporting.

Klíčová slova:

ASP .NET MVC, ASP .NET Web API, REST, správa IT projektů

Abstract

This diploma thesis is focused on development of new system which should simplify administrative agenda of small companies and agencies which are doing their business in IT sector and its related businesses. The first part of this work contains a research that has been done between companies, and the focus was on the applications that they are using and how they are rating them. This part also contains a comparison of most mentioned applications and their analysis. The second part describes a design of new application which includes functions that were mentioned by the companies in the research and was confirmed by a client company. Work describes the basic principles of developing application on ASP .NET MVC platform and mostly describes the implementation of final solution, it architecture and processes. In next part the work describes how the integration to client's company environment has been done. The result of this work is a web application with an open API which covers the functionality needed by companies and their agenda related to projects, financing and reporting.

Key words:

ASP .NET MVC, ASP .NET Web API, REST, IT project management

Obsah

Seznam tabulek

Seznam grafů

Seznam zdrojových kódů

Seznam obrázků

Cizí slova a zkratky

JS	JavaScript
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
REST	Representational State Transfer
request	Dotaz pomocí protokolu HTTP
API	Application Programming Interface, aplikační rozhraní
XML	Extensible Markup Language
AJAX	Asynchronous JavaScript and XML
DTO	Data Transfer Object

1 Úvod

Cílem této diplomové práce je vytvoření webové aplikace na platformě ASP .NET MVC, která bude sloužit k usnadnění administrace projektů menších firem z oblasti IT a přidružených oborů.

Tato aplikace by měla disponovat funkcemi, které vzejdou z průzkumu uskutečněného mezi firmami a následně budou potvrzeny partnerskou firmou Prague Labs s.r.o.. Výsledkem průzkumu je také srovnání nejčastěji zmiňovaných aplikací s ohledem na jejich funkce a cenovou politiku.

Vzniklé řešení bude možné aplikovat ve více firmách najednou s tím, že každá firma bude mít vlastní instanci aplikace v prostředí a v prvotní fázi dojde k integraci do firemních procesů partnerské firmy.

Tato diplomová práce je rozdělena do tří částí. První část je zaměřena na analýzu potřeb firem a průzkum trhu dostupných aplikací. Druhá část této práce pojednává o principech vývoje webových aplikací na platformě ASP .NET MVC. Poslední část se zabývá samotným návrhem a implementací aplikace a integrace u klientské firmy.

2 Průzkum potřeb firem

Pro vymezení cílů této práce proběhla rešerše mezi menšími firmami, která mapuje potřeby firem a jejich problémy související s řízením firmy potažmo projektů a zaměstnanců. Důvod samotného vzniku této práce je osobní zkušenost autora z prostředí malých firem, které bývají často po této stránce neorganizované a řeší svou administrativní agendu napříč mnoha aplikacemi.

2.1 Metoda průzkumu

Průzkum byl prováděn pomocí dotazníkové aplikace Google Forms[1], jež umožňuje jednoduché vytváření dotazníků. Hlavními výhodami této aplikace jsou:

- online vyplnění respondenty,
- zobrazení odpovědí v reálném čase,
- možnost měnit průběh dotazníku (následující otázky) na základě předchozích odpovědí.

Dotazník je rozdělen do několika menších části:

1. informace o firmě a respondentovi,
2. aktuálně používané aplikace pro administrativu projektů,
3. dříve používané aplikace pro administrativu projektů,
4. aktuálně používané aplikace pro měření odvedené práce zaměstnanců,
5. dříve používané aplikace pro měření odvedené práce zaměstnanců,
6. kritéria a funkce ideální aplikace, ke které by firma uvažovala přejít.

Tato struktura dotazníku byla zvolena na základě předchozí konzultace s partnerskou firmou, jelikož bylo dané, že administrativa a měření času jsou klíčové body budované aplikace, které řeší naprostá většina firem.

Ke každé části týkající se sběru informací o aplikacích je respondentovi dán prostor, aby vyjádřil subjektivní názor na pozitiva a negativa daných aplikací. Respondent pro hodnocení měl možnost zvolit ze seznamu předdefinovaných odpovědí a formulovat odpověď vlastními slovy. Tyto předdefinované odpovědi byly do dotazníku zaneseny k snazší unifikaci odpovědí od respondentů a také kvůli tomu, že pro respondenty je obecně jednodušší vybrat předdefinovanou odpověď nežli se zamyslet nad volnou odpovědí.

Kompletní náhled na strukturu dotazníku, jeho otázky a výsledky je možné najít na přiloženém médiu.

2.2 Vybrané firmy

Dotazník a budovaná aplikace jsou zaměřeny na menší firmy, jejichž působnost je blízká odvětví informačních technologií a oborům k nim přidruženým. Toto zacílení je dáno tím, že velké korporátní firmy a firmy mimo odvětví mají naprosto jiné požadavky na řízení administrativy.

Dotazník byl rozeslán firmám, s nimiž má autor této aplikace nějaké spojení – bylo tak snazší získat od firem odpověď i vzhledem k obecnému časovému vytížení reprezentantů firem. Cíleně nebyl dotazník publikován veřejně, například na internetových stránkách či fórech blízkých vybranému odvětví, a to kvůli obavě ze znehodnocení průzkumu „internetovými trolly“[1].

2.3 Analýza odpovědí

Průzkumu se celkem zúčastnilo 12 unikátních firem – unikátních proto, že z některých firem se zúčastnilo více respondentů, což do výsledků dodalo pohled více zástupců firmy, a tím i přesnější náhled na problematiku v dané firmě. Respondenti byli z řad jednatelů, projektových manažerů, případně team leaderů, tedy osob, jež mají vhled do procesů nastavených v dané firmě.

[1] Troll. IT Slovník [online]. ©2008-2018 [cit. 2018-04-20]. Dostupné z: https://it-slovnik.cz/pojem/troll

V níže uvedené tabulce (viz Tabulka 1) je seznam se základními údaji o zúčastněných firmách.

Tabulka 1 - Přehled zúčastněných firem v dotazníku

Jméno firmy	Oblast podnikání	Počet zaměstnanců
24net s.r.o.	Online vydavatelsví	11 - 20
Brightify s.r.o.	Digitální agentura	1 - 10
Cleevio s.r.o.	Vývoj mobilních aplikací, webu	21 - 30
Cloudaper s.r.o.	IT	1 - 10
NMDS s.r.o.	Marketing a Design	11 - 20
Prague Labs s.r.o.	IT software development	11 - 20
Proconom s.r.o.	Software	11 - 20
Reservanto s.r.o.	Vývoj software	1 - 10
Suitable s.r.o.	Grafické studio	1 - 10
Tapnology s.r.o.	IT	1 - 10
wpj s.r.o.	Digitální studio	21 - 30

2.3.1 Aplikace pro administrativu projektů

Z první, hlavní, části dotazníku vyplynulo, že úplně všechny firmy používají nějaké aplikace na řízení projektů a chod firmy, jejichž seznam je uveden v tabulce níže (viz Tabulka 2).

Tabulka 2 - Přehled používaných aplikací pro administrativu

Jméno firmy	Používané aplikace
24net s.r.o.	Trello, Wunderlist, basecamp
Brightify s.r.o.	JIRA, Toggl, Hackmd.io, Google Drive, Budgeta.com
Cleevio s.r.o.	Trello, CostLocker, grafické programy, GitLab
Cloudaper s.r.o.	GitHub, Google sheets
NMDS s.r.o.	Basecamp, Costlocker/Toggl, TeamGant, Slack, Google Drive
Prague Labs s.r.o.	Trello, Gitlab, Google Drive
Proconom s.r.o.	TFS
Reservanto s.r.o.	Vlastní informační systém
Suitable s.r.o.	Trello, TMetric
Tapnology s.r.o.	iDoklad, Podio, Trello, Slack
wpj s.r.o.	Trello

Po analýze zmíněných aplikací v odpovědích bylo zjištěno, že mezi odpověďmi byly i aplikace, které nejsou zcela relevantní pro daný průzkum, jelikož položená otázka byla zaměřena na přímou administrativu projektů („*Tedy aplikace, ve kterých si vedete informace o klientech, projektech, financích a věcech s tím souvisejících.*“[2]), například Slack či repositáře zdrojových kódu nejsou adekvátní aplikace k dané agendě. Výše uvedené aplikace byly ve většině případů zmíněny i jako aplikace, které firmy používaly jako předchozí řešení (76,9 % firem předtím vyzkoušelo i jinou aplikaci). To vede k závěru, že firmy střídají prakticky stejné aplikace a snaží se najít tu nejlepší, která vyhovuje jejich potřebám.

2.3.2 Aplikace pro měření času

Tři čtvrtiny dotázaných firem odpověděly, že si zaměstnanci evidují odpracovaný čas v nějaké aplikaci. Tyto aplikace se částečně shodují s uvedenými v předchozí kapitole, jelikož je běžné kombinovat evidenci nákladů na zaměstnance s finančním přehledem firmy, potažmo projektů.

Tabulka 3 - Přehled používaných aplikací pro měření času

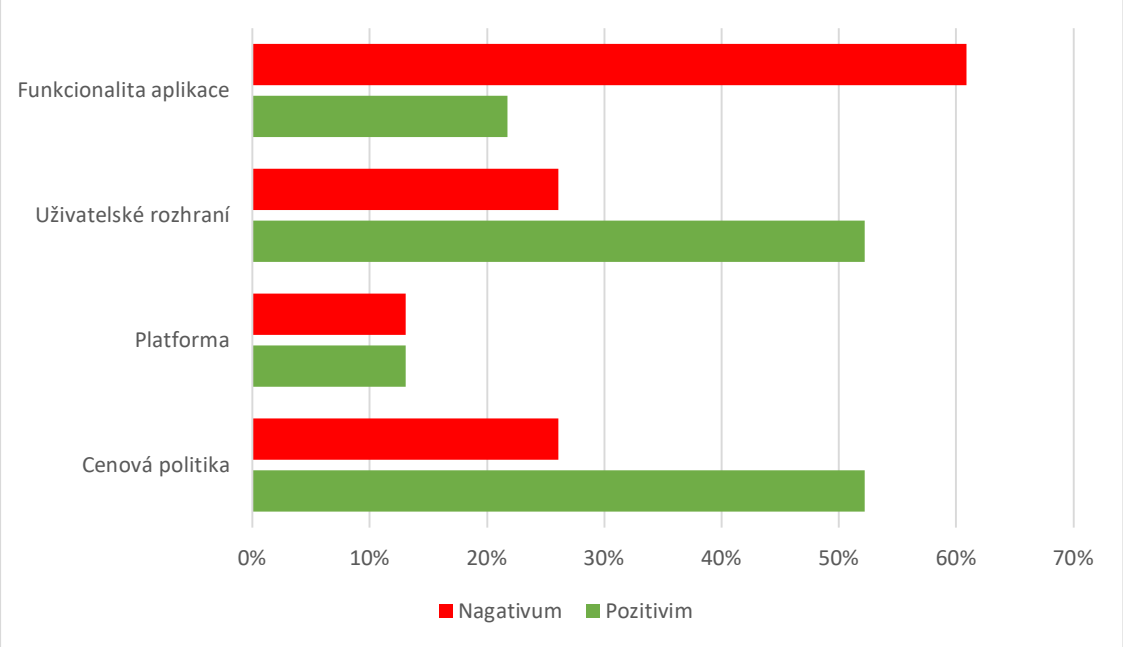
Jméno firmy	Aplikace
24net s.r.o.	Excel tabulka
Brightify s.r.o.	Toggl
Cleevio s.r.o.	CostLocker
Cloudaper s.r.o.	Toggl
NMDS s.r.o.	Costlocker, Toggl
Prague Labs s.r.o.	Interní aplikace
Reservanto s.r.o.	Vlastní aplikace pro měření času
wpj s.r.o.	Tmetric

2.3.3 Hodnocení používaného řešení

Po unifikaci subjektivního hodnocení aplikací od respondentů pro oba typy aplikace byla získána data o tom, v čem jejich uživatelé vidí jejich největší pozitiva nebo negativa (viz Graf 1). Výsledky tedy ukazují, že

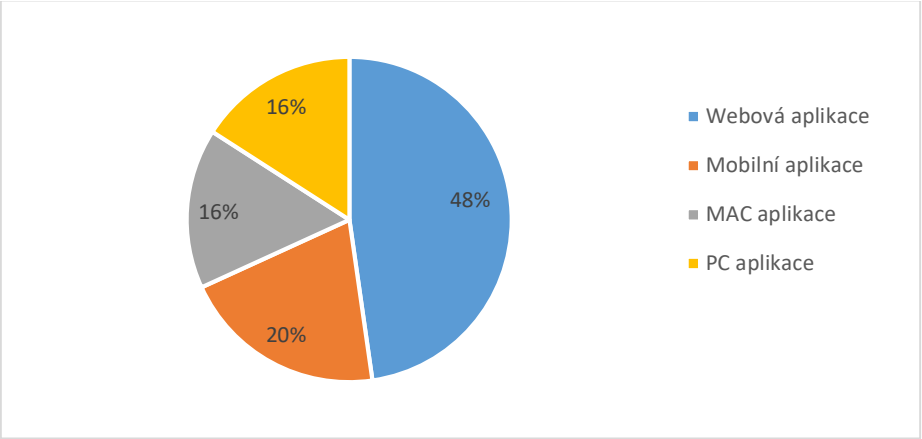
[2] Doplňující informace k otázce z dotazníku.

respondenti nejčastěji jako negativum uvedli chybějící funkcionalitu, cenovou politiku společně s uživatelským rozhraním naopak nejčastěji jako pozitivum. Cena a funkcionalita aplikací byly také dle výsledků zásadním faktorem při přechodu k novým aplikacím. V případě, že respondenti uvedli i hodnocení přechodu k nové aplikaci, byli spokojeni (alespoň částečně) v 85 % případů.



Graf 1 - Shrnutí negativ a pozitiv současně používaných aplikací

Mezi použitými platformami dominuje webový prohlížeč (viz Graf 2) následovaný mobilní platformou, což je pochopitelné vzhledem k trendům ve vývoji software; daný typ aplikací nepotřebuje žádné vysoké hardwarové nároky a postačí tak webový prohlížeč či mobilní telefon.



Graf 2 - Zastoupení platforem používaných aplikací

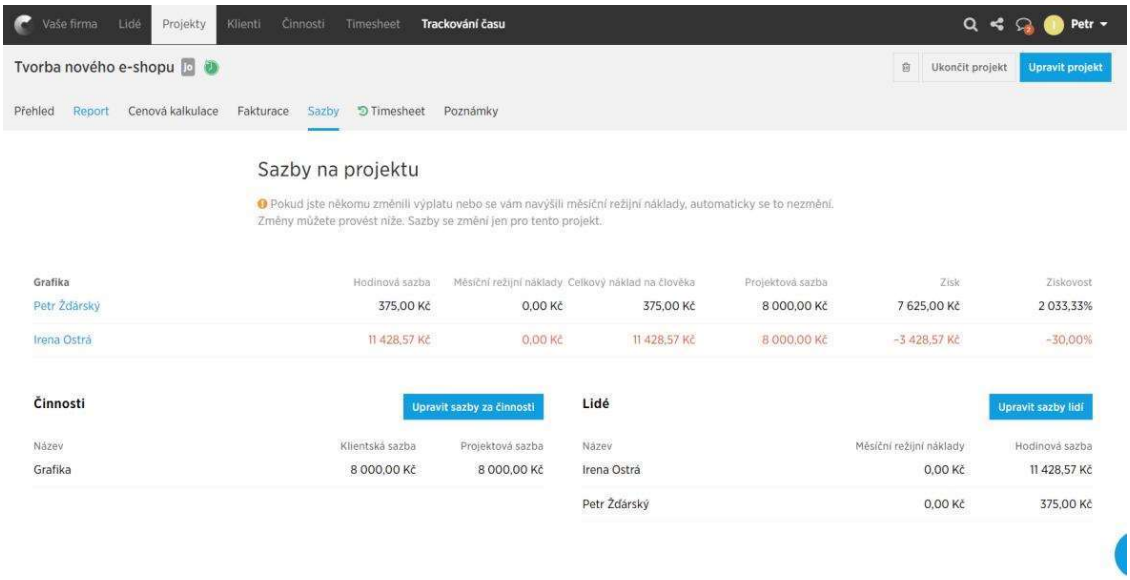
2.4 Porovnání aplikací

Na základě získaného seznamu aplikací z průzkumu (viz kapitoly 2.3.1 a 2.3.2) bylo s partnerskou firmou vybráno 5 aplikací, jež funkcemi splňují předpoklady pro komplexní nástroj pro administrativní úlohy i měření času odvedené práce zaměstnanců. Tyto aplikace jsou v následujících podkapitolách detailněji analyzovány a komparovány.

2.4.1 Costlocker

Online aplikace Costlocker[2] byla vyvinuta českým týmem a prezentuje se jako nástroj, který primárně měří ziskovost u jednotlivých projektů.

Mnoho uživatelů může činit rozhodnutí například na základě sledování výše obratu u jednotlivých projektů. To by ovšem neměl být ten nejdůležitější ukazatel, neboť dostatečně nereflektuje výši nákladů a nelze tedy vypočítat výsledný zisk. Aplikace Costlocker právě toto umožňuje a uživatel tak získává informaci o tom, který projekt je nejvíce ziskový. Následná rozhodnutí jsou tedy postavena na výsledcích porovnávání obratu a nákladů v konkrétním projektu v reálném čase.



Obrázek 1 – Náhled na sazby projektu v aplikaci Costlocker

Aplikace Costlocker umožňuje rozsáhlé nastavení. Nejprve je však potřeba zvolit, o jaký typ projektu se jedná. V tomto kroku je nutné vyplnit informace o tom, zda je projekt pouze jednorázový, nebo se opakuje a jakým způsobem je projekt klientovi účtován – cenou za hodinu, nebo jednorázovým poplatkem.

Aby však bylo možné určit ziskovost projektů, je potřeba do Costlockeru zadat náklady na provoz společnosti i odměny zaměstnanců (viz Obrázek 1). Pomocí tagů lze sledovat náklady na jednotlivé projekty. Užitečné je například sledovat náklady na výběrové řízení, neboť společnost musí nejprve investovat finanční prostředky.

Užitečnou funkcí je přiřazení tagů ke klientům. Tím je klient automaticky zařazen do určité kategorie a následně lze sledovat projekty, ve kterých klient figuruje. Costlocker taktéž nabízí možnost nadefinovat pouze konkrétní úkoly, u kterých si může zvolená osoba měřit čas.

Costlocker nabízí intuitivní rozhraní na desktopové verzi obohacené o naučná videa s tutoriály, jak vše správně nastavit. I na mobilních zařízeních lze Costlocker používat, neboť funguje na responzivní webové stránce. Aplikaci se nedaří uživateli poskytnout stejný komfort během používání, jaký

by měla právě nativní aplikace pro mobilní zařízení. Vzhledem k tomu, že většina podnikatelů v dnešní době mobilní zařízení při podnikání využívá, bylo by vhodné, kdyby měl Costlocker i svou vlastní aplikaci pro mobilní zařízení. Nicméně přínosem jsou desktopové aplikace pro operační systémy MacOS a Windows, jež umožňují měření času.

V detailu projektu jsou znázorněny základní finanční metriky, měření času lze v aplikaci zadávat pouhým zapnutím a vypnutím časovače nebo vykazováním času zpětně. Costlocker neumí vystavovat faktury, ale je možné pomocí něj sledovat projekty, ke kterým se faktura teprve bude vydávat. Informace o tom, kolik se má vyfakturovat v budoucnu jsou užitečné z hlediska cash flow. Aplikace, přestože česká, má ambice získat přízeň klientů i na zahraničních trzích a podporuje zahraniční měny a další nastavení typické pro odlišné regiony.

Velmi vítanou vlastností je možnost implementace API. Například je možné, aby se notifikace z Costlockeru propisovaly do aplikace Slack nebo Asana. Umožňuje také propojení s aplikací Gettrafika – ta je napojena na společný kalendář a lze sledovat utilizaci lidí. Dále se dá jednoduše napojit na Fakturoid, iDoklad a Basecamp.

Costlocker nabízí třicetidenní zkušební verzi bez nutnosti zadávat číslo platební karty. Následně stojí jeden měsíc používání pro jednoho uživatele dvanáct až dvacet amerických dolarů (dle četnosti plateb).

2.4.2 Basecamp

Basecamp[3] je mnohými považován za nejvíce intuitivní aplikaci používanou (nejen) k měření času. Aplikace umí skvěle řešit komunikační problémy mezi osobami, které se podílí na projektu, a zároveň nabízí velmi pěkné a dobře srozumitelné prostředí. Velikou výhodou je také dostupnost aplikace na zařízeních se systémem Android, iOS, MAC, PC, samozřejmě ji lze používat i skrze webové rozhraní. Dostupné je také API aplikace.

Samotná aplikace se dělí na tři hlavní sekce – manažerská, týmová a projektová sekce. Manažerská sekce představuje jakousi centrálu projektu, kam autoři aplikace doporučují přidat všechny osoby, které se na projektu podílejí.

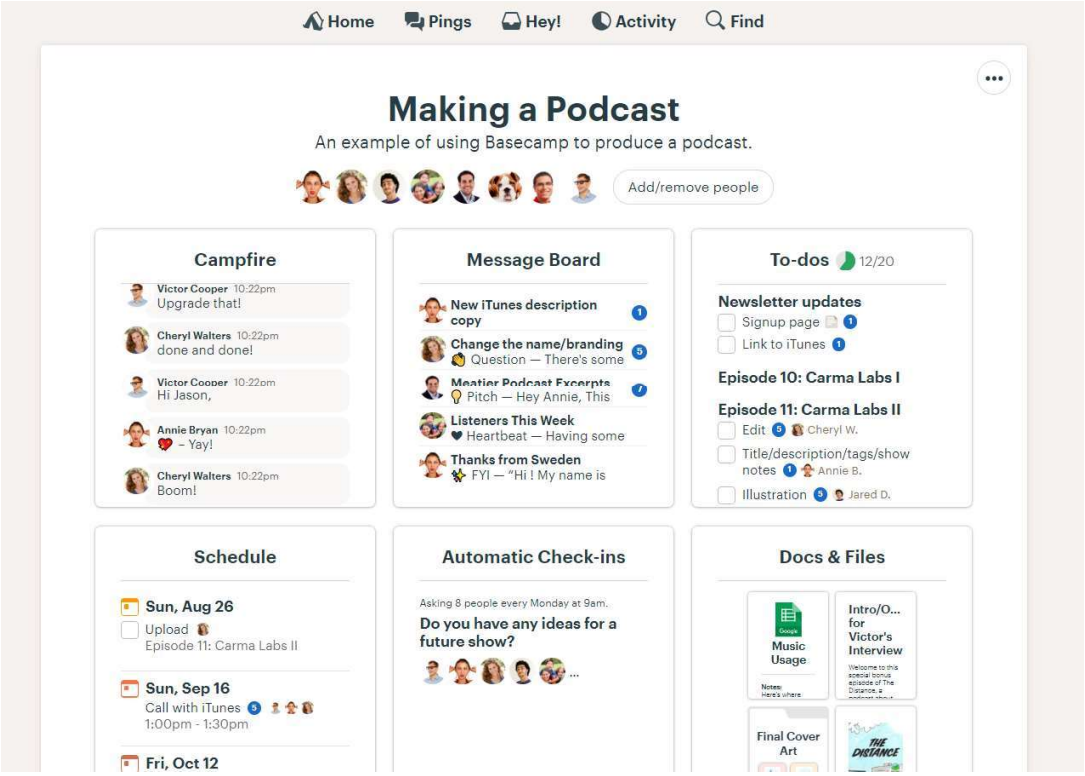
Týmová sekce dává možnost se každého uživatele na konci pracovního dne automaticky dotázat, na čem daný den pracoval a kolik úkolů stihnul splnit. Odpovědi dotazovaných uživatelů jsou následně přístupné všem ostatním. Celý tým má tedy dobrý přehled o tom, na čem pracují ostatní, a to bez potřeby trávit čas na společných projektových poradách. To je i díky tomu, že všichni účastníci projektu by měli díky Basecamp velmi dobře pochopit svou roli i zodpovědnost za úkoly, které jim byly přiděleny. Díky Basecamp mohou být například uživatelé okamžitě upozorněni na to, jaké problémy je potřeba vyřešit a kolik času jim na to bylo přiděleno.

Týmová sekce je dedikována pouze osobám, které pracují ve stejném oddělení, respektive mají stejnou roli. Projektová sekce je vhodným místem pro ty, kterým jsou přiděleny různé role, ale musejí navzájem spolupracovat.

Basecamp má kromě tří základních sekcí několik dalších užitečných vlastností. Pomocí sekce Message Board lze jednoduše pracovat s informacemi ohledně vývoje práce. Navíc je sbírána i zpětná vazbu na práci zúčastněných.

To-do sekce, jak název napovídá, umožňuje uživatelům přiřadit úkoly a sledovat, kdo který úkol splnil či nesplnil. Schedules sekce zobrazuje důležité termíny pro projekt, zatímco Docs and Files je místo, kam lze nahrát všechny multimediální soubory, u kterých je žádoucí, aby byly všem k dispozici.

Campfire slouží jako místo pro chatování, kde se každý uživatel může zeptat ostatních, a to i na dotaz, u nějž si není jistý, komu přesně by měl být určen (rozcestník k této a dalším uvedeným funkcím je možné vidět na Obrázek 2). Basecamp umožňuje mít spuštěných více projektů zároveň. Ke každému projektu může být přidělen jiný tým.



Obrázek 2 – Ukázka dashboardu aplikace Basecamp

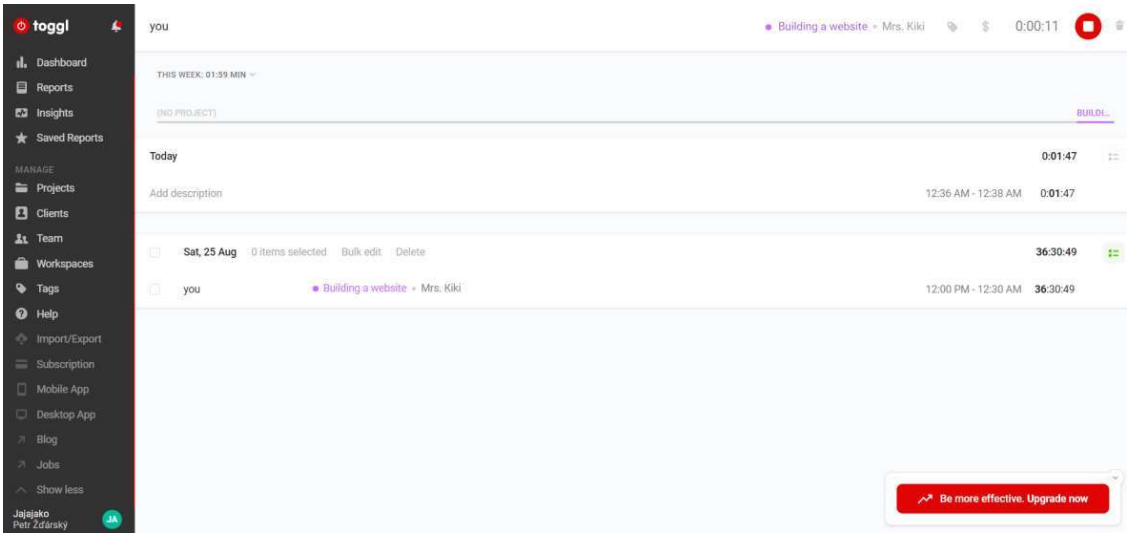
Velmi užitečnou funkcí je také možnost nastavit, jaké informace se zobrazí klientovi, čímž vznikne prostor pro případnou zpětnou vazbu klienta na práci týmu. Pokud by klient nechtěl využívat Basecamp, lze nastavit automatický import e-mailů od klienta, tudíž bude vše zachyceno i v samotné aplikaci.

Basecamp lze pořídit za 99 dolarů měsíčně a tak získat 500GB úložného prostoru a možnost administrace neomezeného počet uživatelů a projektů.

2.4.3 Toggl

Toggl[4] je jedna z nejrozšířenějších aplikací sloužících primárně k měření času. Její předností je jednoduchost, uživatel se tedy nemusí zabývat složitým nastavováním měření času, aplikace vše vyřeší za něj. Navíc není potřeba se omezovat pouze na jedno zařízení, jelikož aplikace je multiplatformní. Je tedy možné začít pracovat na desktopu a později například dokončit práci na tabletu.

Obdobně jako Costlocker poskytuje takzvanou „idle“ funkci – pokud je uživatel nečinný, přestane aplikace čas měřit nebo se zeptá, co má dál udělat (na Obrázek 3 lze vidět rozhraní pro měření aktivit). Toggl umožňuje sledovat, který z projektů má adekvátní finanční návratnost a který se nevyplatí, což vede následně k lepšímu rozhodování. Další funkcí, kterou sdílí s Costlockerem je možnost vytvářet tagy.



Obrázek 3 – Rozhraní měření aktivit v aplikaci Toggl

Aplikace nabízí základní a rozšířenou verzi. V rozšířené verzi navíc umožňuje business intelligence a dostávat tak reporty e-mailem a automatizovat časové plány.

Aplikace je dostupná na desktopu, iOS, Android a samozřejmě také přes webové rozhraní. Také podporuje rozšíření do prohlížeče Chrome a Firefox, které umožňuje měřit strávený čas. Je možno také pracovat s API. Výhodou je, že umožňuje pokročilou integraci do více než sta dalších aplikací včetně Basecampu, Asany, Evernote, Google Drive a dalších.

Přidanou hodnotou je také možnost tvorby reportů, které lze následně vyexportovat ve formátech PDF, CSV a XLS.

Toggl nabízí čtyři různé druhy členství – Free, Starter, Premium a Enterprise. Všechny verze s výjimkou varianty zdarma podporují tagy, režim

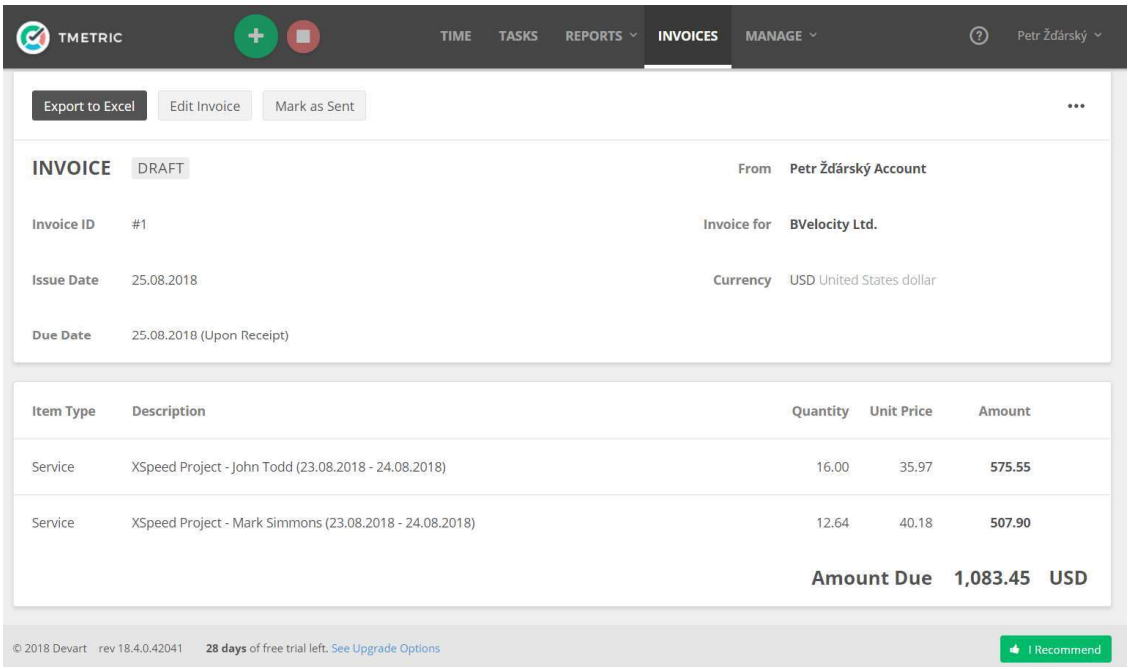
nečinnosti, exporty, neomezenou velikost týmů. Za verzi Starter si Toggl nechá zaplatit 9 dolarů za uživatele na měsíc. Verze Premium pak stojí 18 dolarů a vrcholná varianta Enterprise na 49 dolarů měsíčně.

2.4.4 TMetric

TMetric[5] měří čas a dává uživateli kontrolu nad rozpočty jednotlivých projektů. Umožňuje nastavit rozdílné sazby pro zaměstnance a kontraktory a podporuje také reporting. Navíc nabízí přehlednou týmovou spolupráci a integraci do desítek aplikací.

Aplikace zároveň zhodnotí výkonnost týmu, neboť poskytne dobrý přehled nad tím, co tým udělal a jaký čas mu tato činnost zabrala. Odpadá tedy potřeba manuálního reportování. Dokáže přiřadit úkoly jednotlivým členům týmu, sledovat jejich pracovní nasazení a porovnat všechny výsledky na jedné obrazovce.

Aplikace nabízí kvalitní pokrytí téměř všech existujících platforem – iOS, Android, desktop, Linuxu a také doplňky do prohlížečů Chrome, Firefox, Opera a Edge. Obdobně jako Toggl umožňuje měřit čas přímo v prohlížeči. Aplikace je připravena na integraci s přibližně padesáti jinými nástroji a umožňuje i fakturování práce přímo v aplikaci.



Obrázek 4 – Náhled na fakturovanou práci v aplikaci TMetric

Aplikace má pěkný a přehledný interface a během prvního seznámení s aplikací může uživatel zhlédnout hned několik video tutoriálů, které mu vše vysvětlí. Mnoho uživatelů této aplikace oceňuje právě jednoduchost rozhraní aplikace, která je nezahlcuje zbytečnými informacemi tak, jako u konkurenčních aplikací.

Již základní bezplatná verze Free může mít až 5 uživatelů, funkci měření času, reporting a integraci s jiným aplikacemi. Verze Professional za 4 dolary měsíčně za uživatele (případně 5 dolarů při platbě měsíčně) nabízí nastavení cen, které jsou účtovány zákazníkovi, a fakturaci. Schopnost vyfakturovat poskytnuté služby je jednou z velkých výhod aplikace, neboť se u většiny konkurenčních aplikací nenachází. Dále také nabízí možnost vytvářet úkoly, to-do listy a umožnění přístupu do aplikace až třem klientům. Nejvyšší Business verze stojí 6 dolarů měsíčně za uživatele (7 dolarů při platbě měsíčně). Přidává možnost rozlišovat náklady na zaměstnance a kontraktory, synchronizaci s aplikací JIRA a Redmine a neomezený počet klientů s přístupem.

2.4.5 Budgeta

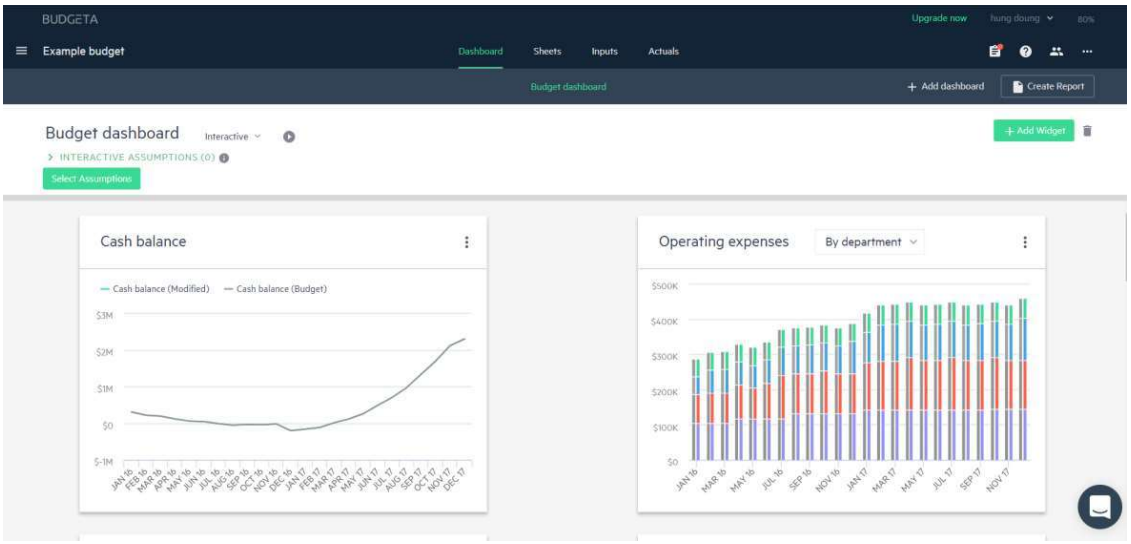
Aplikace Budgeta[6] představuje jednoduchou aplikaci sloužící k vytváření rozpočtů pro startupy, menší a střední podniky. Dle tvůrců aplikace Budgeta je klíčovou funkcí „What-if scenarios“. Funkce umožňuje prozkoumat několik odlišných scénářů, podle kterých se rozpočet může vyvíjet. Uživatel se díky této funkci může lépe připravit na reálné situace, které eventuálně mohou nastat. Tato funkce umožňuje v rozpočtu upravit jednotlivé položky a následně pozorovat, jaký bude mít změna údajů vliv na rozpočet. Uživateli je zároveň prezentováno srovnání s původním rozpočtem.

Rozpočty lze jednoduše sdílet a upravovat napříč všemi zainteresovanými stranami, přičemž ty mohou být rovněž znázorněny graficky a upraveny tak, aby sledovaly klíčové ukazatele ziskovosti.

Jakmile si uživatel vytvoří profil, může okamžitě naimportovat své stávající rozpočty ve formátu .xlsx a veškeré údaje upravit dle potřeb. Ihned tak může pozorovat, jaký účinek každá úprava má. Samozřejmostí je také export do zmíněného formátu.

Dle počtu požadovaných funkcí nabízí tvůrci aplikace čtyři různé varianty – Basic, Advanced, Pro a Enterprise. Basic varianta, která je přístupná zdarma, umožňuje využívat jen tak základní funkce, jako je například plánování, z finančních výkazů umožňuje sledovat pouze cash flow. Dá se sdílet až mezi pěti uživateli a dashboard aplikace se nedá upravovat. Ke všem placeným variantám má uživatel třicetidenní zkušební lhůtu.

Nejlevnější placená varianta Advanced za 75 dolarů měsíčně zaznamenává historii úprav, nabízí „What-if scenarios“, má neomezený počet uživatelů, upravitelný dashboard a kromě výkazu cash flow umí pracovat i s rozvahou a výkazem zisku a ztráty (viz Obrázek 5).



Obrázek 5 – Ukázka přehledu rozpočtu v aplikaci Budgeta

Pro varianta nabízí za cenu 200 dolarů měsíčně porovnávání rozpočtů se skutečnými náklady projektu a až nejdražší verze Enterprise nabízí za 500 dolarů měsíčně také API sloužící k propojení s účetním systémem. Bohužel se dá využít pouze pro software Sage-Intacct, tudíž se API nedá považovat za opravdu přínosné.

Vizuálně je prostředí velmi jednoduché, zároveň přehledné. Nevýhodou pro mnohé uživatele může být to, že Budgeta neumožňuje měřit čas obdobně jako například aplikace Toggl nebo Costlocker, jež mají z pěti srovnávaných aplikací právě k aplikaci Budgeta nejblíže.

Tvůrci bohužel nenabízejí další aplikace pro mobilní zařízení či desktop. Aplikace je responzivní a přizpůsobí se velikosti obrazovky. Práce na zařízeních s menší obrazovkou však moc pohodlná není. Aplikaci Budgeta lze tedy plnohodnotně využívat pouze na počítači.

2.4.6 Závěrečné porovnání

Všechny uvedené aplikace nabízejí zajímavé spektrum funkcí, přestože některé se soustředí více na konkrétní problematiku (například tvorbu rozpočtů), zatímco jiné se snaží oslovit více zákazníků nabídkou většího množství funkcí. Každá z těchto aplikací si najde své uživatele, je tedy obtížné

určit, která z aplikací poskytuje nejlepší funkce, neboť vždy záleží na konkrétních potřebách zákazníka.

U všech testovaných aplikací bylo zjištěno, že podporují funkci reportingu, nabízí API a s výjimkou aplikace Budgeta umožňují měřit čas. Všechny aplikace také nabízejí třicetidenní zkušební dobu, během které může uživatel zjistit, zdali aplikace vyhovuje jeho potřebám. Srovnání funkcí a cen aplikací je možné nalézt v Příloha B.

Větší rozdíl lze pozorovat až u ceny jednotlivých aplikací. Aplikace Costlocker, Toggl a TMetric si nechávají platit měsíční poplatek za každého uživatele. Aplikace Budgeta a Basecamp se vydaly jinou cestou a za poplatky desetinásobně větší nabízejí služby neomezenému počtu uživatelů a projektů.

V oblasti podpory aplikace napříč různými platformami na tom jsou velmi dobře aplikace TMetric, Basecamp a Toggl. Nejhůře si z tohoto hlediska vedla aplikace Budgeta, která je použitelná prakticky pouze na desktopu.

Na základě subjektivního názoru autora této diplomové práce vyšla jako nejvýhodnější aplikace Basecamp, která jako jediná z testovaných umožňuje vystavovat faktury přímo v aplikaci, nabízí velmi přehledný interface, který se pohodlně používá, a dá se velmi dobře provázat s aplikací Toggl. Za poplatek 99 dolarů měsíčně nabízí neomezené užívání aplikace.

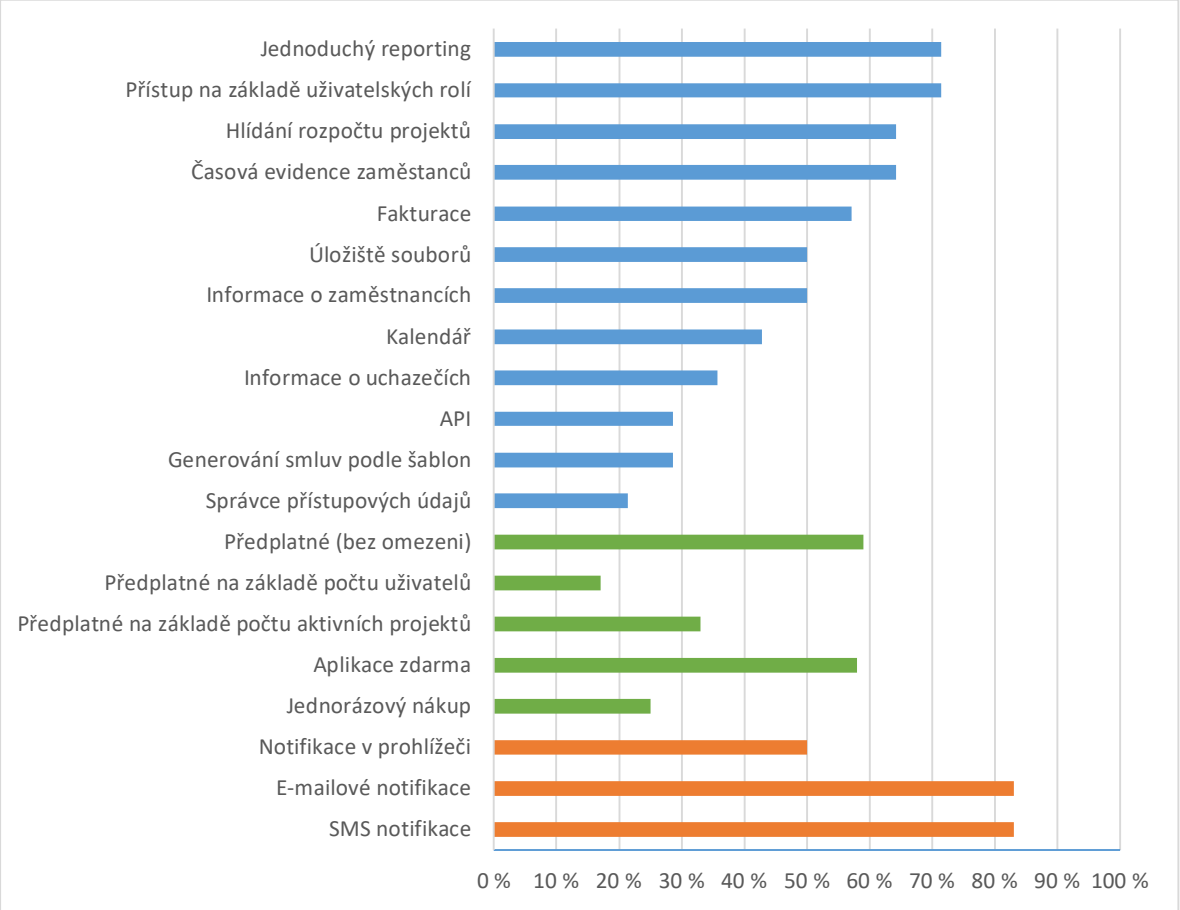
Uživatelé, kteří chtějí primárně sledovat náklady, rozpočty a ziskovost, by mohli také uvažovat o české aplikaci Costlocker. Ta nabízí užitečnou funkci předpovídání budoucího vývoje na základě předchozích zisků a příjmů projektu, má solidní dostupnost na různých platformách, přívětivou cenu a výtečnou podporu.

2.5 Závěry průzkumu

Poslední část dotazníku se zabývala ochotou firem přejít k nové aplikaci a zjištěním jejich potřeb pro danou aplikaci. Naprostá většina zúčastněných firem (92,3 %) by zvažovala o přechodu k jiné aplikaci, a to za předpokladu,

že by tato aplikace splňovala jejich kritéria na funkcionalitu a cenovou politiku řešení (viz Graf 3). Tento výsledek odráží i to, že cena a funkce jsou hlavním kritériem při výběru aplikace.

Průzkum tedy dodal sadu funkcí, které firmy očekávají od ideální aplikace, v neposlední řadě také představu o cenové politice, pokud by výsledná aplikace byla převedena do komerčního provozu.



Graf 3 - Zastoupení požadovaných funkcí v ideální aplikaci

3 Vymezení požadovaných funkcí aplikace

Na základě informací získaných z průzkumu (viz kapitola 2.3) a požadavků partnerské firmy došlo k vymezení funkcí pro základní verzi aplikace, která bude splňovat zejména požadavky partnerské firmy.

Nezbytným požadavkem byla taková architektura systému, která by umožňovala současný provoz více firem (instancí) v případě, že bude aplikace v budoucnu používána komerčně. Tento požadavek je splnitelný při správném použití a integraci databázového frameworku poskytnutého partnerskou firmou. Aplikace tedy bude obsahovat systémovou administraci pro správu jednotlivých instancí a jejich systému předplatného, na jehož základě je jim umožněn přístup do systému. Model předplatného však bude pouze připraven, jelikož zatím nepůjde o komerční produkt, který by měl přesně definovaný business model. Samozřejmým předpokladem je vytvoření části aplikace umožňující používání aplikace samotnými firmami.

Další součástí systému by měla být správa projektů a jejich přidružených prací. To znamená, že každý projekt se bude skládat z více součástí, kde každá součást má vlastní ceníkové sazby. Další součástí systému musí být správa informací o klientech a zaměstnancích firmy.

Systém také musí zvládnout řešit finanční stránku projektů a zaměstnanců vzhledem k odpracovanému času a jejich sazbám. Vzhledem k tomu, že firma disponuje vlastním řešením pro měření času (viz 6), bylo jedním z požadavků vytvoření aplikačního rozhraní, které umožní synchronizovat data do systému. Možnost napojení aplikací třetích stran bylo i jedním z požadavků respondentů. V základní verzi aplikace tedy nebude existovat jiný způsob měření času než přes ono aplikační rozhraní. V případě komerčního použití výsledné aplikace je nasnadě upravení existující aplikace i pro použitelnost dalšími firmami, tedy mít 2 aplikace: tenkého klienta na měření času a plnohodnotný systém.

V souvislosti s finanční funckionalitou systému musí systém dokázat evidovat tok peněz napříč virtuálními účty v aplikaci, které mohou být vedeny v různých měnách. Napojení na finanční/účetní aplikace, jako je iDoklad či POHODA, jsou mimo rámec požadavků na minimální verzi aplikace.

Jedním z požadavků respondentů byl také přístup na základě uživatelských rolí, nicméně po diskuzi s partnerskou firmou se došlo k závěru, že tato funkcionalita bude posunuta až k další verzi aplikace, jelikož každá další uživatelská role bude obsahovat pouze nějakou podmnožinu ze všech funkcí systému. Výsledná aplikace tedy bude sloužit zejména vyššímu managementu zástupců firmy, kteří jsou oprávněni k přístupu k citlivým datům, jako jsou sazby klientů či zaměstnanců.

Další funkcí by měla být správa uchazečů o zaměstnání, jelikož v současné době na to partnerská firma nemá definované žádné procesy a celý systém je založen pouze na jedné sdílené složce mezi lidmi, kteří mají na starosti lidské zdroje.

Vzhledem k tomu, že do aplikace bude mít přístup omezený počet lidí z dané firmy, docházíme k závěru, že alternativní způsoby přihlašování do systému (jméno a heslo vůči 2 fázovému ověřování nebo Google autentizaci), jež byly součástí průzkumu stejně jako notifikace produkované systémem, budou součástí následného evolučního procesu.

Definované funkce zdaleka neobsahují všechny funkce, které by pokryly každodenní agendu firmy nebo které vzešly z výsledků průzkumu. Nicméně tato aplikace bude v prvotní fázi sloužit zejména jako důkaz o proveditelnosti konceptu a její další vývoj bude v režii partnerské firmy.

4 Použité technologie

Pro implementaci webové aplikace byl zvolen framework ASP .NET MVC[7], jelikož se jedná o nejpoužívanější framework pro webové aplikace na platformě .NET a autor této práce s ním má dobré zkušenosti. Existuje i jeho varianta ASP .NET Core MVC, jehož popularita je značně na vzestupu, (vzhledem k jeho výkonnosti a možnosti provozu na více platformách oproti platformám firmy Microsoft). Avšak tato varianta frameworku nebyla vybrána, jelikož některé z knihoven, jež jsou použity ve frameworku poskytnutého partnerskou firmou, nemají své alternativy pro .NET Core a migrace celého frameworku, na kterém je závislý business firmy, je v rámci této práce příliš riskantní.

Pro vývoj aplikačního rozhraní byl zvolen framework ASP .NET Web API, jelikož jde opět o nejrozšířenější způsob vývoje aplikačních rozhraní na dané platformě a „přepoužitelnost“ některých součástí použitých pro MVC aplikace je nasnadě.

Tato kapitola seznamuje s principem frameworků ASP .NET MVC a Web API představením databázového frameworku poskytnutého partnerskou firmou.

4.1 ASP .NET MVC

ASP .NET MVC je webový aplikační framework od firmy Microsoft založený na návrhovém vzoru Model-View-Controller (MVC) a slouží k vytváření aplikací, jejichž struktura je složena ze tří nezávislých komponent. Na základě tohoto rozložení komponent je umožněn jejich paralelní vývoj a znovupoužitelnost v jiných aplikací.

Model

Model je nejuniverzálnější část celé architektury, jelikož se nestará o prezentační logiku aplikace. Definuje tzv. doménový model, jenž modeluje vztahy reálného světa. Doménový model se skládá z datového modelu, jenž popisuje datové struktury (například schéma databázové modelu), a dále

z business logiky samotné aplikace. Model tedy není v kontextu MVC jenom datovou strukturou, ale nese značnou část funkcionality celého systému.

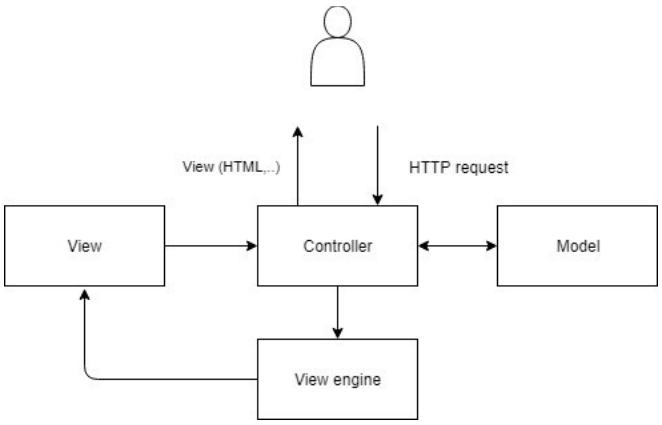
Controller

Controller je zásadní součástí MVC architektury, jelikož se jedná o prostředníka mezi Modelem a uživatelem používajícím aplikaci. Controller tedy reaguje na uživatelovu interakci a je přímo napojen na Model, na němž může provádět operace, přijímá tedy data, která může validovat, a přesouvá je dále ke zpracování do Modelu. Výsledkem této interakce je *View*.

View

View je přímým bodem interakce mezi uživatelem a aplikací, jelikož se jedná o samotné uživatelské rozhraní aplikace, které prezentuje data a akce uživateli.

Architektura MVC aplikovaná na framework ASP .NET MVC je znázorněna na obrázku Obrázek 6.



Obrázek 6 – MVC architektura aplikovaná na framework ASP .NET MVC

Samotný framework ASP .NET MVC obsahuje nespočetné množství funkcí, jež usnadňují vývoj webových aplikací. Detailnější popis funkcionality frameworku není předmětem této práce a mohl by být námětem pro samostatnou kvalifikační práci, nicméně představa o principu frameworku je zřejmá z poskytnutého vysvětlení MVC architektury.

4.2 ASP .NET Web API

Aplikační rozhraní je vybudované na platformě ASP .NET Web API[8] a vyhovuje standardům REST architektury, jejíž principy byly poprvé sepsány v dizertační práci Roye Fieldinga[9]. Základní vlastnostmi této architektury jsou:

- Uniform interface,
- Cachaeble,
- Stateless,
- Client-Server,
- Layered system,
- Conde on Demand.

Shrnutí jednotlivých vlastností (viz následující odstavec) přináší představu o tom, jak REST funguje v porovnání s jinými architekturami webových služeb.

Jedná se o Resource-based službu, tzn. že je služba orientována na zdroje (nikoli například na funkce/akce); těmito zdroji mohou být přímo záznamy v databázi nebo zpravidla DTO (Data Transfer Object) objekty, které kumulují data ze systému a v potřebné podobě je vystavují na API. Každý tento zdroj je identifikován unikátní URI (tedy adresou) a akce nad ním jsou prováděny pomocí metod, jež jsou definovány v protokolu HTTP. Tyto metody jsou:

- GET – slouží k získání zdroje na dané URI,
- HEAD – identická s GET metodou, avšak nevrací data, ale pouze meta informace v hlavičce dotazu,
- POST – slouží zpravidla k vytvoření nového zdroje v aplikaci,
- PUT – slouží k editaci zdroje, anebo jeho vytvoření, pokud neexistuje,
- PATCH – slouží k částečné aktualizaci zdroje a nevyžaduje poslání všech informací zdroje,

- DELETE – slouží k odstranění zdroje.

Odpovědi takového rozhraní zpravidla obsahují tělo odpovědi (samotná data) a také HTTP stavový kód – tyto kódy jsou definovány v protokolu (existuje jich celá řada) a poskytují jednoznačnou informaci o tom, co se se zdrojem stalo. Příklady nejčastěji používaných stavových kódů:

200 OK – zdroj byl nalezen a je součástí odpovědi,
201 Created – zdroj byl vytvořen,
204 NoContent – akce nad zdrojem byla vykonána, ale server nevrací žádná data, například při mazání zdroje,
400 BadRequest – dotaz nemůže být zpracován, například kvůli syntaktické chybě,
401 Unauthorized – autorizace nebyla schválena pro daný dotaz,
404 NotFound – zdroj nebyl naleze
500 InternalServerError – na serveru vznikla chyba při vykonávání dotazu.

Formát, který je používán pro popis objektů, není součástí REST architektury, každá aplikace tak může obsahovat vlastní definici formátu odpovědí, zpravidla se však jedná o nějakou variantu JSON nebo XML dokumentů.

Součástí odpovědí jsou také meta informace v hlavičce dotazu a odpovědi, jež například informují o tom, zda odpověď serveru může být cachována (ukládána do paměti klientské aplikace) a po jakou dobu – to umožňuje snížit datovou náročnost aplikací u dat, která se zřídkakdy mění.

Struktura ASP .NET Web API (viz Obrázek 7) aplikací je velice podobná MVC aplikacím, jelikož se jedná o podobné principy, jediným rozdílem je, že Web API má místo prezentační vrstvy veřejné rozhraní, které je určené pro komunikaci s aplikacemi třetích stran.

Obrázek 7 – Web API architektura

Informace v odstavcích výše slouží pouze k představení základních principů REST architektury a zdaleka nepopisují všechny. Architektura je sama o sobě natolik obsáhlá, že by mohla být použita jako námět pro samostatnou kvalifikační práci.

4.3 PTS framework

PTS framework poskytnutý firmou Prague Labs s.r.o., jehož autorem je pan Paul Tesar BSc., slouží jako základní vrstva pro všechny systémy, jež firma buduje. Tento framework má v sobě implementován základní doménový model, který je společný napříč budovanými systémy, jedná se tedy například o entity reprezentující uživatele, objednávku, kalendář a věci jim relevantní. Funkcionalita aplikací, jež není aplikovatelná na existující entity, je předmětem diskuze a je nadále abstrahována tak, aby byla relevantní i v jiných projektech.

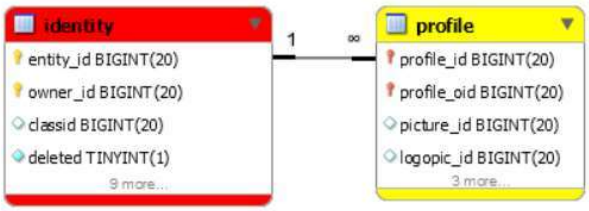
4.3.1 Přístup do databáze

Framework je koncipován jako model-first, tzn. že třídy reprezentující databázové entity jsou definovány až na základě modelu databáze. Tyto třídy jsou staticky definovány, díky tomu je celý framework daleko méně náročný na zdroje serveru než jiné frameworky. Například Entity framework, jenž se standardně využívá pro vývoj aplikací na frameworku ASP .NET MVC, využívá reflexi – tato technika je mnohonásobně náročnější na zdroje. Opačným přístupem, jenž poskytnutý framework nepodporuje, je code-first, kdy dochází k definici databázového modelu v kódu aplikace a databáze se poté na základě definice generuje.

Framework naprosto odděluje přístup od databáze pomocí handlerů a jiných funkcí, programátor tak vždy pracuje s objekty. Toto oddělení

umožňuje záměnu databázového systému a programátor nemusí znát žádný dotazovací jazyk. Framework v sučasné době využívá MySQL databázový systém, ovšem jeho výměna je možná a spočívá ve vytvoření třídy, která implementuje rozhraní sloužící k definici syntaxe jazyka a generování dotazů.

Framework ve značné míře využívá vztahu dědičnosti mezi entitami, které jsou na úrovni databázového modelu řešeny jako cízi klíče, jež jsou nastaveny zároveň jako primární klíče (viz Obrázek 8). Dědičnost tak umožňuje sdílení části společné business logiky pro dané entity. Framework se automaticky stará o získávání a vytváření potřebných záznamů v tomto vztahu.



Obrázek 8 – Ukázka řešení dědičnosti v databázovém modelu.

Definice entit

```
public static class PictureFields
{
   public static readonly Table Table = Table.Make( "picture" );

   public static readonly Field<ulong> Id =
      Table.AutoIdField<ulong>( "picture_id" );
   public static readonly Field<ulong?> OwnerId =
      Table.PrimaryField<ulong?>( "picture_oid" );

   public static readonly Field<string> Guid =
      Table.NullField<string>( "guid" );
   public static readonly Field<string> Name =
      Table.NullField<string>( "name" );
}
```

Zdrojový kód 1 – Ukázka definice entity

Každá entita má staticky definované pole (viz Zdrojový kód 1), následně je k ní vytvořen generický handler (viz Zdrojový kód 2; pozn.: tento handler může být rozšířen o libovolnou funckionalitu), jenž se stará o CRUD operace:

- C – create (vytvoření),
- R – read (čtení),
- U – update (aktualizace),
- D – delete (mazání).

```
public class PictureHandler : OwnedEntityHandler<Picture>
{
  public PictureHandler( )
    : base( PictureFields.Table, PictureFields.Id,
            PictureFields.OwnerId, PictureFields.Guid )
  { }
}
```

Zdrojový kód 2 – Ukázka definice handleru entity

K entitě je také možné vytvořit filtr (viz Zdrojový kód 3), jenž umožnuje selektivní vybírání záznamu z databáze. Tento filtr je definován jako třída

```
public class PictureFilter : OwnedEntityFilter<ulong>
{
  public enum SortOptions {  Name }
  public enum SearchOptions  {    Name }

  public SortMap<SortOptions> Sort { get; } = new
SortMap<SortOptions>( );
  public Index<SearchOptions> SearchIn { get; } = new
Index<SearchOptions>( );
  public List<string> Names { get; } = new List<string>( );
  public DateRange DateTime { get; } = new DateRange( );

  public override void ApplyToDataSet( Dataset dts )
  {
    dts.FilterRange( PictureFields.DateTime, this.DateTime );
    dts.OrderBy( this.Sort.Get( SortOptions.Name ),
      PictureFields.Name );
  }
}
```

Zdrojový kód 3 – Ukázka definice filtru entity

s výčtem možných filtrovacích, řadících a hledacích hodnot, které budou použity při generování dotazu do databáze.

Práce se entitami

Vytvoření záznamu v databázi probíhá přes handler, který náleží oné entitě. Vzhledem k tomu, že handlery jsou také vázané k instanci aplikace, dochází k automatické inicializaci části primárního klíče, jež definuje náležící instanci. Pro vytvoření nového objektu (viz Zdrojový kód 4) je na handleru volána metoda *Create*, po nastavení vlastností tohoto objektu dojde k uložení do databáze pomocí metody *Save*. Tato metoda také vrací boolean hodnotu, zda byl záznam úspěšně zapsán do databáze. Databázový handler novým záznamům automaticky přiřazuje inkrementální ID, jehož hodnota je poskytnuta databází. Po volání metody *Save* je tedy k dispozici kompletní informace o záznamu včetně jeho příslušných primárních klíčů.

```
Picture pic = Instance.M<PictureHandler>().Create();
...
pic.Save();
```

Zdrojový kód 4 – Vytvoření nového záznamu v databázi

V případě, že entita nemá autoinkrementální identifikátor, je možné tuto hodnotu zadat jako parametr metody *Create*. Velice užitečnou funkcí je také metoda metoda *Allocate*, jež pro zadanou hodnotu primárního klíče nejprve zjistí, zda záznam s touto hodnotou již existuje, a pokud ne, dojde k vytvoření nového záznamu. Výsledkem této metody je rozhraní *IAllocationResult,* jež kromě objektu samotné entity v sobě nese informaci o tom, zda byl záznam vytvořen, nebo zda byl načten z databáze.

Pro aktualizaci záznamu v databázi je nutné pouze nad daným objektem entity zavolat metodu *Save;* entita samotná v sobě nese informaci o tom, které její vlastnosti byly změněny, a pouze tyto vlastnosti budou součástí aktualizačního dotazu do databáze. Pro smazání záznamu z databáze je nutné zavolat metodu *Delete* nad daným objektem, dojde tak k jeho

odstranění. Způsobům, jak získat záznam z databáze, se věnuje následující kapitola.

Získávání záznamů z databáze

Framework disponuje několika způsoby získání záznamů z databáze, přičemž ten nejjednodušší je za použití příslušného handleru a volání metody *Get* s parametrem primárního klíče požadovaného záznamu. Tento způsob umožnuje jednoduché získání jednoho konkrétního záznamu, u kterého je předem znám identifikátor, který je například dodán jak parametr URL adresy.

Dalším způsobem získání či kolekce záznamů je použití filtru, kterému jsou dány vyhledávácí kritéria. Filtry jsou nastaveny tak, že jsou pomocí případných cizích klíčů propojeny s filtry propojených entit. Na níže uvedeném zdrojovém kódu (viz Zdrojový kód 5) je možné vidět filtrování entity *Picture*, jako kritérium vyhledávání je dán profil a typ linkovaného obrázku k onomu profilu. Oba tyto parametry jsou nastaveny jako filtrovací podmínky pro vazební entitu reprezentující link mezi profilem a obrázkem. Tento způsob vyhledávání záznamů je velmi jednoduchý a při znalosti databázového modelu i velmi účinný, jelikož se vzhledem k vazbám v modelu dá filtrovat prakticky napříč celým modelem.

```
public Picture GetPicture( IProfile profile, ProfilePictureType type
)
{
   PictureFilter filter = new PictureFilter();
   filter.PictureLinkFilter.Profiles.Add( profile );
   filter.PictureLinkFilter.Types.Add( type );
   filter.IsDeleted = false;
   return profile.GetInstanceModule<PictureHandler>( ).One( filter );
}
```

Zdrojový kód 5 – Získání entity za použití filtru

Pro filtrování entit je také možné využít implementace LINQ jazyka (viz Zdrojový kód 7) jež je také součástí frameworku, a je s ním obeznámen prakticky každý vývojář pracující s jazykem C# nebo Entity frameworkem.

Díky tomu je pro nové zaměstnance firmy velice snadný přechod mezi jinými frameworky a proprietárním řešením partnerské firmy.

```
IQueryable<Picture> list = from picture in instance.Pictures
  join link in instance.ProfilePicsLinks on picture
      equals link.Picture
   where  link.Profile == user
   select picture;

IQueryable<Picture> list1 = instance.Pictures.Join(
instance.ProfilePicsLinks, picture => picture, link => link.Picture,
    ( picture, link ) => new { link, picture } )
   .Where( join => join.link.Profile == user )
   .Select( join => join.picture );
```

Zdrojový kód 7 - Ukázka použití LINQ pro získávání záznamů z databáze

V případech, kdy je nutné získat data napříč několika entitami a sekvenční dotazování databáze by bylo příliš pomalé, je možné vytvoření tzv. *DataReport*. Jedná se o způsob jak v daném frameworku zapsat komplexní databázový dotaz, ze kterého je poté generována ekvivalentní query. Typicky je tato metoda pro získání dat použita na místech, u kterých je požadavek na extrémní efektivitu a rychlost. Na ukázce níže (viz Zdrojový kód 6) je možné vidět ukázku reportu, jenž slouží k získání kolekce dnů, které mají nějaký časový záznam v databázi, který nemá status *Completed.*

```
public class PendingDaysToBilledReport
   : FilteredDataReport<EventTimeFilter,
PendingDaysToBilledReportResult>
{
   protected override void SelectData( )
   {
      base.SelectData( );
      this.Select( EventTimeFields.StartsOn );
      this.Dataset.FilterValue( EventTimeFields.Status,
         EventTimeStatus.Completed, DataOperator.NotEqual );
      this.Dataset.FilterValue( EventTimeFields.SystemCode,
         EventTimeSystemCode.WorkTimeRecord.GetEnumValueUnsigned( ) );
      this.GroupBy( EventTimeFields.StartsOn );
   }
}
```

Zdrojový kód 6 – Ukázka vytvoření DataReportu

4.3.2 Multi-instanční aplikace

Jedním z požadavků na budovanou aplikaci bylo to, aby systém umožňoval současný běh více instancí – v kontextu budované aplikace se jedná o firmy, jež systém budou používat. Tato možnost je již zabudována v základní logice frameworku tím, že každá entita má jako součást primárního klíče identifikátor instance (*picture_oid,* který je možné vidět v entitě *picture* na Obrázek 9), do které náleží.

Obrázek 9 – Ukázka kombinovaného primárního klíče entity

Konfigurace volení instance je součástí souboru *Global.asax*, a to jak u aplikačního rozhraní, tak u webové aplikace. Webová aplikace je nakonfigurována tak, aby volila instanci na základě jejího aliasu, jenž je součástí URL (viz Obrázek 10). Tento alias je unikátní a je vytvořen při vytváření nové instance. Díky tomuto mechanismu (viz Zdrojový kód 8) je zaručeno, že firmy budou schopny prohlížet jen svá vlastní data, jelikož framework ručí za přístup k datům pouze v aktuální instanci.

https://www.companeero.com/p/dev-company/dashboard/projects

Obrázek 10 – Příklad aliasu instance v URL adrese

```
if ( this._Instance == null && InstanceUrlOption.IsOneOf(
InstanceUrlOption.Path ) )
{
  Uri uri = CurrentHttpContext?.Request?.Url;
  if ( uri != null )
  {
    uint tokenPosition = InstanceTokenPosition;
    string parentPath = uri.GetParentPath( tokenPosition );

    if ( parentPath.EndsWith( InstanceUrlParentPath ) )
    {
      string instanceToken = uri.GetToken( tokenPosition );

      if ( Strings.NotEmpty( instanceToken ) )
      {
        this._Instance =
          this.GlobalInstance.GetInstance<TInstance>(
            instanceToken, InstanceResolverOption.Alias );
      }
    }
  }
}
```

Zdrojový kód 8 – Vytvoření instance na základě aliasu přítomného v URL

Aplikační rozhraní má nakonfigurovanou volbu instance pomocí povinné hlavičky v dotazu (viz Obrázek 11, Zdrojový kód 9). Hodnota této hlavičky je unikátní identifikátor instance ve formátu GUID.

Díky této konfiguraci obou aplikací byl splněn jeden z požadavků multi-instančnosti systému.

```
GET /v1/Projects HTTP/1.1
Host: api.companeero.com
Connection: keep-alive
Pragma: no-cache
Cache-Control: no-cache
Accept: application/json
X-InstanceOwnerKey: c5a52c1e-ef0f-ef0f-a2b9-188b2f4fb811
```

Obrázek 11 – Příklad hlavičky pro výběr instance v aplikačním rozhraní

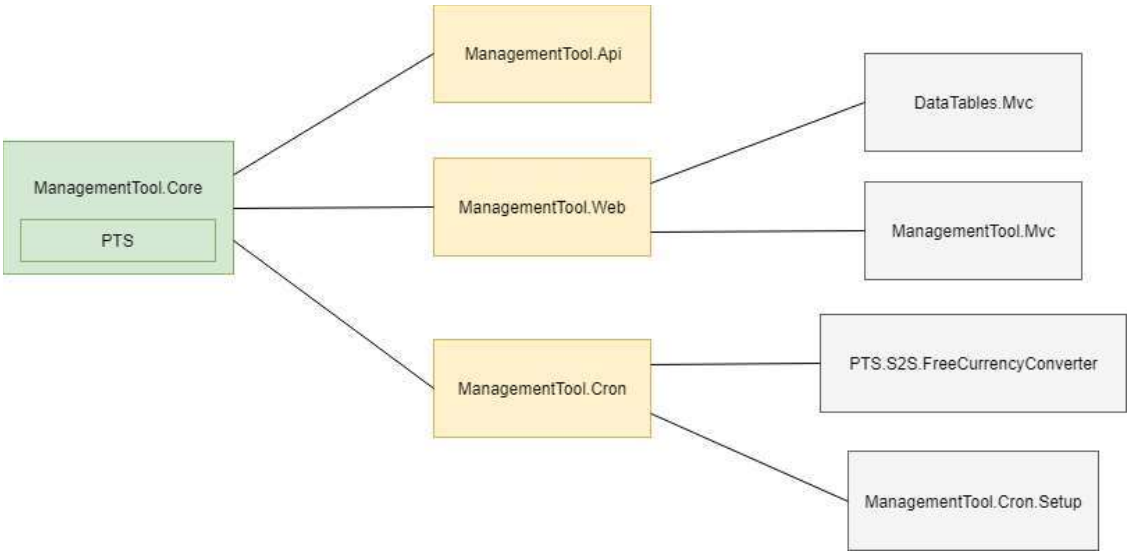
```
if ( MetaDataSource == WebContextMetaDataSource.HttpHeaders )
{
  string instanceId = CurrentHttpContext.GetHttpHeader(
HttpHeaderInstanceKey );

  if ( instanceId.NotEmpty( ) )
    this._Instance = this.GlobalInstance.GetInstance<TInstance>(
instanceId, InstanceResolverOption.Guid );
}
```

Zdrojový kód 9 – Vytvoření instance na základě HTTP hlavičky

5 Návrh řešení a implementace řešení

Architektura systému byla rozdělena do několika logických celků, které jsou navzájem pospojované dle jejich zaměření a funkcionality. Systém byl tak rozdělen do několika menších projektů, jejichž propojení je znázorněno na obrázku níže (viz Obrázek 12).

Stěžejním bodem celého systému je projekt *ManagementTool.Core,* jenž definuje celou business logiku aplikace a jenž je postaven nad frameworkem dodaným partnerskou firmou. Tento projekt je možné dále referencovat v jiných aplikacích.



Obrázek 12 – Diagram členění projektů

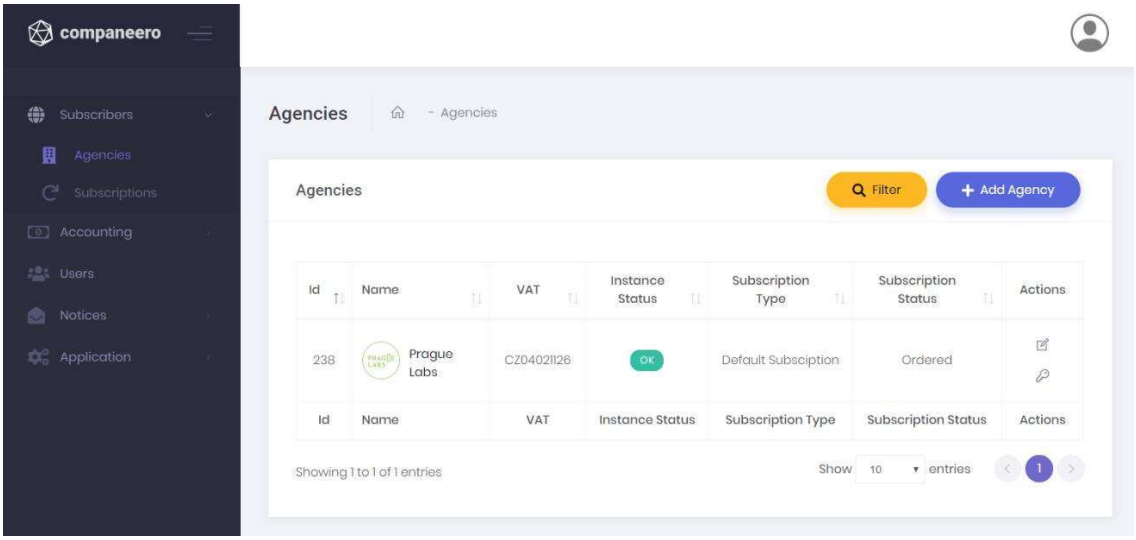
Další součástí systému jsou dvě samostatné aplikace *ManagementTool.Api* a *ManagementTool.Web*, z hlediska vícevrstvé architektury by tyto aplikace představovaly prezentační vrstvu. Další samostatnou aplikací je projekt *ManagementTool.Cron,* jenž slouží jako Windows služba, která koná asynchronní a plánované operace. Další nedílnou součástí systému jsou podpůrné projekty, které těmto uvedeným aplikacím dodávají další funkcionalitu nebo některou funkcionalitu separují tak, aby mohla být později použita v jiných systémech.

5.1 Systémová administrace

Systémová administrace je část aplikace, která slouží správcům produktu k nastavení aplikace, správě instancí, administrátorů a dalších věcí, jež budou podrobněji představeny v této kapitole. Tato část, stejně jako sekce pro přístup pro firmy samotné, je součástí projektu *ManagementTool.Web*, který je postaven na frameworku ASP .NET MVC. Pro vytvoření uživatelského rozhraní bylo použito HTML5 a CSS3, pro usnadnění vytváření rozhraní se vychází ze zakoupené šablony Metronic[10], která je postavena na CSS frameworku Bootstrap a byla zakoupena klientskou firmou. Použitím šablony se usnadnilo designování klientské části aplikace, jelikož se vycházelo z již předdefinovaných komponent. Samozřejmostí byla také responzivita celé aplikace díky grid systému Boostrap frameworku.

5.1.1 Správa instancí

Správa instancí (firem používajících aplikaci) je klíčovým bodem systémové administrace (viz Obrázek 13), jelikož slouží k vytváření nových firem a ovládá jejich přístup do systému.



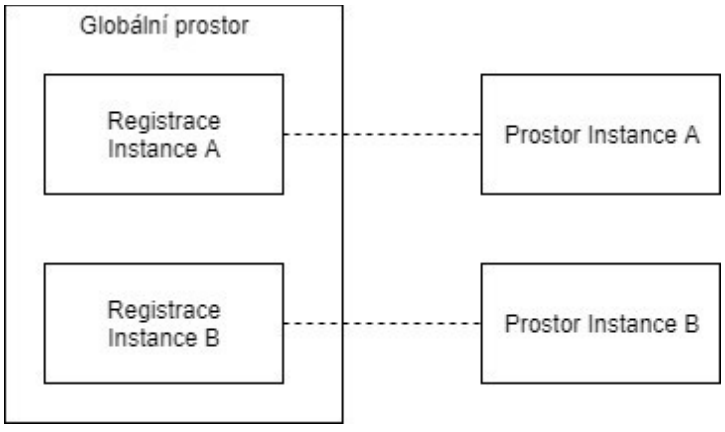
Obrázek 13 – Náhled na přehled firem registrovaných do aplikace

Jak již bylo zmíněno, multiinstančnost systému je již podporována v dodaném frameworku. Proces vytváření nové instance spočívá ve vytvoření

nového záznamu do tabulky *instanceowner* a ve vytvoření souvisejících meta informací v ostatních tabulkách, které jsou buď ve vztahu dědičnosti, nebo jsou připojené jinou vazbou.

Obrázek 14 – Ukázka entity představující instanci

Data (firmy, předplatné, notifikace,…), která jsou spravována přes systémovou administraci, se nacházejí v tzv. globálním prostoru, za který je považována instance, jejíž identifikátor je roven nule (viz Obrázek 15). Data náležící instanci samotné se nacházejí pod jejím identifikátorem a jsou oddělena od jiných instancí. Framework také disponuje funkcí, jež umožňuje ukládání dat instance na jiném MySQL serveru, jelikož každá instance má možnost vytvoření jiného databázového připojení, než který je použit ostatními, a umožnuje tak systém škálovat.



Obrázek 15 – Náhled na rozložení dat

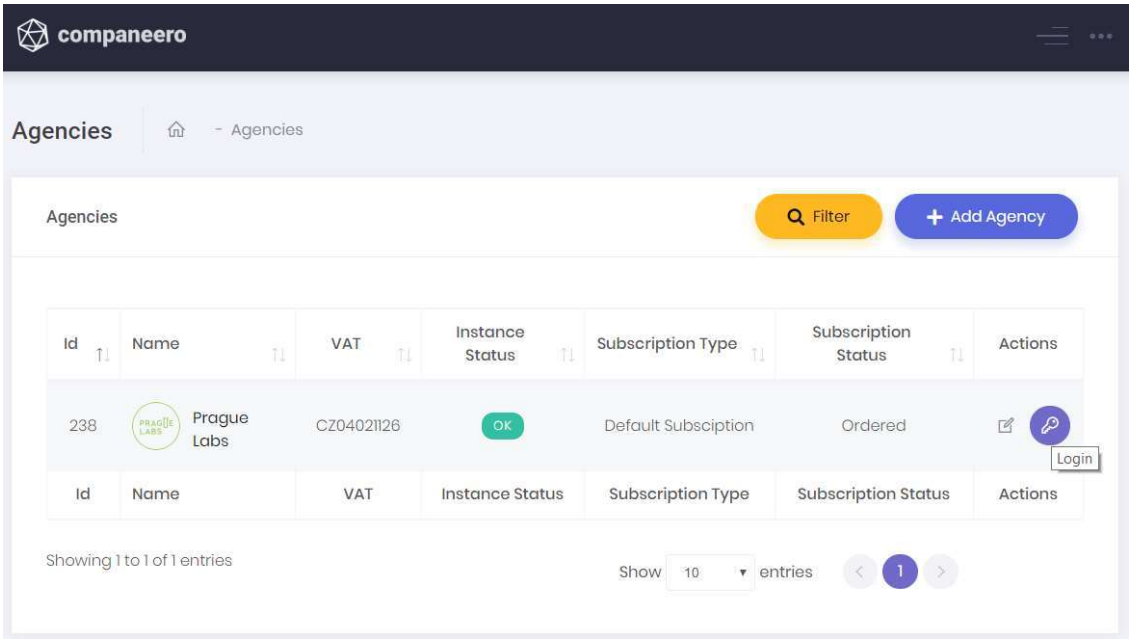
Pro každou instanci (potažmo firmu) jsou v systému uloženy informace, které upravují aplikaci pro použití firmou. Například se jedná o jazyk, výchozí

měnu, loga použitá v aplikaci a také například alias firmy, jenž je součástí URL adresy, na které firma aplikaci používá.

5.1.2 Správa administrátorů

Další funkcionalitou systémové administrace je správa uživatelských účtů administrátorů. Díky této sekci je možné vytvořit nové uživatelské účty, jež jsou oprávněny ke správě instancí a přístupu do nich.

Administrátoři také mají přístup do jednotlivých instancí, jež umožňuje kontrolu v případě nefunkčnosti či jiného zásahu uživatelské podpory. Tato funckionalita však bude pravděpodobně v případě komerčního užití aplikace odstraněna vzhledem k možnému narušení soukromí firem. Přístup administrátorů do instance není vytvářen automaticky, ale je vytvořen až v případě kliknutí na tlačítko *Login* dané instance (viz Obrázek 16). Tyto uživatelské účty jsou napříč instancemi párovány pomocí stejného GUID identifikátoru. Při každém přihlášení tak dochází buď k vytvoření nového účtu, anebo aktualizaci toho existujícího. Administrátorské účty mají nastaveny jiné role než uživatelé aplikace, tudíž se například neobjeví ve výpisu uživatelů dané instance.

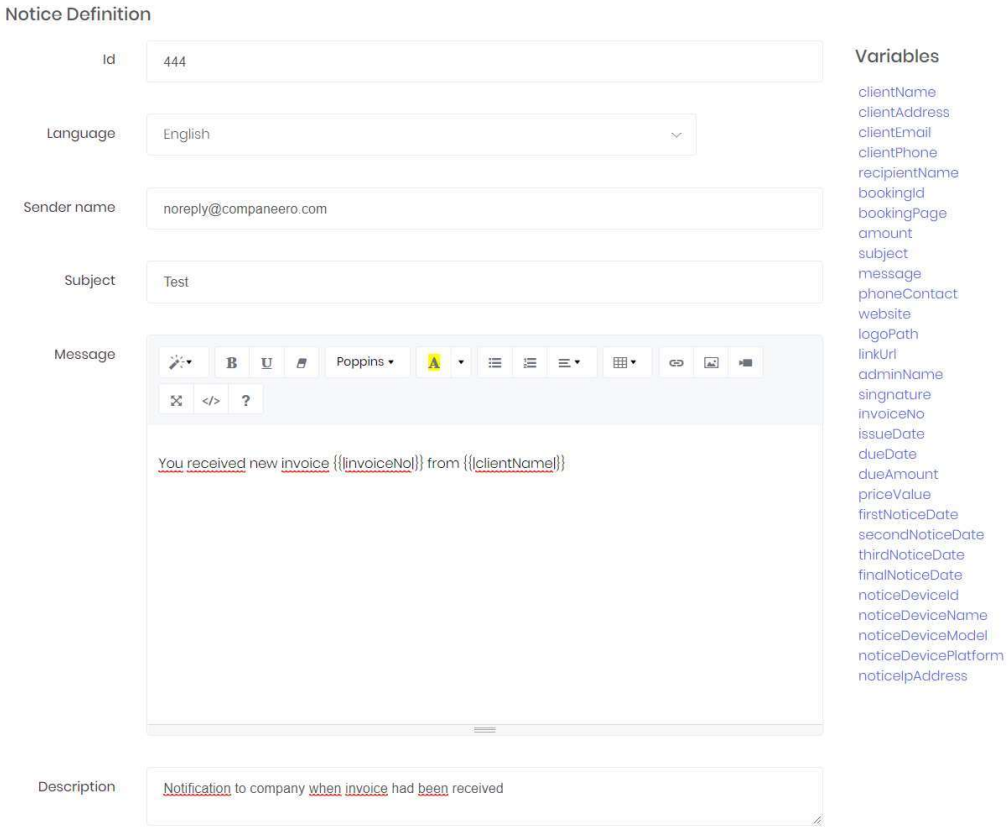


Obrázek 16 – Přihlášení do instance

5.1.3 Připravené funkce

Systémová administrace také umožňuje správu předplatného firmy, která slouží k udělení oprávnění přístupu do aplikace pro danou firmu. V současné verzi však nejsou definované žádné typy předplatného, jejich ceník či omezení funkcionality aplikace. Dalšími funkcemi, které jsou připraveny, jsou například výpisy transakcí a faktur, jež budou sloužit k přehledu plateb za předplatné.

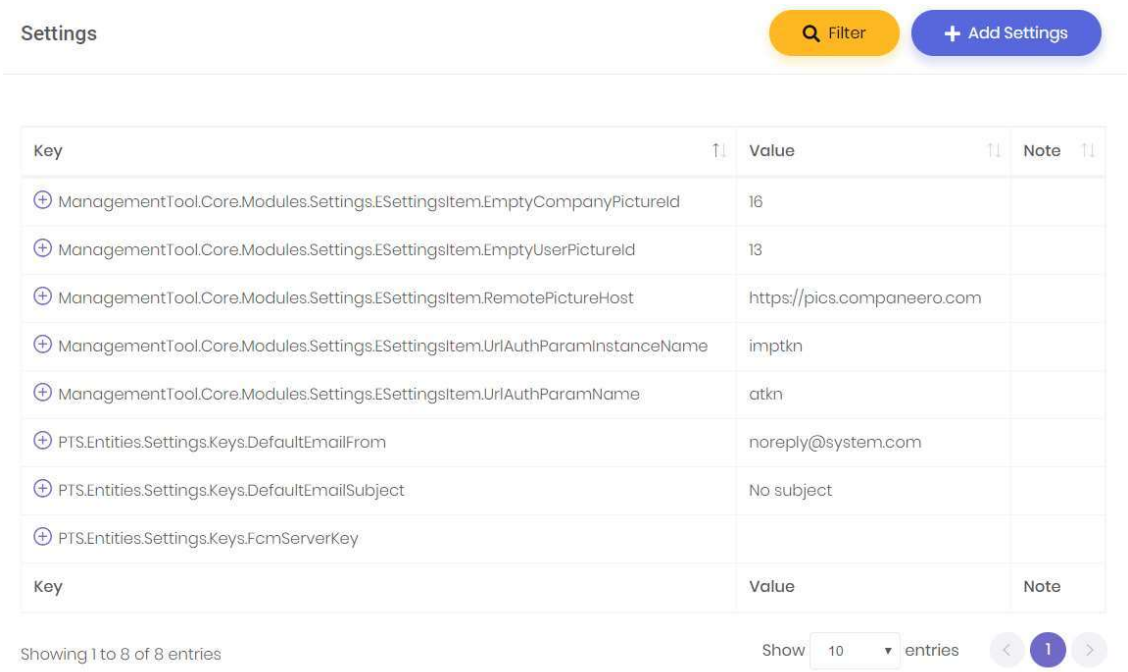
Další přípravou pro komerční provoz aplikace je definice šablon notifikací (viz Obrázek 17), které systém bude schopný generovat. Tyto šablony mají statické identifikátory definované uživatelem pro jejich snadnější použití a referencování v kódu aplikace. Šablony mohou obsahovat proměnné, jež je poté možné vyplnit relevantními informacemi.



Obrázek 17 – Definice šablony notifikace

Aplikace také nabízí detailní přehled vygenerovaných notifikací, který může sloužit ke kontrole korektního nastavení pravidel pro jejich odeslání. Aplikace také umožňuje notifikaci znovu zařadit do fronty k odeslání, která je zpracována asynchronně (viz 5.5.2). Systém v současné době negeneruje žádné notifikace, nicméně aplikace je připravena na rozesílání e-mailových notifikací pomocí služby Amazon Simple E-mail Service.

Systémová administrace také obsahuje systémové nastavení nabízející dodatečnou konfiguraci aplikace. Toto nastavení je obdobou konfigurace, která by mohla například být umístěna do souboru *web.config,* jenž se standardně používá pro MVC aplikace; ovšem konfigurace nastavení přímo v aplikaci je pohodlnější a může být měněna za běhu aplikace. Toto nastavení v současné době například obsahuje adresu obrázkového serveru či názvy autentizačních parametrů v URL adrese.



Obrázek 18 – Systémové nastavení

Další funkcionalitou systémové administrace je definice dynamických textů, které slouží k překladu aplikace do dalších jazyků. Je tedy možné

všechny řetězce z aplikace přepsat uložit je do databáze a poté je referencovat ve views (pohledech, viz Zdrojový kód 10).

```
<li class="m-menu__item  m-menu__item--submenu">
  <a href="#" class="m-menu__link m-menu__toggle">
    <i class="m-menu__link-icon fas fa-users"></i>
    <span class="m-menu__link-text">@Html.DynamicText(345)</span>
  </a>
</li>
```

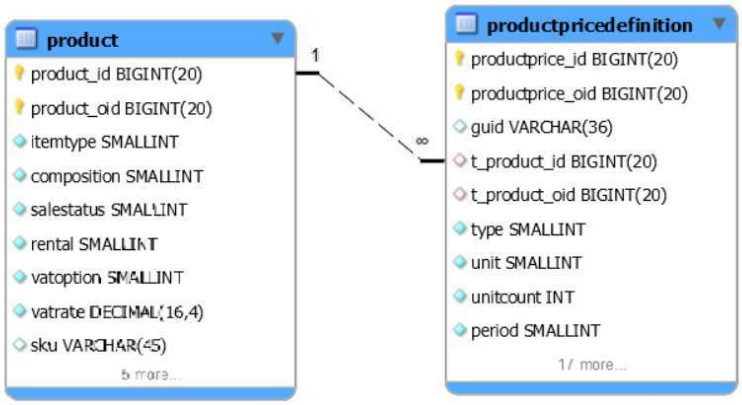
Zdrojový kód 10 – Ukázka použití dynamického textu ve view

5.2 Dashboard firmy

Tato část aplikace slouží k přístupu samotnými firmami, je středobodem celého systému a obsahuje veškerou funkcionalitu pro hlídání ziskovosti projektů a jejich nákladů v podobě platů zaměstnanců. Jednotlivé části aplikace jsou popsány v následujících kapitolách a obsahují architekturu, která dané funkce realizuje na úrovni databázového modelu. Databázový model, včetně jeho úprav, je možné nalézt na přiloženém médiu se zvýrazněnými částmi, jež byly do frameworku přidány nebo byly opraveny.

5.2.1 Služby

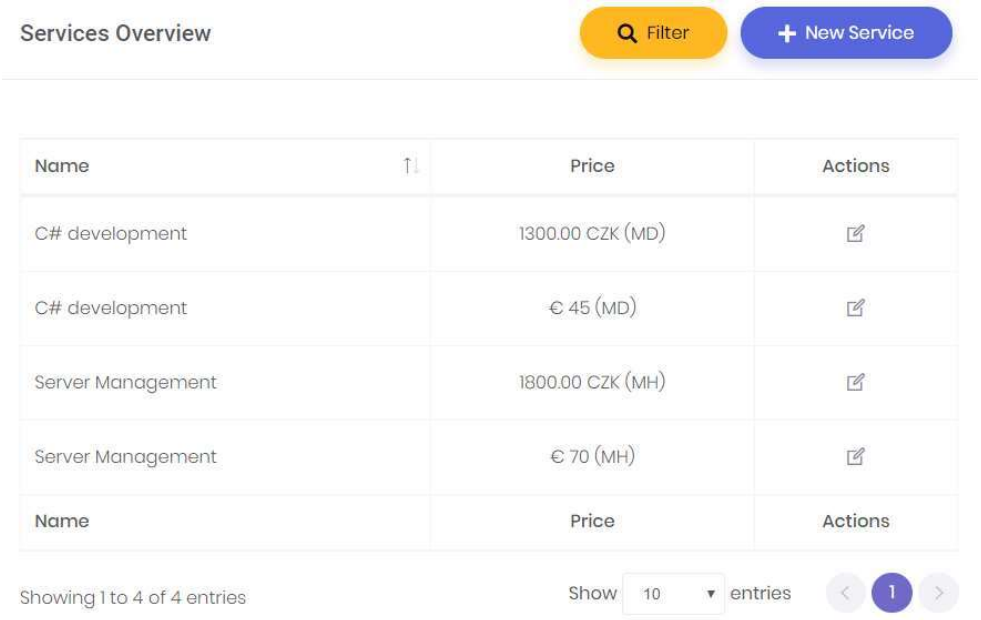
Agentura může v aplikaci definovat ceník služeb, jenž nabízí (viz Obrázek 20), tyto služby pak mohou být později použity jako předloha pro vytvoření služby/práce v rámci projektu. Každá služba je v rámci databázového modelu (viz Obrázek 19) realizována jako entita *product* a je k ní pomocí připojené entity *productpricedefinition* stanovena cena.



Obrázek 19 – Náhled na databázový model realizující definici služby

Každá služba tak umožňuje definování následujících vlastností:

- název,
- popis,
- platnost – časový rozsah,
- typ účtování – fixní, man day (1 den / 8 hodin práce), man hour (hodina práce),
- měna ve které je cena stanovena,
- cena.

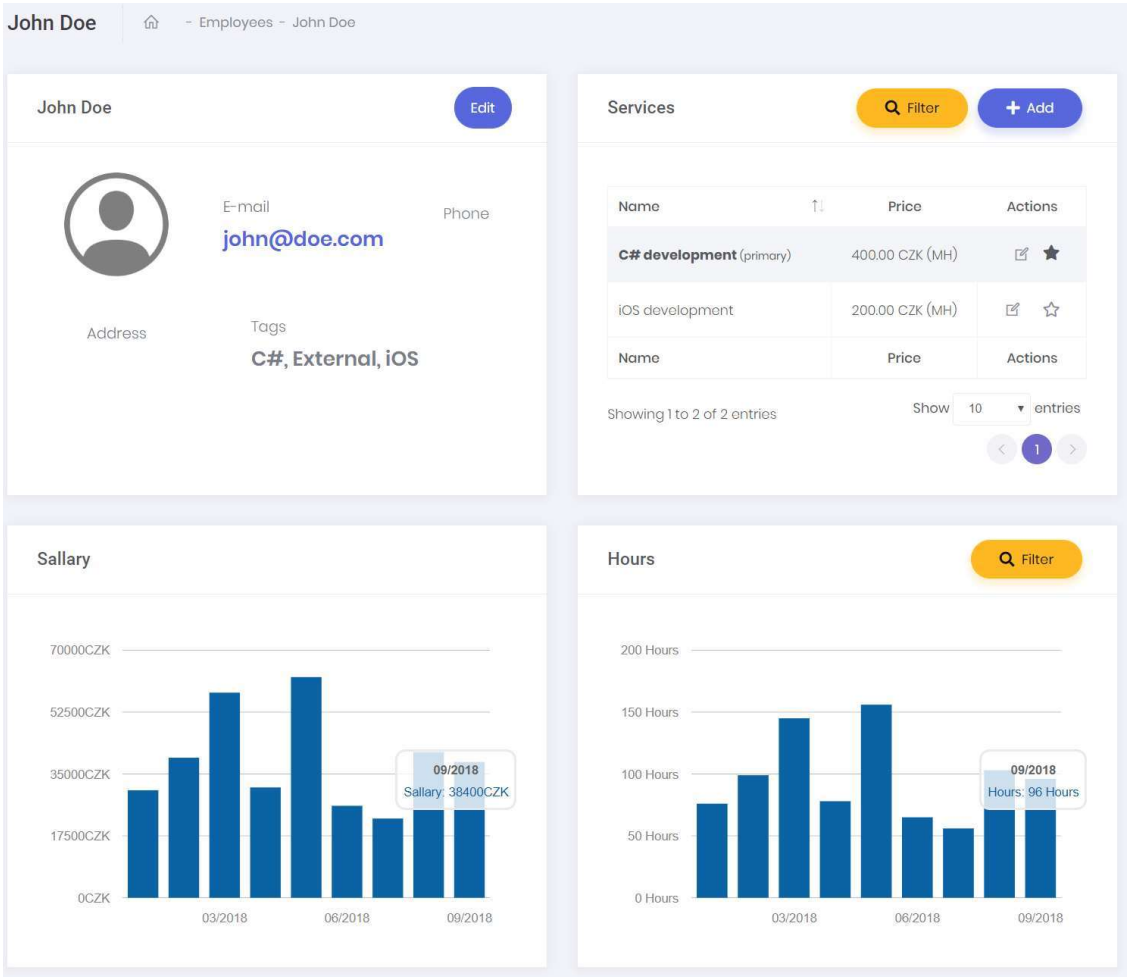


Obrázek 20 – Náhled přehledu služeb agentury

5.2.2 Zaměstnanci

Aplikace eviduje registr zaměstnanců či kontraktorů dané firmy. Ke každému zaměstnanci je možné uložit základní informace. Klíčovou funkcí je definice služeb zaměstnance, které jsou realizovány identicky jako služby agentury. Každý zaměstnanec tak má možnost definovat jednu či více služeb, jež agentuře nabízí. Z hlediska zaměstnanců se bude jednat o jednu službu s fixní sazbou, ale pro kontraktory je tato funkce velkým přínosem, jelikož je možné definovat různé sazby na různé typy prací. To v reálném světě například odráží to, že jeden kontraktor může být senior programátor na jedné platformě, ale není tak zkušený s jinou platformou, což se projeví na jeho sazbě.

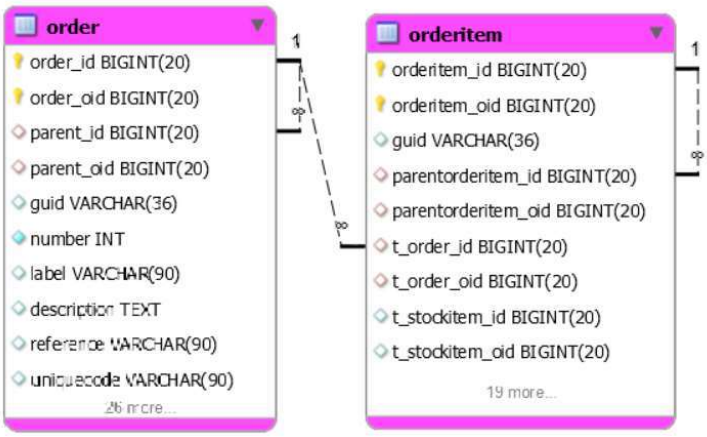
Na následujícím screenu (viz Obrázek 21) je možné vidět náhled části stránky detailu zaměstnance, ten nabízí rychlý přehled toho, jaké služby nabízí a kolik odvedl práce (kvantifikace počtu hodin), což se odrazí na jeho výplatě. Dále je na této stránce detailní výpis toho, na čem pracoval (výpis ze systému měření času) a výpis příslušných transakcí. Následně se na stránce také nachází výpis faktur, jež v této aplikaci slouží jako report.



Obrázek 21 – Náhled na detail zaměstnance

5.2.3 Projekty

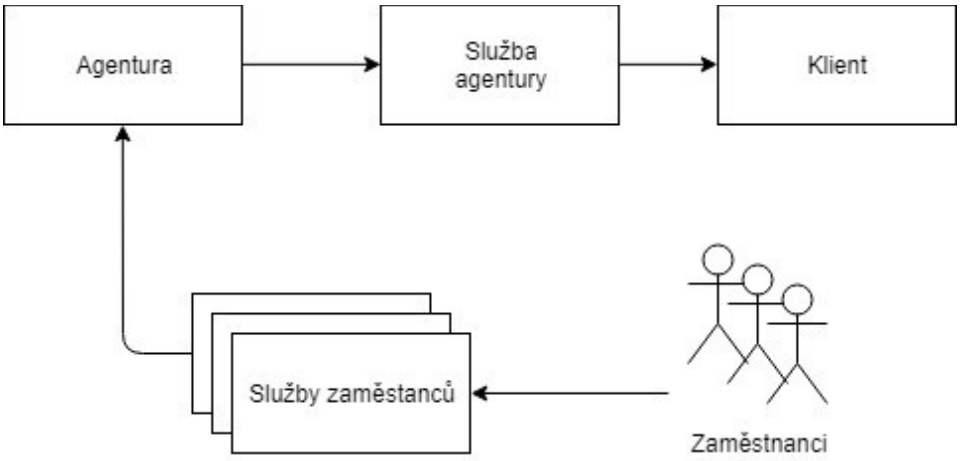
Stěžejním bodem a přínosem celé aplikace je evidence projektů. Každý projekt je přiřazen k určitému klientovi, jež si jej objednal. Funkcionalita projektů byla na databázovém frameworku abstrahována na existující entity realizující objednávky (viz Obrázek 22), jelikož projekt je v podstatě objednávka služeb agentury klientem. Ke každému takovému projektu (objednávce) je možné definovat objednané služby – položka objednávky (viz 5.2.1).



Obrázek 22 – Zjednodušený náhled na databázový reprezentující projekty

Koncept

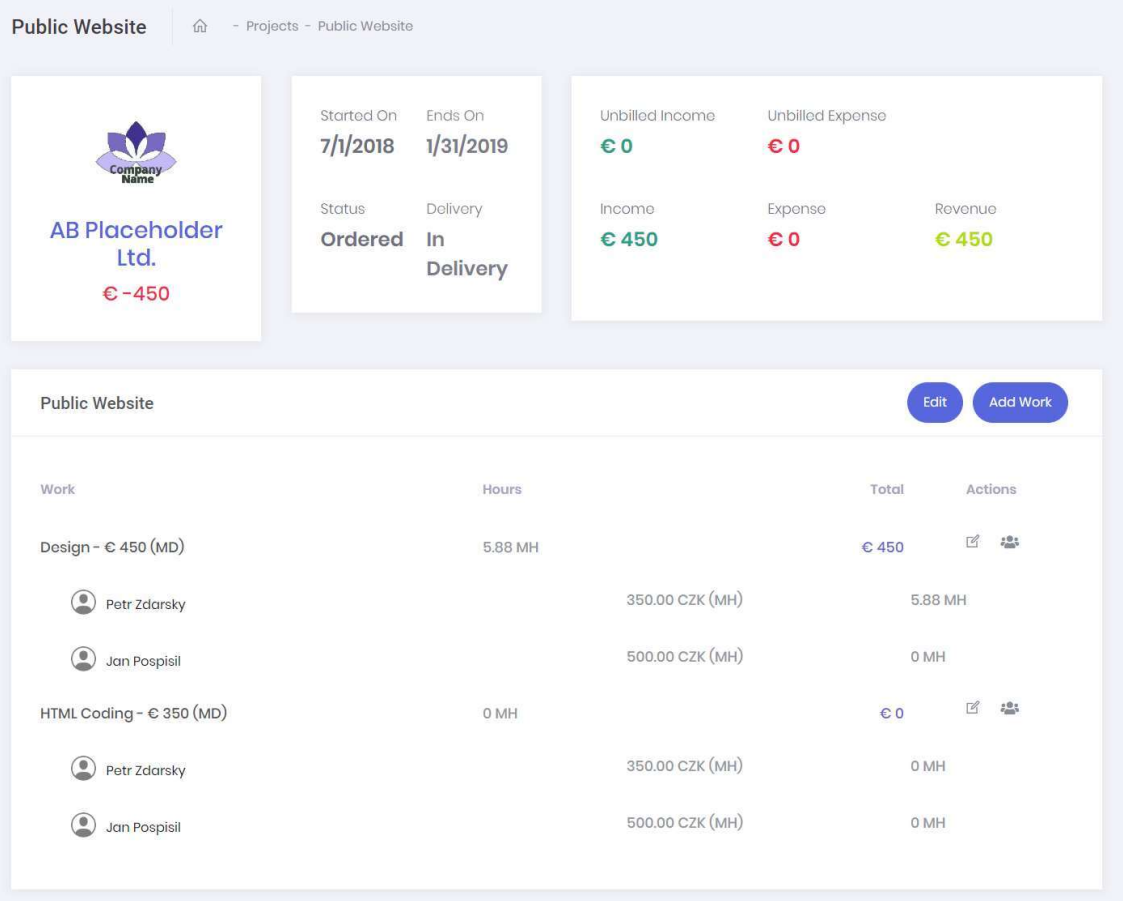
Projekty a hlídání jejich nákladů jsou klíčovou funckionalitu celé aplikace, od níž se odvíjejí další možnosti systému. Koncept, který byl pro jejich realizaci zvolen, je přeprodej služeb zaměstnanců od agentury klientovi. To v praxi znamená, že agentura objednává služby od svých zaměstnanců a práci těchto zaměstnanců dodává klientovi jako svou službu (viz Obrázek 23). Kompletní náhled architektury včetně měření času a dalších souvisejících věcí je možné nalézt na přiloženém médiu a v Příloha C.



Obrázek 23 – Diagram konceptu přeprodeje služeb

Celý systém je tak rozdělen na dvě části, jež jsou z větší části identické, jelikož jsou postaveny na objednávce služeb: agentura ➔ zaměstnanec, klient ➔ agentura.

Pomocí dalších vazeb poté dochází k propojení objednaných služeb od zaměstnance ke službám prodávaných klientovi; to umožňuje sledovat náklad a zisk na jednotlivých službách, jelikož je zřejmé, kolik stojí hodina práce zaměstnance, jež je přeprodávána klientovi. Na obrázku níže (Obrázek 24) je vidět náhled detailu projektu s jeho objednanými službami, ke kterým jsou přiřazeny služby zaměstnanců.

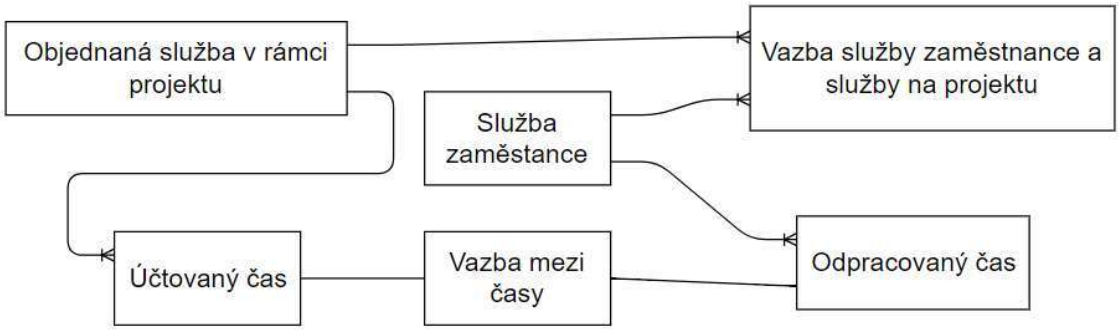


Obrázek 24 – Náhled na detail projektu

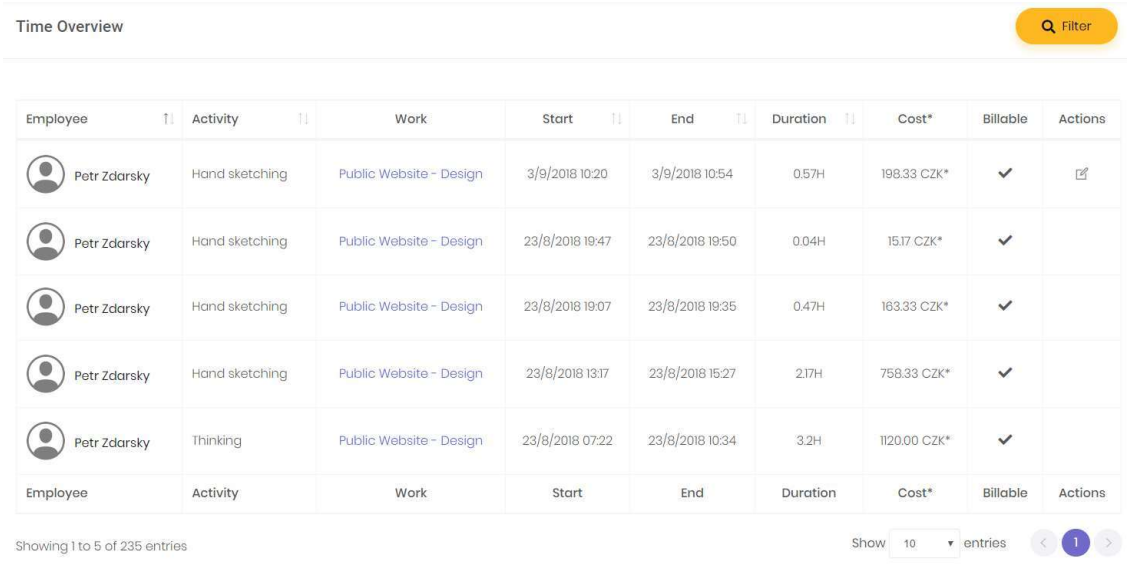
Měření času

Nezbytnou funkcionalitou aplikace je možnost evidování času práce odvedené zaměstnanci, jež je poté účtována klientovi. Tato funckionalita je v současné době realizována pomocí dat, která jsou do systému synchronizována z aplikací třetích stran (viz 6.1). Systém funguje zrcadlově a ve skutečnosti tak jsou pro jednu aktivitu vytvořeny dva časové záznamy: jeden je proplacen zaměstnanci, druhý je účtován klientovi. Systém funguje

tak, že tyto dva záznamy jsou automaticky synchronizovány při jejich úpravě. Aktualizace záznamů může být přerušena tím, že dojde k vyúčtování jednoho z časů nebo manuální editací v systému. Manuální editace tak umožnuje například změnit délku času, jenž je klientovi účtován, případně tento čas nastavit jako neúčtovatelný klientovi. Oba časy jsou vázány k příslušným službám (viz Obrázek 25), je tak možné získat informaci o tom, kolik hodin zaměstnanec poskytl na dané službě nebo kolik hodin bylo odvedeno na objednané službě klientovi.



Obrázek 25 – Náhled na vazby určené pro měření času

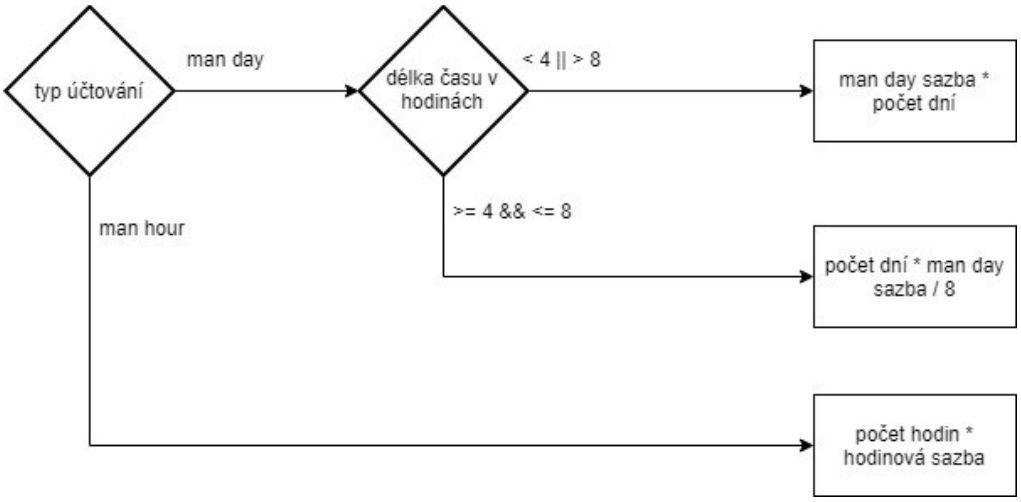


Obrázek 26 – Náhled na přehled aktivit (měřeného času)

Účtování

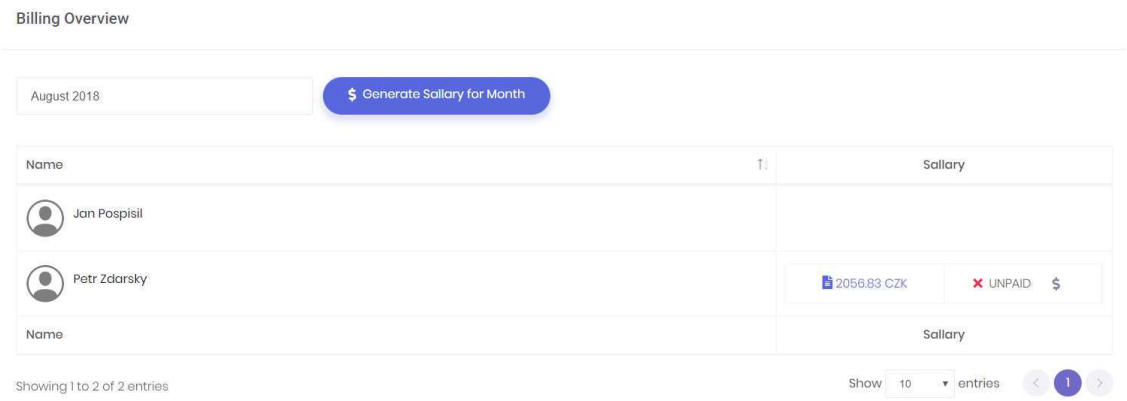
Jak již bylo zmíněno v předchozí kapitole, účtování času je rozděleno do dvou částí, a to účtování času klientovi a výplata odvedené práce

zaměstnancem. Tyto části jsou na sobě nezávislé, avšak vycházejí ze stejných pravidel (viz Obrázek 27). Veškerá aktivita je agregována do dnů a služeb, výsledná suma je poté počítána na základě celkového času za daný den a typu účtování služby.



Obrázek 27 – Diagram pravidel použitých pro účtování času

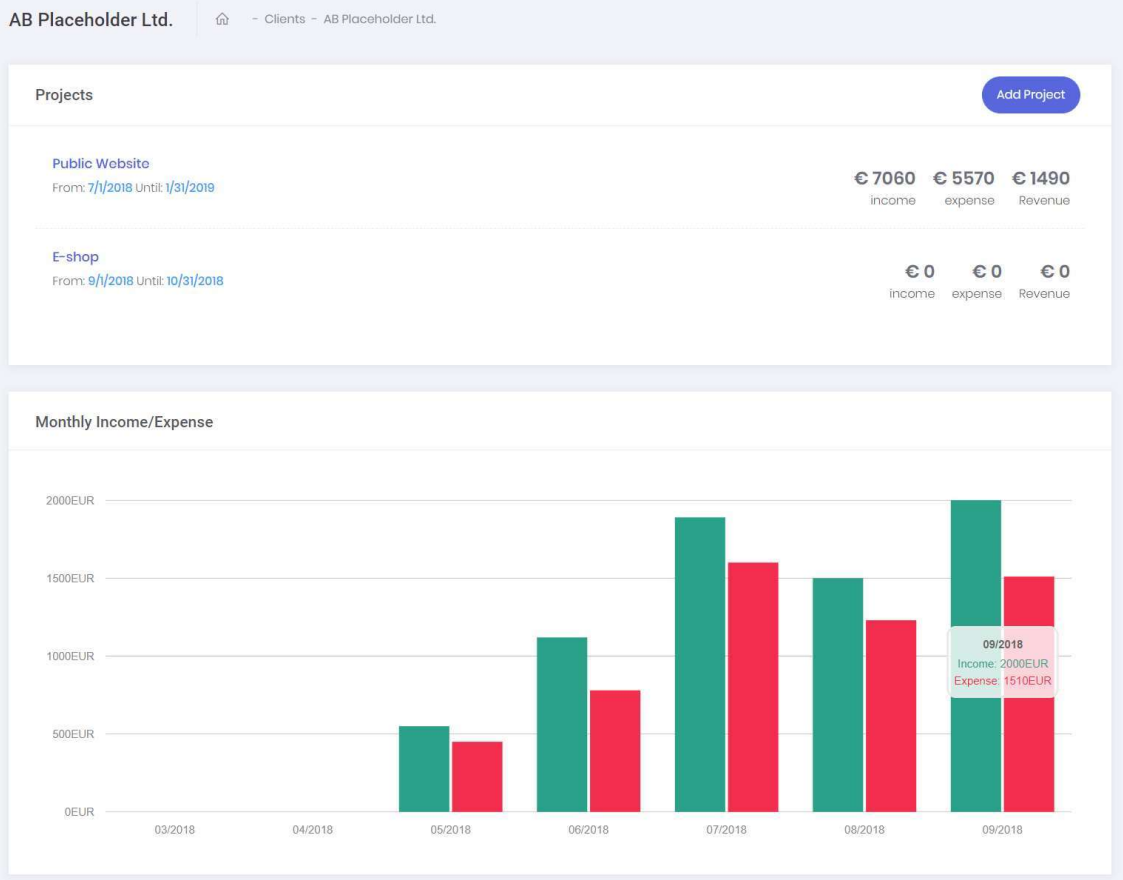
Tyto vygenerované transakce jsou poté umístěny do faktury, která v tomto systému momentálně slouží jako výkaz účtované práce. Aplikace tak disponuje přehledem faktur, které je poté možné v rozhraní nastavit jako proplacené. Na obrázku níže (viz Obrázek 28) je možné vidět rozhraní pro přehled výplat zaměstnanců, rozhraní pro platbu od klienta vypadá obdobně.



Obrázek 28 – Výpis částek k proplacení pro zaměstnance

5.2.4 Klienti

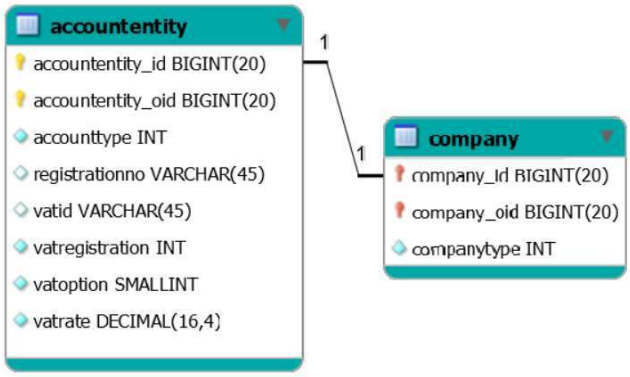
Aplikace umožnuje vytvářet databázi klientů agentury, o nichž jsou ukládána základní data včetně účetních informací. Pro každého klienta je automaticky vytvořen peněžní účet v měně, která klientovi byla nastavená. Zvolení měny závisí na tom, v jaké měně se klientovi fakturuje dodaná práce.



Obrázek 29 -Náhled na detail klienta

Na obrázku výše (viz Obrázek 29) je možné vidět část detailu klienta. Tato zobrazená část umožňuje získání rychlého přehledu o tom, jak si který projekt finančně stojí a jaké jsou příjmy či výdaje spojené s tímto klientem. Dalším obsahem této stránky je také přehled transakcí a souvisejících faktur.

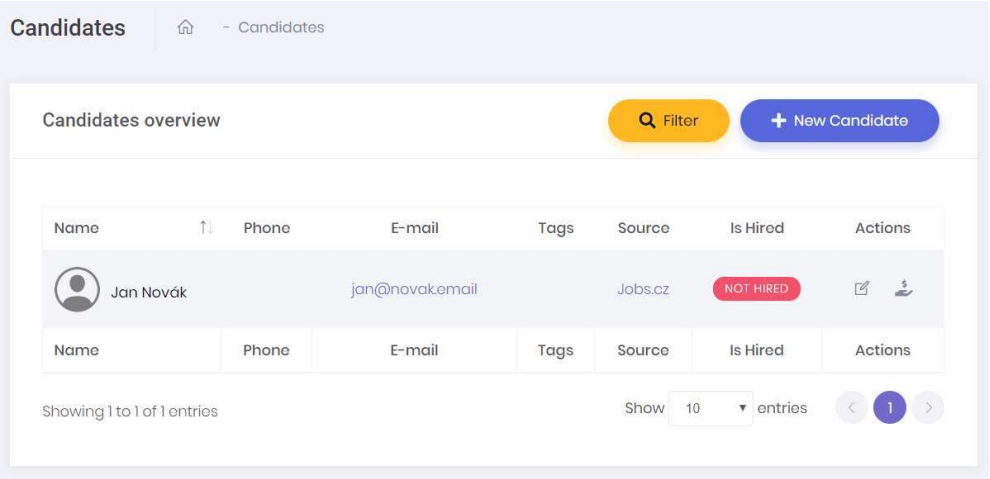
Klienti jsou v databázi reprezentováni entitou *company* (viz Obrázek 30jež je vázána dědičností ve frameworku na entitu *accountentity,* která umožňuje uložit náležité účetní informace a také související informace pomocí dalších záznamů v jiných tabulkách.



Obrázek 30 – Náhled na databázový model reprezentující klienta

5.2.5 Kandidáti

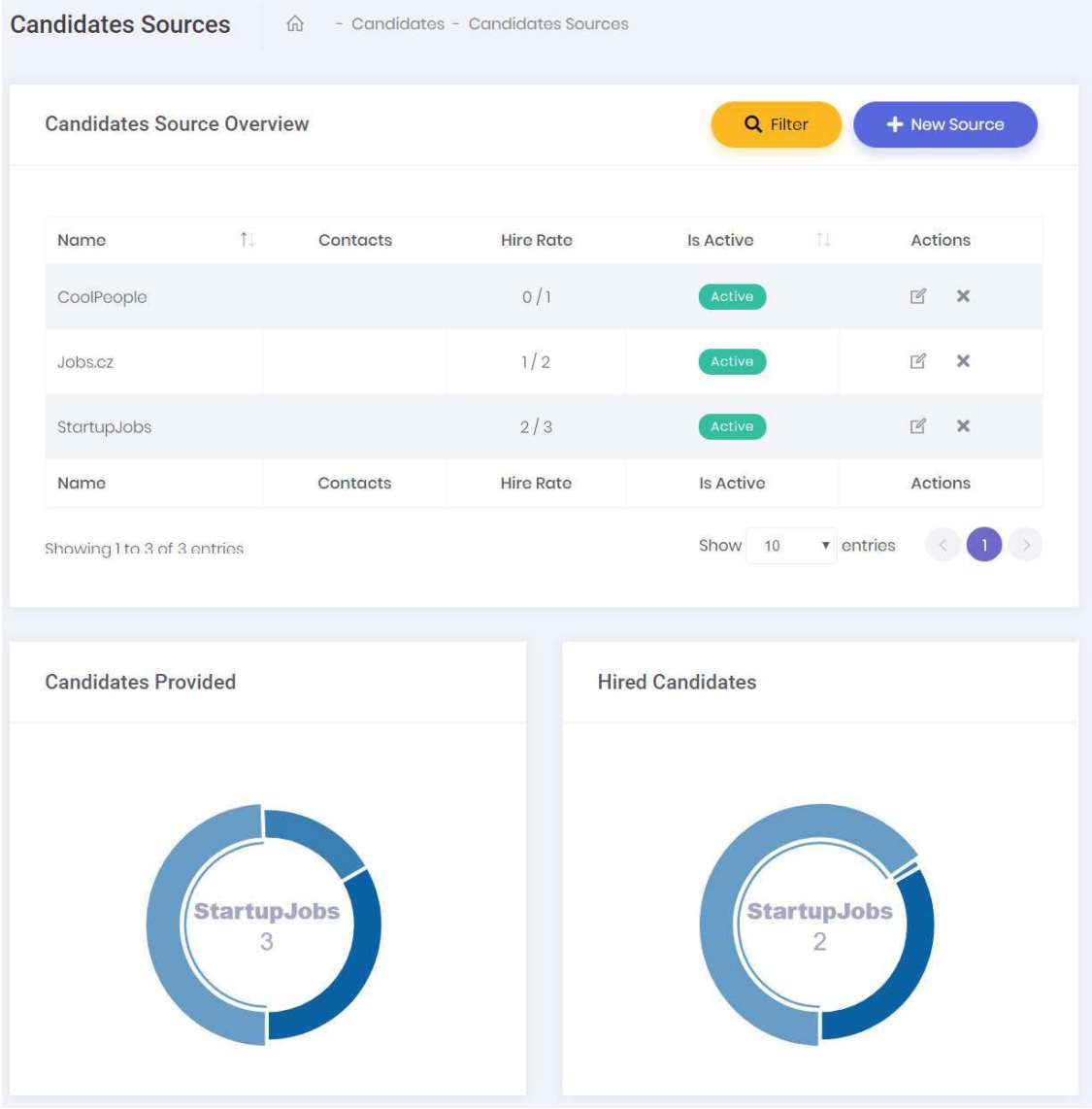
Aplikace umožňuje vytvářet databázi uchazečů o zaměstnání (viz Obrázek 31). Ke každému uchazeči je možné si definovat tagy. Tyto tagy lze také použit ke klasifikaci zaměstnanců, jelikož se dá říct, že zaměstnanec je další evoluční stupeň kandidáta. Ke každému kandidátovi je možné uložit základní informace (jméno, kontakt, adresa, …). Ke kandidátovi je také možné ukládat poznámky, které slouží k doplnění finančního ohodnocení, časové dostupnosti či jiných informací. Po úspěšném přijímacím pohovoru je z kandidáta možné jedním kliknutím vytvořit zaměstnance a posléze k němu doplnit chybějící informace.



Obrázek 31 – Náhled na přehled kandidátů

Aplikace nabízí u každého kandidáta k uložení informaci, přes který zdroj (HR agentura či portál nabízející práci) přišel. To umožňuje získávání

zajímavého přehledu (viz Obrázek 32), který zdroj poskytuje kvalitní uchazeče a který naopak ne.



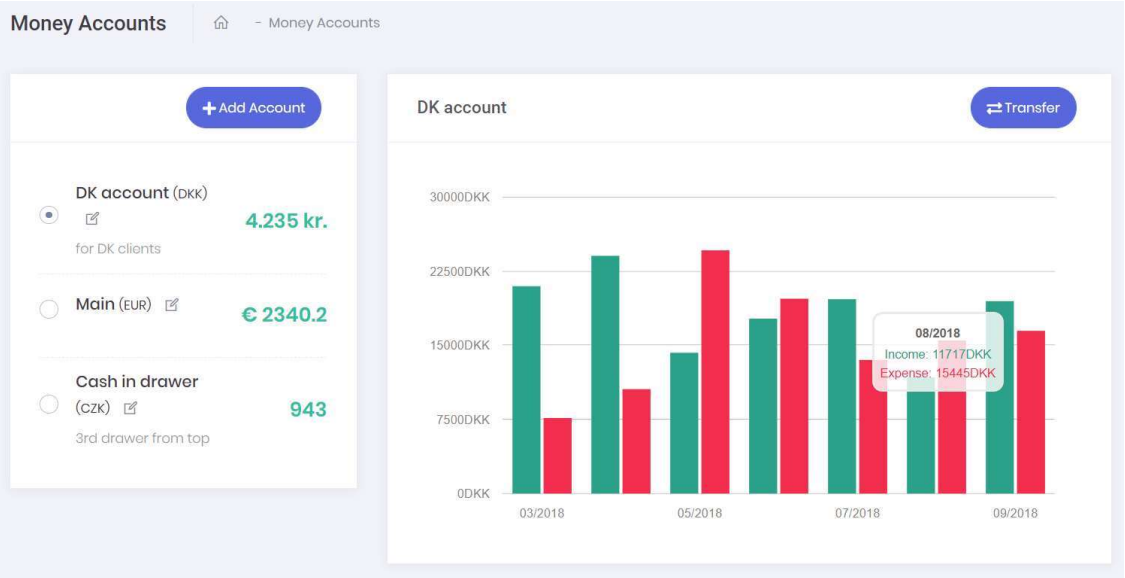
Obrázek 32 – Náhled na přehled zdrojů kandidátů

5.2.6 Peněžní účty

Virtuální peněžní účty udržují finanční balanc firmy, jenž se odvíjí od projektů a nákladů. Tyto účty jsou vytvářeny automaticky pro každou měnu, ve které firma obchoduje s klientem. Aplikace také umožnuje vytvářet tyto účty manuálně, což může agentuře pomoci s hlídáním zůstatku na jiných účtech. Tyto účty tak například mohou reprezentovat i „pokladnu“, tedy

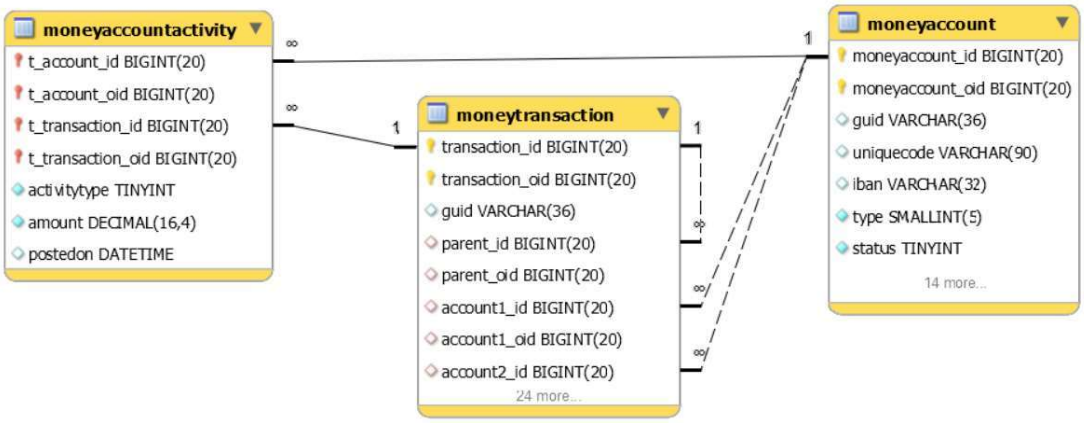
hotovost, kterou firma disponuje. Aplikace nabízí jednoduchý přehled všech účtů a jejich náhled na měsíční odchozí a příchozí platby (viz Obrázek 33).

Systém také umožňuje převádět peníze na jiný virtuální účet v aplikaci. Pokud má cílový účet nastaven jinou měnu, dochází ke konverzi částky a uživatel je informován o výsledné částce a převodním kurzu.



Obrázek 33 – Náhled přehledu stavů peněžních účtů

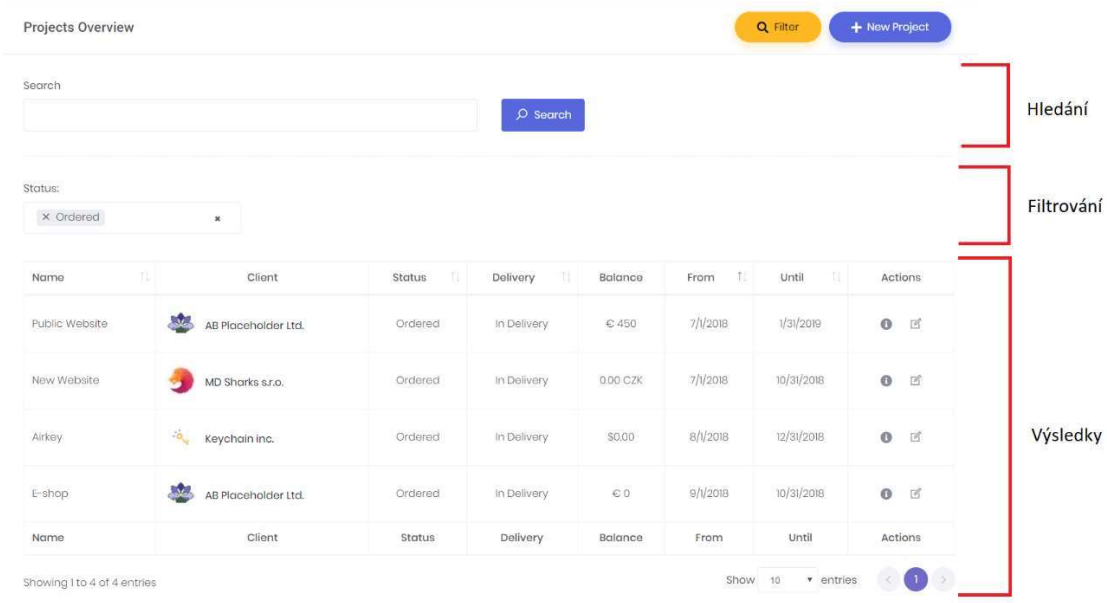
Tato funkcionalita je dosažena pomocí entit, jež je možné vidět na obrázku níže (viz Obrázek 34). Každá transakce je vázána k danému cílovému účtu a ovlivňuje jeho zůstatek. Samozřejmostí je i ovlivnění zůstatku zdrojového účtu, pokud byl definován. Pro zrychlení výpočtu zůstatků generuje každá transakce v tabulce *moneyaccountactivity* hodnotu, jež umožnuje náhled k rychlému přehledu výdajů a příjmů na účtu.



Obrázek 34 – Náhled na databázový model starající se o finanční účty

5.3 DataTables

Klíčovou komponentou uživatelského rozhraní je zobrazování informací v interaktivní tabulce (viz Obrázek 35). Tato komponenta je realizována pomocí doplňku DataTables[11] do knihovny jQuery[12]. Tento doplněk umožňuje vytváření interaktivních tabulek s rozličným nastavením a funkcemi.



Obrázek 35 – Ukázka DataTables v aplikaci

Implementace na klientské straně

Pro zjednodušení a zrychlení vytváření tabulek byla za pomoci návrhového vzoru *modul* vytvořena nadstavba nad doplňkem. Tato nadstavba zastřešuje jednotnou implementaci, vazbu se serverovou stranou a aplikační rozhraní pro snadné použití.

```
<table id="table1"></table>

<script type="text/javascript">
 var table = new AppDataTable("#table1", '/getsubscriptions')
   .addColumn("@SubscriptionsColumns.Id", (r) => { return r.id },
     "Id", true).center()
   .addColumn("@SubscriptionsColumns.Type", (r) => { return r.type },
     "Type").center()
   .addColumn("@SubscriptionsColumns.ValidFrom", (r) => {
     return r.validFromString }, "Valid  From", true).center()
   .addColumn("@SubscriptionsColumns.ValidUntil", (r) => {
        return r.validUntilString }, "Valid Until", true).center()
   .addColumn("Actions", (r) => {
     return `
       <a href="` + r.editUrl + `" title="Edit">
         <i class="la la-edit"></i>
       </a>`;
   }).center()
   .init();
</script>
```

Zdrojový kód 11 – Ukázka vytvoření tabulky pomocí nadstavby nad DataTables

Vytvoření interaktivní tabulky, jež implementuje vazbu na serverová data a další funckionalitu, je vidět na zdrojovém kódu výše (viz Zdrojový kód 11). K inicializaci je pouze nutné vytvořit prázdný HTML tag *table* a pomocí JavaScriptového kódu vytvořit instanci nového objektu, jenž se bude starat o onu tabulku. Objekt umožňuje dynamicky přidávat nové sloupce, které mohou a nemusí být vázané se serverovou stranou (vazba na serverovou stranu je nutná kvůli možnosti seřazení dat podle sloupců), a definovat jejich základní vlastnosti:

- obsah sloupce – HTML či řetězec,
- název sloupce,

- klíč sloupce – váže sloupec na serverovou aplikaci,
- řazení – zapnuto/vypnuto,
- zarovnání.

Nadstavba také umožnuje připojení vyhledávácího pole a různých filtrů, jimiž mohou být například výběr položek (dropdown) nebo pole pro vložení data či jeho rozsahu (date / daterange picker). Tyto filtry jsou automaticky nadstavbou registrovány na základě toho, že jsou umístěny ve stejném HTML kontejneru a jejich změnou dojde k automatickému opětovnému načtení výsledků v tabulce. Tyto filtry jsou automaticky připojeny k requestu na server, jenž doplněk vytváří, a serverová strana je dokáže rozpoznat.

Implementace na serverové straně

Doplněk umožňuje značnou konfiguraci sloupců i samotné tabulky, jež je například řazení a vyhledávání na úrovni sloupců. Tato konfigurace společně s daty, která na tabulku aplikuje uživatel, jsou součástí HTTP dotazu na server pro získání dat. Struktura dat v dotazu (viz Zdrojový kód 12) je poněkud nezvyklá a nelze ji automaticky mapovat na serverové modely reprezentující onu tabulku.

Pro implementaci DataTables na serverové straně byla provedena rešerše, zda existuje již hotové řešení, které by snadno a jednoduše řešilo zmíněnou problematiku mapování. Nejblíže ideálnímu řešení byla implementace, jejímž autorem je Anderson Luiz Mendes Matos[13]. Z této implementace byla převzata idea na vytvoření model binderu[3], ta usnadnila plnohodnotnou implementaci na serverové straně v podobě těchto klíčových funkcí:

- vazba sloupců na enumerátor,
- mapování mnoha různých filtrů přiřazených k tabulce,
- snadné napojení na databázový framework,

[3] Třída mapující data z HTTP dotazu na instanci objektu.

- vlastní řešení DTO objektů pro návrat dat do tabulky.

```
draw: 9
columns[0][data]: Name
columns[0][name]:
columns[0][searchable]: true
columns[0][orderable]: true
columns[0][search][value]:
columns[0][search][regex]: false
......
order[0][column]: 5
order[0][dir]: asc
start: 0
length: 10
search[value]:
search[regex]: false
companeero_custom[0][name]: Status
companeero_custom[0][value]: Ordered
```

Zdrojový kód 12 – Ukázka requestu doplňku DataTables

Model binder vytvořený pro mapování tohoto requestu namapuje veškerá příchozí data na objekt, jenž reprezentuje danou tabulku. To umožní jejich snadné použití a aplikování na filtrování databázových výsledků (viz Zdrojový kód 13).

```
[HttpPost]
public ActionResult GetTexts( [ModelBinder(
   typeof(DataTablesModelBinder<TextsDataTableFilter> ) )]
   TextsDataTableFilter model )
{
   var cols = model.GetColumns<TextColumns>();
   FilterResult<DynamicText> result =
      new FilterResult<DynamicText>( );

   DynamicTextFilter pf = new DynamicTextFilter( );
   result.TotalRecordsCount = base.Instance.DynamicText.Count( pf );

   pf.Count = model.PageSize;
   pf.Page = model.PageNumber;
   pf.Languages.AddRange( model.LanguageCodes );

   foreach ( var sort in cols.GetSortingColumns( ) )
   {
      switch ( sort.Key )
      {
         case TextColumns.Id: pf.Sort.Add(
              DynamicTextFilter.SortOptions.Id, sort.Desceding );
            break;
      }
   }
   pf.Search = model.SearchTerm;
   result.FilteredRecordsCount =
      base.Instance.DynamicText.Count( pf );
   result.Items = base.Instance.DynamicText.List( pf );
   return new DataTablesResponse( model,
      result.Items.Select( t => new DynTextDto( t ) ), result );
}
```

Zdrojový kód 13 – Aplikace příchozího DataTables requestu na filtrování výsledků z databáze

5.4 Aplikační rozhraní

Aplikační rozhraní, jak již bylo zmíněno, je budováno na platformě ASP .NET Web API. Volba této platformy byla naprosto jednoznačná vzhledem k tomu, že samotná webová aplikace je postavená na příbuzné platformě a sdílení některých součástí systému včetně business logiky systému bylo nasnadě. Rozhraní je vybudováno jako samostatná aplikace, což v budoucnu umožní škálovatelnost systému na další servery.

Implementace rozhraní není příliš odlišná od stavby aplikací na platformě ASP .NET MVC. Zásadním rozdílem je, že controllery aplikace dědí z třídy ApiController, která je součástí platformy a která řeší například automatickou serializaci objektů do požadovaného formátu. Aplikace samotná definuje vlastní bázový controller, jenž definuje chování aplikace pro odvozené controllery.

```
[InstanceOwnerAuthorize, ApiKeyAuthorize]
[ExecuteActionFilter]
[EnableCors( origins: "*", headers: "*",methods: "*" )]
[RequireProductionHttps]
public class ApiBaseController :
    WebApiController<AppInstance>
    {
    }
```

Zdrojový kód 14 – Definice bázového controlleru aplikačního rozhraní

Controller je nastaven tak, že v případě produkčního prostředí je automaticky provedeno přesměrování na zabezpečenou verzi spojení, tedy z protokolu HTTP na protokol HTTPS. Dále je povoleno se na controller dotazovat z jakéhokoli místa (origin, v terminologii HTTP, což je například doména jiné aplikace). Nezbytnou součástí definice je autentizace k aplikačnímu rozhraní, která je prováděna pomocí dvou hodnot v hlavičce dotazu:

- X-ApiKey – jedná se tajný klíč, jenž aplikace musí použít při dotazování na rozhraní,
- X- InstanceOwnerKey – jedná se o klíč, jenž definuje instanci aplikace (v tomto kontextu firmy, která je registrována k používání systému) a stará se o inicializaci správného databázového kontextu.

Na základě těchto hlaviček je aplikacím třetích stran povolen, či zamítnut přístup k rozhraní. Aplikační rozhraní je koncipováno jako bod pro

synchronizaci s dalšími systémy, proto například neobsahuje tokeny, které by byly závislé na uživateli.

Aplikační rozhraní obsahuje množinu endpointů, která je nezbytná pro synchronizaci s dalšími aplikacemi a která se týká správy časových záznamů a dalších. Rozhraní v současné verzi neumožňuje kontrolu systému ve stejné míře jako webová aplikace.

Vzhledem k tomu, že aplikační rozhraní je orientováno na synchronizaci s existujícími aplikacemi, nepoužívají se identifikátory zdrojů ze systému, ale je třeba je nastavit v prostředí webové aplikace tak, aby odpovídaly právě připojované aplikaci. Tato varianta byla zvolena kvůli tomu, že je potřeba na jedné z aplikací vytvořit tzv. mapu, tedy mapování objektů na objekty jiného systému. Volba budování mapy v tomto systému, který je předmětem této diplomové práce, usnadní připojování dalších aplikací. U těchto identifikátorů je při jejich zadávání validováno, zda jsou opravdu unikátní v rámci celého systému.

```
/// <summary>
/// Allocates time record in the system.
/// </summary>
/// <param name="model">Time record data.</param>
/// <response code="201">Time record allocated.</response>
/// <response code="400">Time record data error.</response>
/// <response code="401">User unauthorized.</response>
/// <response code="404">The work or account was not
found.</response>
[HttpPut, Route( "works/{work_code}" )]
[RequestExample( typeof( TimeRecord ) )]
public HttpResponseMessage AllocateTime( TimeRecord model )
{
  OrderItem work = Instance.M<OrderItemHandler>().GetGuid(
    model.WorkCode );

  if ( work == null )
  {
    return ResponseMessages.NotFound( Request,
      $"Work was not found." );
  }

  Account account = Instance.M<AccountHandler>().GetGuid(
     model.AccountCode );

  if ( account == null )
  {
    return ResponseMessages.NotFound( Request,
      $"Account was not found." );
  }

  if ( !work.IsEmployeeAssignedToWork( account ) )
  {
    return ResponseMessages.BadRequest( Request,
      "Account is not assigned to the work." );
  }

  if ( ModelState.IsValid )
  {
    work.AllocateTimeRecordForClient( model, account );
    return ResponseMessages.Created( Request );
  }
  return ResponseMessages.BadRequest( Request, ModelState );
}
```

Zdrojový kód 15 – Ukázka implementace endpointu aplikačního rozhraní

Na zdrojovém kódu (viz Zdrojový kód 15) je ukázka implementace jednoho z endpointů aplikačního rozhraní, který slouží k alokaci časového záznamu v systému. Jedná se o dotaz využívající HTTP metody PUT, která slouží k vytvoření či editaci již existujícího záznamu

5.4.1 Dokumentace aplikačního rozhraní

REST architektura nevyžaduje tvoření dokumentace rozhraní, protože jeden z principů je objevitelnost rozhraní tzv. HATEOAS[4]. Tento princip umožnuje objevovat dostupné akce nad daným zdrojem pomocí přidružených informací v těle odpovědi serveru (viz Zdrojový kód 16).

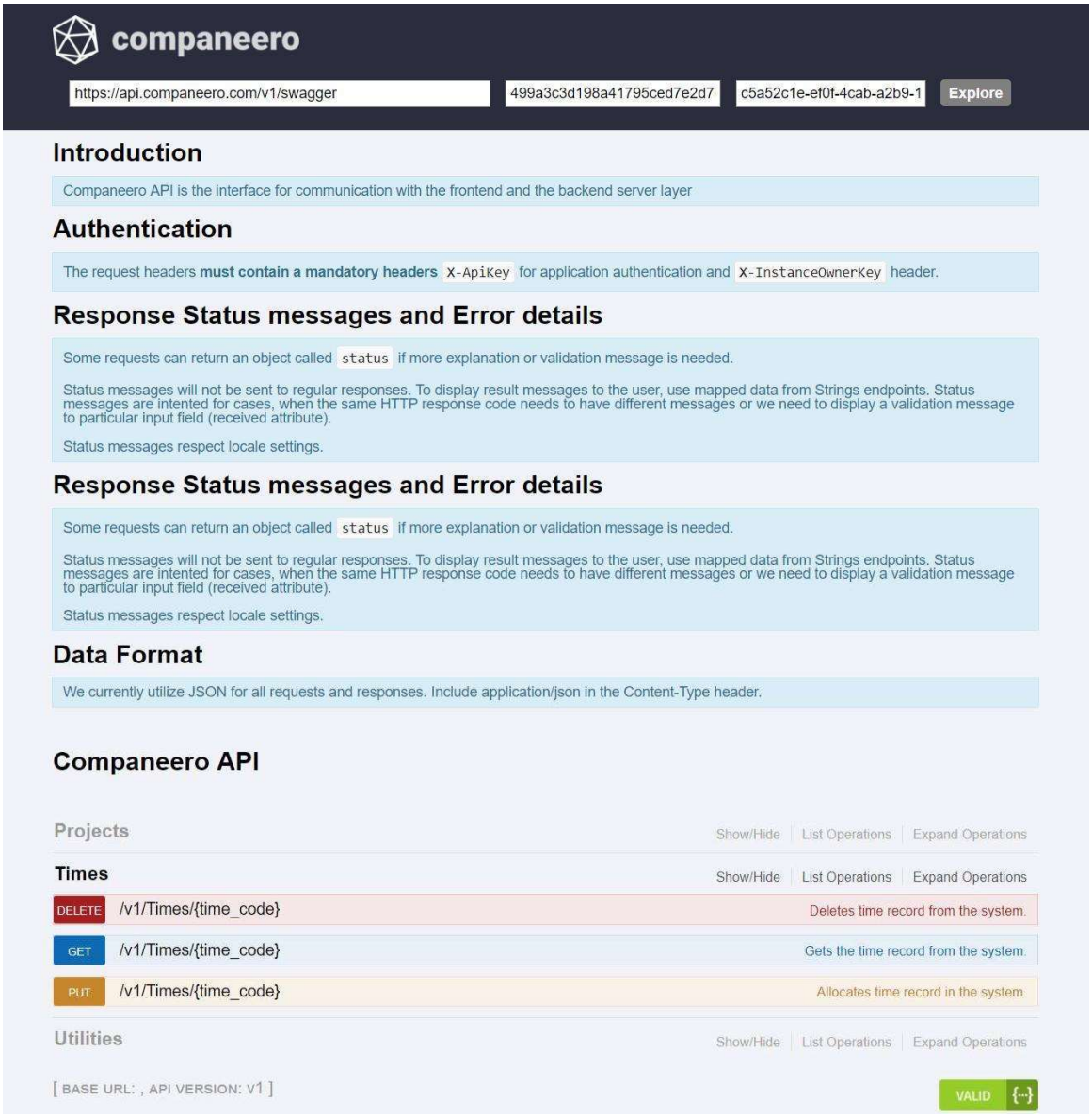
```
{
  "surname":"John",
  "lastname":"Doe"
  "links":[
    {
      "rel":"self",
      "href":"http://page.com/people/1/"
    },
    {
      "rel":"contacts",
      "href":"http://page.com/people/1/contacts/"
    }
  ]
}
```

Zdrojový kód 16 – Ukázka HATEOAS

Ovšem pro větší komfort a testovatelnost rozhraní byl do aplikace integrován nástroj Swashbuckle[14][15]. Tento nástroj generuje Swagger dokument[15], jenž byl navržen speciálně pro rozhraní typu REST a popisuje rozhraní a jeho endpointy ve standardizovaném formátu. Síla tohoto nástroje spočívá v tom, že dokumentace aplikačního rozhraní je tvořena na základě komentářů ve zdrojovém kódu, tudíž je vždy zaručeno, že dokumentace odpovídá realitě a není třeba udržovat dokumentaci rozhraní v jiném dokumentu či aplikaci. Udržitelnost takové dokumentace by časově byla velice nákladná. Výsledkem konfigurace je interaktivní stránka (viz Obrázek

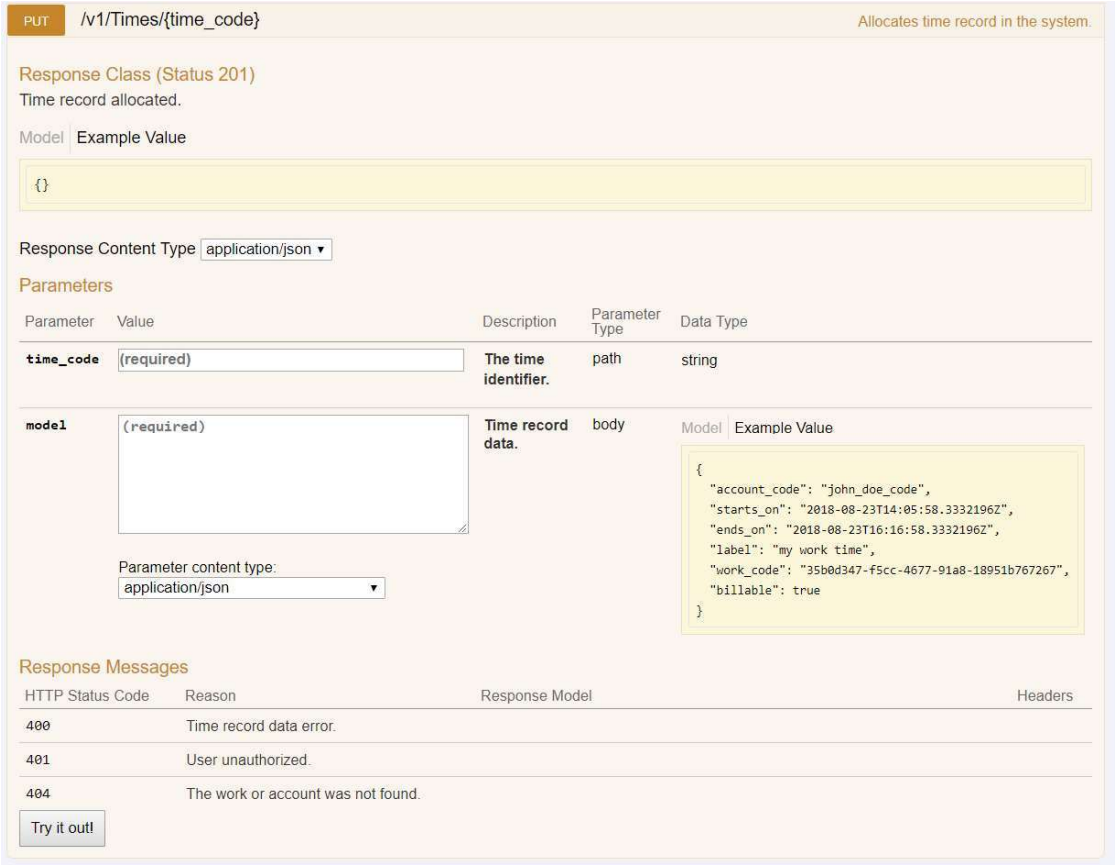
[4] *What is HATEOAS and why is it important for my REST API?. The RESTful cookbook* [online]. © 2012-2014 [cit. 2018-05-1]. Dostupné z: http://restcookbook.com/Basics/hateoas/

36), kde vývojář pracující na integraci rozhraní do jiné aplikace vidí exaktní popis endpointů a může je rovnou otestovat.



Obrázek 36 - Ukázka dokumentace aplikačního rozhraní

Veškeré endpointy jsou jasně popsány, dokumentace tak obsahuje informaci o použité HTTP metodě, stavových kódech a vzory dat, které server akceptuje nebo vrací (viz Obrázek 37).



Obrázek 37 - Ukázka testovacího rozhraní pro endpoint

5.5 Windows služba

Pro vykonávání asynchronních, plánovaných či dlouhotrvajících operací byla vytvořena aplikace, jež je na aplikační server nainstalována jako služba. Tato aplikace vychází ze základní konfigurace služby, která je již obsažena v dodaném frameworku, jelikož i tato služba musí řešit instanční závislost při jejím běhu. Tato závislost je řešena tak, že služba automaticky cykluje přes všechny dostupné instance v systému a spouští operace na příslušném databázovém kontextu.

5.5.1 Vytvoření Windows služby

Vytvoření Windows služby na PTS frameworku spočívá ve vytvoření konzolové aplikace a její dodatečné konfigurace umožňující instalaci této aplikace jako služby. Tato funkcionalita tak umožnuje vykonávání asynchronních operací ve spuštěné konzolové aplikaci, případně ve službě,

která je obecně preferována v produkčním prostředí, jelikož služby jsou dobře vázány s operačním systémem Windows a je tak například zaručen jejich start i po restartování serveru. Běh v konzolové aplikaci je zejména určen pro vývojáře, kteří tak mohou snadno vidět průběh vykonávání operací.

```
public abstract class LocalServiceInstaller : Installer
{
   protected System.ServiceProcess.ServiceProcessInstaller Installer;
   protected System.ServiceProcess.ServiceInstaller Service;

   public LocalServiceInstaller( )
      : base( )
   {
      this.Installer = new
         System.ServiceProcess.ServiceProcessInstaller( );
      this.Service = new System.ServiceProcess.ServiceInstaller( );

      this.Installer.Account =
         System.ServiceProcess.ServiceAccount.LocalSystem;
      this.Installer.Password = null;
      this.Installer.Username = null;

      this.Installers.AddRange( new
System.Configuration.Install.Installer[] { this.Installer,
this.Service} );
   }
}
```

Zdrojový kód 17 – Ukázka definice Windows služby

Na výše uvedeném zdrojovém kódu (viz Zdrojový kód 17) je možné vidět bázovou třídu, ze které je aplikace odvozena. Tato třída dědí z třídy *Installer,* jež je součástí .NET frameworku a v jejímž konstruktoru dochází k registraci samotné služby. Na zdrojovém kódu níže (viz Zdrojový kód 18) je možné vidět zjednodušenou verzi jak registrovat služby k jejich běhu. Službou se tak může stát libovolná třída, jež dědí z třídy *ServiceBase.* Tyto ukázky jsou pouze nástinem toho, jak registrovat Windows službu[16]; implementace v dodaném frameworku je o řád složitější, jelikož poskytuje dodatečné funkce k časování, logování a přepínání mezi jednotlivými instancemi.

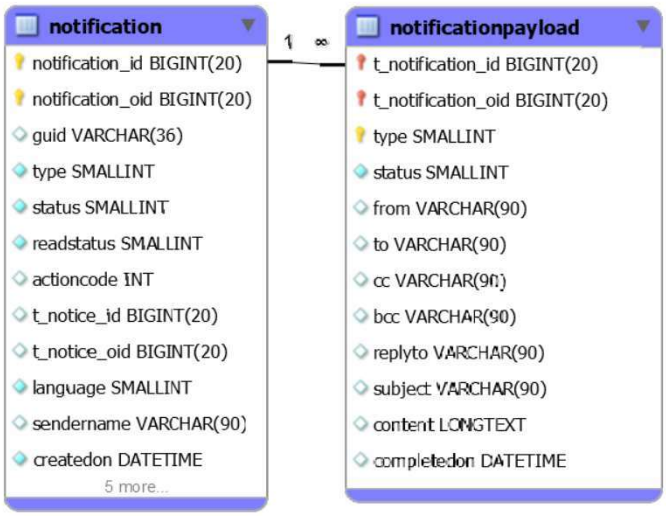
```
static void Main(string[] args)
{
    ServiceBase[] ServicesToRun = new ServiceBase[] { new
       MyNewService(args)
    };
    ServiceBase.Run(ServicesToRun);
}
```

Zdrojový kód 18 – Ukázka registrace služeb

5.5.2 Asynchronní operace vykonávané systémem

Odesílání notifikací

Veškeré systémem vygenerované notifikace jsou uloženy v databázi (viz Obrázek 38), model této databáze umožnuje vytvářet notifikaci, která má několik různých obsahů pro různé distribuční kanály. Tudíž je tedy možné naplánovat notifikování uživatele přes SMS a e-mail. Distribuční kanál určuje vlastnost *Type* entity *NotificationPayload*.



Obrázek 38 – Databázová struktura notifikací

Služba každou minutu hledá notifikace, které ještě nebyly odeslány (mají status *Queued)* a mají čas odeslání v minulosti. Všechny nalezené notifikace (viz Zdrojový kód 19) jsou přes správný kanál odeslány, v případě úspěchu jsou notifikace označeny jako odeslané a nebudou již nalezeny při dalším odesíláním.

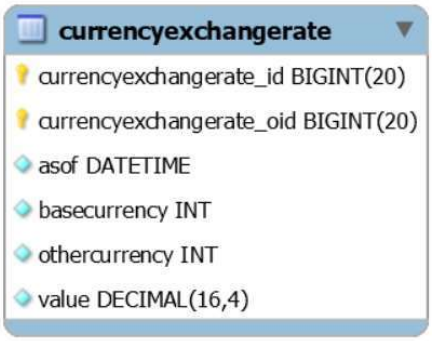
```
public void CronSendQueuedNotifications( )
{
   NotificationPayloadFilter filter =
      new NotificationPayloadFilter( );
   filter.Statuses.Add( NoticeStatus.Queued );
   filter.NotificationFilter.SendOn.Until = DateTime.Now;

   filter.Types.Add(NotificationType.EmailText);
   var payloads = Instance.M<NotificationPayloadHandler>( )
      .List( filter );

   foreach ( var payload in payloads )
   {
      var email = payload.GetNotification( )?
      .GetRecipients( )?.First( )?.GetContact( ContactType.Email );
      if ( Instance.ManagementTool.Notifications
         .SendEmail( payload, email ) )
      {
         RuntimeLogs.Log( $"Notification #{payload.Id}-{payload.Type}
has been sent." );
      }
   }
}
```

Zdrojový kód 19 – Odesílání plánovaných notifikací

Získávání konverzních poměrů měn

Vzhledem k požadavkům systému, který obsahoval možnost operovat s transakcemi v různých měnách, byly do poskytnutého frameworku dodány funkce, které umožnují vytváření účetních transakcí mezí dvěma měnami. Transakční funkcionalita byla již součástí frameworku, ale vzhledem k tomu, že doposud firmou budované systémy byly vždy pouze v rámci jedné měny, bylo nutné tuto funkcionalito dodat.

Prvním krokem byl návrh nové entity (viz Obrázek 39), která by umožňovala ukládání konverzních poměrů mezi jednotlivými měnami. Tato entita a její příslušné provázání s účetními entitami (a jejich úpravou) byly po akceptaci přidány jako součást frameworku a implementovány.



Obrázek 39 – Entita pro ukládání konverzního poměru mezi měnami

Pro získání konverzních poměrů byl vytvořen task (úkol) služby, jenž periodicky stahuje z dříve bezplatné služby Fixer.io[17]. Tato služba byla v červnu 2018 zpoplatněna, tudíž musela být implementována nová služba Free Currency Converter[18]. Obě tyto služby definovaly jednoduché REST aplikační rozhraní, jehož parametry byly zdrojová a cílová měna. Rozhraní poté vracelo konverzní poměr mezi danými měnami. Za pomoci této služby se jednou denně uloží do databáze konverzní mapa mezi podporovanými měnami aplikace. Funkcionalita byla na úrovni frameworku abstrahována do jednotného rozhraní, jež umožnuje definice libovolného zdroje konverzních poměrů dle potřeb aplikace (viz Zdrojový kód 20). V současné době aplikace není budována jako plnohodnotná účetní aplikace, avšak pokud bude v budoucnu používána jako fakturační aplikace, která by spravovala reálné účetnictví, je nasnadě implementování aplikačního rozhraní České národní banky, které je podle § 4 odst. 5 zákona o DPH zdrojem kurzovního lístku[19]. Díky tomu, že jsou konverzní poměry ukládány do databáze s časovou informací, lze tak dohledat kurz k libovolnému datu.

```
public class ExchangeRateTask : CurrencyExchangeRateTask
{
  protected override void InitProvider( LocalInstance instance )
  {
    FreeCurrencyConverterProvider.Currencies =
      AppInstance.SupportedCurrencies;
    base.Provider = new FreeCurrencyConverterProvider( );
  }
}
```

Zdrojový kód 20 – Inicializace zdroje konverze měn ve službě

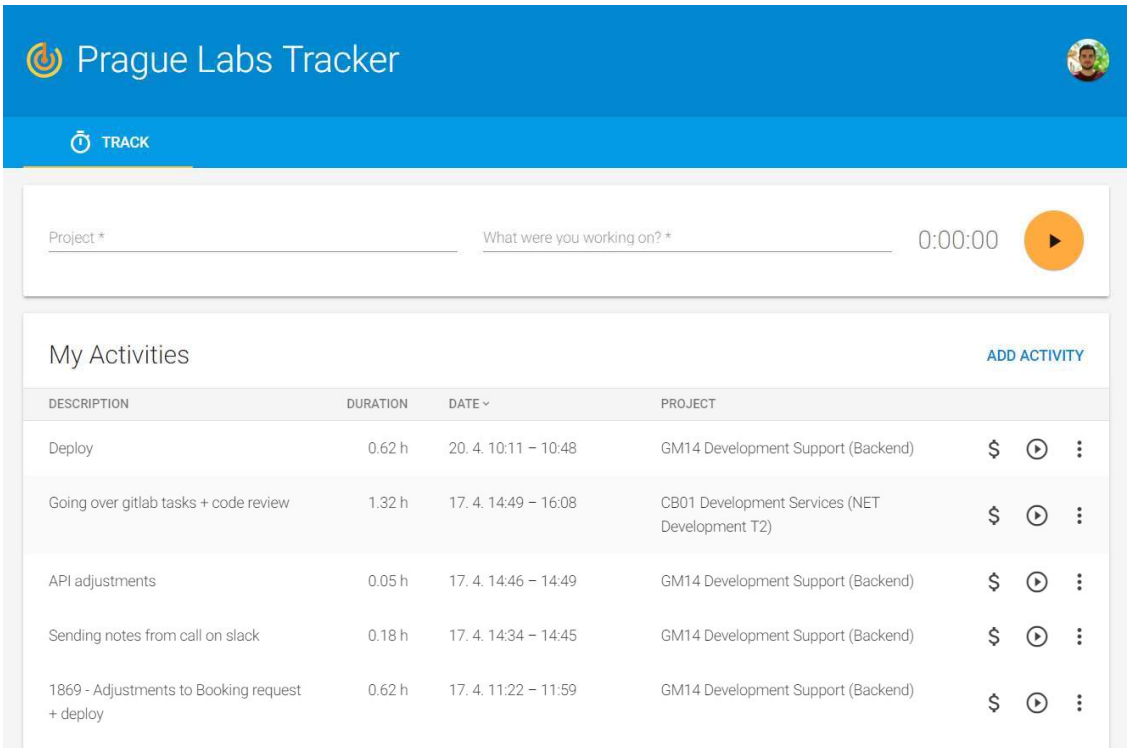
6 Integrace aplikace do provozu u partnerské firmy

Pro uvedení do provozu bylo nutné výslednou aplikaci a její součásti publikovat veřejně online, jelikož by provoz na dedikovaných lokálních serverech umístěných v sídle partnerské firmy byl ve výsledku dražší na provoz a neodpovídal by požadavkům na možný komerční provoz aplikace. Proto byl pro provoz zakoupen jeden Windows server pro běh samotné aplikace a jeden Linux server pro hostovaní statických souborů a obrázku. Servery byly zakoupeny na platformě Amazon Web Services od společnosti Amazon. Pro snadnější přístup k aplikaci byla také zakoupena doména, na které je aplikace dostupná.

Po konfiguraci obou serverů, DNS záznamů domény a publikování aplikací na server bylo vše připraveno k používání. Následujícím krokem bylo tedy vytvoření instance pro partnerskou firmu, vytvoření účtů jejich reprezentantům a nastavení prostředí – to vše v rámci uživatelského rozhraní aplikace.

6.1 Integrace aplikace na měření času

Partnerská firma disponuje vlastním řešením pro měření času odvedené práce zaměstnanců v podobě jednoduché webové aplikace (viz Obrázek 40) napsané v jazyce Angular 6 s daty uloženými ve Firebase Cloud Firestore[20] databázi.



Obrázek 40 – Aplikace pro měření času partnerské firmy

Tato NoSQL databáze je rozšiřitelná pomocí tzv. Cloud Functions, což je Node.js kód, který slouží jako handler událostí nad entitami v dané databázi. Pro každou entitu je možné registrovat 4 základní typy handlerů:

- *onCreate*, událost vytvoření nového záznamu entity,
- *onUpdate,* událost aktualizace existujícího záznamu,
- *onDelete*, událost smazání záznamu,
- *onWrite,* událost značící jakoukoliv změnu (výše uvedené události).

Pro aplikační rozhraní byla implementována třída, jež zastřešuje veškerou synchronizaci pomocí http dotazů mezi klientskou aplikací a rozhraním. Vytvořením této třídy tak došlo k oddělení logiky existujících handlerů od logiky synchronizace, která se tak zmenšila na pouhé volání metody oné třídy. Implementaci této třídy najdete na přiloženém médiu.

Alokace času

Pro zachycení změn v tabulce pro evidenci časových záznamů byl použit handler *onUpdate* (viz Zdrojový kód 22), který pomocí implementovaného API klienta alokuje časový záznam v aplikaci, kde dojde k vytvoření či úpravě časového záznamu v systému. Důvodem proč synchronizace nebyla navázána i na handler *onCreate* je, že v tuto chvíli by časový záznam neměl žádný čas konce, jelikož v klientské aplikaci dochází k měření v reálném čase, kdy se záznamy každých 30 vteřin aktualizují. Tudíž k prvnímu vytvoření záznamu v systému dojde až po uplynutí 30 sekund od začátku měření nebo v případě jeho ukončení.

```
export const activityUpdateListener =
functions.firestore.document('activities/{documentUid}').onUpdate((
change, context) => {
    const snap = change.after;
    const moment = require('moment');

    const companeeroAPI = new CompaneeroAPI();
    return companeeroAPI.activityPut(snap).then(() => {
        return null;
    });
});
```

Zdrojový kód 22 – Handler zachytávající změnu záznamu v klientské aplikaci

```
export const activityDeleteListener =
functions.firestore.document('activities/{documentUid}').onDelete((s
nap, context) => {
    const companeeroAPI = new CompaneeroAPI();
   return companeeroAPI.activityDelete(snap).then(() => {
        return null;
    });
});
```

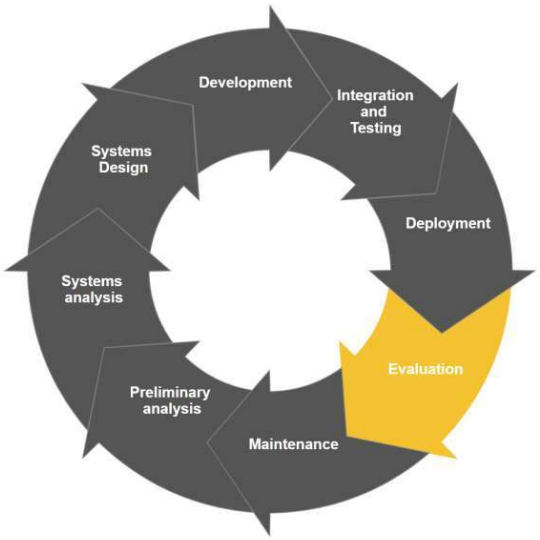
Zdrojový kód 21 – Handler zachytávající smazání záznamu z klientské aplikace

Mazání času

Zachycení mazání záznamu je vázáno pomocí handleru (viz Zdrojový kód 21) *onDelete* a opět dochází k volání API endpointu aplikace, tentokrát pomocí metody *DELETE*.

7 Získání zpětné vazby

Získání zpětné vazby je nedílnou součástí životního cyklu každého projektu, potažmo softwarové aplikace. Tyto životní cykly popisuje obecný model Systems Development Life Cycle (SDLC) [21], ze kterého pak vycházejí iterativní metodiky pro projektové řízení, například Scrum či Waterfall. Během zpracovávání této diplomové práce prošla aplikace životními cykly, a to od analýzy až po integraci, nasazení a testování v prostředí klientské firmy. Následujícím krokem životního cyklu je tedy zhodnocení vybudovaného systému.



Obrázek 41 - Zjednodušený SDLC model[5]

Pro získání zpětné vazby byla zvolena metoda nahrávaného rozhovoru, což autor této práce vyhodnotil zároveň jako osobnější a detailnější způsob získání potřebných informací. Rozhovor byl veden se zástupcem klientské firmy, která bude primárním uživatelem systému. Mgr. Martin Pultzner byl jakožto budoucí hlavní uživatel aplikace (z pozice COO[6]) na základě testovacího používání obeznámen s tím, jak aplikace funguje. Audio

[5] Z modelu byla vynechána fáze Disposal, jež značí terminaci aplikace/projektu.
[6] Chief Operations Officer

nahrávku rozhovoru a jeho přepis je možné nalézt v přílohách (Příloha A a Příloha D).

7.1 Připomínky k funkcionalitě aplikace a její hodnocení

V této kapitole jsou uvedeny připomínky a hodnocení, jež Mgr. Martin Pultzner postupně uvedl během rozhovoru.

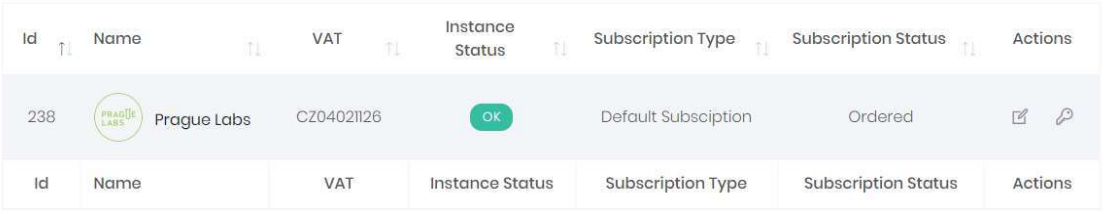
7.1.1 Úroveň DPH

[Martin Pultzner] Co se týče DPH (VAT rate), tak to tak dát třeba nějakým procentem.

Zadávání úrovně DPH je v systému řešeno pomocí numerických polí, jež mají rozsah od 0 do 1, zadávání DPH 21 % je tedy řešeno jako vložení hodnoty „0,21“, jež není uživatelsky přívětivá. Tento způsob zadávání byl zvolen, aby nemuselo docházet ke konverzi čísla, jelikož vybudovaný účetní systém i databáze pracují s hodnotu v desetinném čísle. K této připomínce se nabízí řešení v MVC aplikaci, která bude tyto hodnoty automaticky konvertovat pro zobrazování a při odesílání formuláře tak, aby systém vždy pracoval s desetinným číslem a uživatel s procentuální hodnotou.

7.1.2 Uživatelské akce v aplikaci

[Martin Pultzner] Možná takový feedback (a to se asi prolíná celým tím systémem, je určitě hrozně jednoduché to udělat), že já vždycky ve všech systémech mám tendenci, když vidím jméno někoho nebo jméno společnosti, jméno uživatele a tak podobně, tak rovnou na něj jakoby kliknout, případně kliknu na tu ikonku – a tady to jakoby nejde a vždycky si musím nějak vpravo vybrat tu akci. … ono to i pro ten touch… pro ten dotykový display je super, že si potom vždycky kliknu jenom na toho člověka a hned mi to otevře ten pohled.



Obrázek 42 – Ukázka uživatelských akcí

Uživatelské akce nad danou entitou jsou v případě tabulek a dalších widgetů řešeny pomocí akčních tlačítek, které mají specifickou funkci, jako například Editace a Přihlášení (viz Obrázek 42). Tato akční tlačítka vyjadřují svou funkci pomocí ikonek a mohou být také relativně malá pro dotykové ovládání. Řešením tohoto problému, jak již Mgr. Pultzner uvedl, je přidání výchozí akce na kliknutí na jméno či obrázek.

7.1.3 Definování ceny služby

> [Martin Pultzner] Jo, o tomhle jsme se bavili i s kolegama, že tohle je hodně důležité, že ten systém umí pracovat jak s man days, tak s hodinovýma sazbama, protože ono se to hodně míchá mezi klientama… a taky občas pracovníkama, takže tohle je důležité. A taky je tam možnost mít ty services v různých měnách, což je pro nás taky hrozně důležité, takže to je určitě jedna z nejdůležitějších součástí.

Respondent pozitivně hodnotil, že systém dokáže účtovat služby v různých měnách a s různým systémem účtování služby (man day, man hour, fixní sazba).

7.1.4 Zdroje kandidátů (HR)

> [Martin Pultzner] Jo, to se mi hrozně líbí, ten graf tam, za to velká pochvala – já mám grafy rád a vím, že Pavel, co mi říkal, tak Pavel má taky grafy rád, takže to je… to, že tam je vidět, kolik těch kandidátů přišlo z jakého zdroje, tak to je super. Víc takových grafíků!

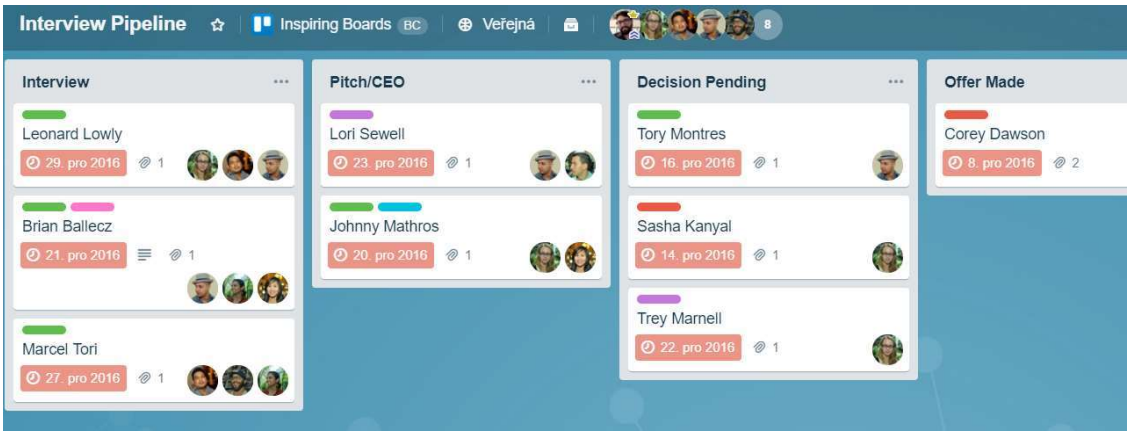
Obrázek 43 – Ukázka grafů zobrazujícího statistiku HR zdrojů

Mgr. Pultzner a kolektiv obecně pozitivní hodnotí zobrazování dat v systému pomocí grafů, konkrétně poskytnutých kandidátů (viz 5.2.5 Kandidáti).

7.1.5 Proces přijímání kandidátů

> [Martin Pultzner] My teď pro hireovací proces používáme Trello, a tam je to vlastně stylem karet, které se přesouvají z jednoho sloupečku do dalšího podle toho, v jakém stavu ten kandidát je, a vlastně tady zatím jsou dva stavy – s těm stavama tam už počítáš a vlastně tam je buď není hired, anebo je hired. Tak jestli by šlo (a bylo by to podle mě hodně důležité v tom celém HR procesu) mít přehled, v jaké fázi toho onboarding procesu nebo toho hireovacího procesu se nachází. … Teď to úplně stačí, ale aby to mohlo fungovat potom úplně přesně, třeba jako to máme teďka nastavený – to by bylo úplně perfektní, kdyby tam těch příznaků „hired“, „not hired“ bylo víc, třeba šest, sedm, ale to asi taky není problém…

Proces přijímání kandidátů je dle zpětné vazby potřeba rozšířit o další stavy, v nichž se kandidát nachází. Možnými dalšími stavy kromě přijat/nepřijat (hired/not hired), jsou stavy jako například „pozvat na pohovor“, „odmítnut“, „druhé kolo“ a další. Tuto funkcionalitu bude nutné do aplikace postupně implementovat, v úvahu také připadá předělání uživatelského rozhraní z tabulky do kanban boardu, obdobně jako v aplikaci Trello.



Obrázek 44 – Ukázka nastavení aplikace Trello[7] pro HR účely.

7.1.6 Nastavení DPH pro externisty a klienty

> [Martin Pultzner] Ještě teda jedna věc k těm zaměstnancům, jestli můžu – když jsem tam spravoval nějaké věci a lidi, tak potom jsem narazil na to, že tu sekci „Billing" nemůžu editovat.

Sekce „Billing" obsahuje nastavení úrovně DPH, DIČ/IČ a informaci, zda je subjekt plátcem DPH; toto nastavení je fixní od vytvoření entity, jelikož změny v čase by mohly negativně ovlivnit výpočty a nastavení systému. Jako řešení tohoto problému se nabízí přesunutí těchto informací na úroveň objednané služby, která je v čase závislá. Tato úprava systému bude nezbytná vzhledem k tomu, že se stává, jak Mgr. Pultzner poznamenal, že externisté se například stávají plátci DPH apod.

7.1.7 Účetní adresa

> [Martin Pultzner] A ještě druhá věc: k té adrese, která tam teď už je v profilu toho uživatele, tak ještě přidat billing adresu – ona může být odlišná od adresy trvalého bydliště, kam třeba může zasílat nějaké písemnosti, ale billing adresa, na kterou se vystavují faktury, ta může být jiná…

Mgr. Pultzner dodal, že ke všem subjektům, klientům i externistům, je třeba doplnit účetní adresu, která se může lišit od místa trvalého bydliště či

[7] Ukázka z oficiálního dema aplikace Trello, dostupná z https://trello.com/b/UQqGyNFG/interview-pipeline

sídla firmy. Tato úprava systému nebude zásadním problémem vzhledem k tomu, že databázový model umožňuje uložit více adres k jednomu profilu.

7.1.8 Automatické rozdělování bonusů

> [Petr Žďárský] Kdysi jsme se bavili o tom, jak přidávat bonusy zaměstnancům … můžeme zadat bonus, který jde přímo z účtu Prague Labs, ale to se pak nedělí mezi ty projekty, případně klienty. Takže to je asi věc, na které taky můžeme v budoucnu zapracovat.
>
> [Martin Pultzner] Jo, teď se to vlastně dělá manuálně, že účetní to rozpočítává poměrově podle počtu hodin, které jaký zaměstnanec trávil na kterém projektu, tak ten bonus rozpočítává podle toho. Takže teď je to supermanuální, pokud se to tímhle zautomatizuje, tak tím líp.

Během rozhovoru bylo jedním z témat také to, jak aplikace rozděluje vyplacené bonusy zaměstnancům. Tyto bonusy by měly být rozpočítány do výdajů projektu úměrnou částkou dle odpracovaných hodin.

7.2 Vyhodnocení rozhovoru

Mgr. Martin Pultzner, jenž v rozhovoru zastupoval vedení firmy Prague Labs s.r.o., pozitivně hodnotil vytvořenou aplikaci a potvrdil, že aplikace splnila jeho očekávání. Nejlépe hodnoceno bylo zobrazování dat, tedy grafy a tabulky, a také účetní schopnosti systému umožňující pracovat se službami, které jsou v různých měnách či účtovacích periodách. Hlavními nedostatky aplikace jsou fixní účetní informace a způsob implementace udělování bonusů. Tyto a další připomínky budou sloužit jako seznam vylepšení, na kterých se bude v budoucnu pracovat, avšak nyní nejsou překážkou používání systému.

Závěr

Prvním výsledkem této práce je analýza průzkumu mezi menšími firmami a agenturami působícími v IT odvětví a v přidružených oborech. Tato analýza dodala přehled aplikací, které firmy používají ke každodenní agendě, a jejich porovnání.

Druhým výsledkem této práce jsou návrh a implementace aplikace, která pokrývá větší část funkcí vzešlých z průzkumu a potřeb klientské firmy. Architektura této aplikace umožňuje provoz u více firem, čímž byla splněna jedna z podmínek pro možný budoucí komerční provoz. Tato aplikace také obsahuje rozhraní pro přístup vyššího managementu firmy, který tak získá ucelený přehled o ziskovosti projektů a souvisejících nákladech. Systém také disponuje aplikačním rozhraním, jež je možné použít pro napojení a synchronizaci s aplikacemi třetích stran.

Dalším výstupem byla integrace budované aplikace pomocí aplikačního rozhraní do existující aplikace klientské firmy. To umožňuje synchronizaci času měřeného zaměstnanci a dovoluje klientské firmě použití aplikace.

Hlavním cílem této práce bylo ověření konceptu funkčnosti aplikace a zjištění, zda má taková aplikace své místo na existujícím trhu. Výsledky dotazníkového šetření, integrace aplikace do provozu klientské firmy a následné zpětné vazby potvrdily funkčnost konceptu i zájem firem o takovou aplikaci. Vzhledem k tomu, že aplikace v současné míře nepokrývá veškerou požadovanou funckionalitu (například omezené přístupy pro další uživatelské role) a nejsou prozatím zpracovány připomínky vzešlé ze zpětné vazby, není aplikace na trhu konkurenceschopná a bude nadále podléhat dalšímu vývoji a optimalizaci, což by z ní mělo vytvořit komerčně aplikovatelný produkt.

Zdroje

[1] *Google Forms: create and analyze surveys, for free.* [online]. Mountain View, 2018 [cit. 2018-02-12]. Dostupné z: https://www.google.com

[2] *Costlocker: Timesheet & time tracking app* [online]. Praha: Costlocker SE, 2018 [cit. 2018-05-27]. Dostupné z: https://costlocker.com/

[3] *Basecamp: Project Management & Team Communication Software* [online]. Chicago: Basecamp, 2018 [cit. 2018-05-27]. Dostupné z: https://basecamp.com/

[4] *Toggl: Free Time Tracking Software* [online]. Tallinn: Toggl OÜ, 2018 [cit. 2018-05-27]. Dostupné z: https://toggl.com/

[5] *TMetric: Free Employee Time Tracking Software & App* [online]. Praha: Devart, 2018 [cit. 2018-05-27]. Dostupné z: https://tmetric.com/

[6] *Budgeta: Budgeta is Budgeting and Planning for Business Done Right* [online]. Mountain View,: Budgeta, 2018 [cit. 2018-05-27]. Dostupné z: https://budgeta.com/

[7] *The ASP.NET Site: ASP.NET MVC* [online]. Redmond: Microsoft, 2018 [cit. 2018-05-29]. Dostupné z: https://www.asp.net/mvc

[8] *The ASP.NET Site: ASP.NET Web API* [online]. Redmond: Microsoft, 2018 [cit. 2018-05-29]. Dostupné z: https://www.asp.net/web-api

[9] *FIELDING, Roy Thomas. Architectural Styles and the Design of Network-based Software Architectures.* [online]. University of California, 2000 [cit. 2018-03-15]. Dostupné z:https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

[10] Metronic - The Ultimate Bootstrap & Angular 6 Admin Theme. *KeenThemes.com - Bootstrap Dashboard Themes &*

Templates[online]. Keenthemes, 2018 [cit. 2018-09-01]. Dostupné z: https://keenthemes.com/metronic/

[11] *DataTables: Table plug-in for jQuery* [online]. Scotland: SpryMedia, ©2007-2018 [cit. 2018-06-02]. Dostupné z: https://datatables.net

[12] *JQuery* [online]. The jQuery Foundation, 2018 [cit. 2018-06-02]. Dostupné z: https://jquery.com/

[13] *ALMMa/datatables.aspnet: Microsoft Asp.Net server-side support and helpers for jQuery DataTables* [online]. Brazil: Anderson Luiz Mendes Matos, 2017 [cit. 2018-06-02]. Dostupné z: https://github.com/ALMMa/datatables.aspnet

[14] *Swashbuckle. 2015. GitHub* [online]. [cit. 2015-08-02]. Dostupné z: https://github.com/domaindrivendev/Swashbuckle

[15] *Swagger / The World's Most Popular Framework for APIs.* [online]. 2018. [cit. 2018-05-02]. Dostupné z: http://swagger.io/

[16] Walkthrough: Creating a Windows Service Application in the Component Designer. *Microsoft Docs* [online]. Redmond: Microsoft, 2017 [cit. 2018-05-31]. Dostupné z: https://docs.microsoft.com/cs-cz/dotnet/framework/windows-services/walkthrough-creating-a-windows-service-application-in-the-component-designer

[17] *Foreign exchange rates and currency conversion API* [online]. Wien: Fixer.io, ©2018 [cit. 2018-08-31]. Dostupné z: https://fixer.io/

[18] *Free Currency Converter API: Currency Conversion Tool For Devs* [online]. Manny Vergel, 2016 [cit. 2018-08-31]. Dostupné z: https://free.currencyconverterapi.com/

[19] Kurz pro přepočet vystavené faktury pro účely DPH - Aktuality - Účetní portál - účetnictví, mzdy, daně, audit - informační portál v oblasti účetnictví a daní. *Účetní portál: účetnictví, mzdy, daně, audit - informační portál v oblasti účetnictví a daní* [online]. ©2018

[cit. 2018-06-03]. Dostupné z: https://www.ucetni-portal.cz/kurz-pro-prepocet-vystavene-faktury-pro-ucely-dph-966-a.html

[20] *Firebase: Cloud Firestore* [online]. 2018 [cit. 2018-05-21]. Dostupné z: https://firebase.google.com/docs/firestore/

[21] Systems development life cycle. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-, 2018-05-22 [cit. 2018-05-22]. Dostupné z: https://en.wikipedia.org/wiki/Systems_development_life_cycle

Příloha A **Obsah přiloženého média**

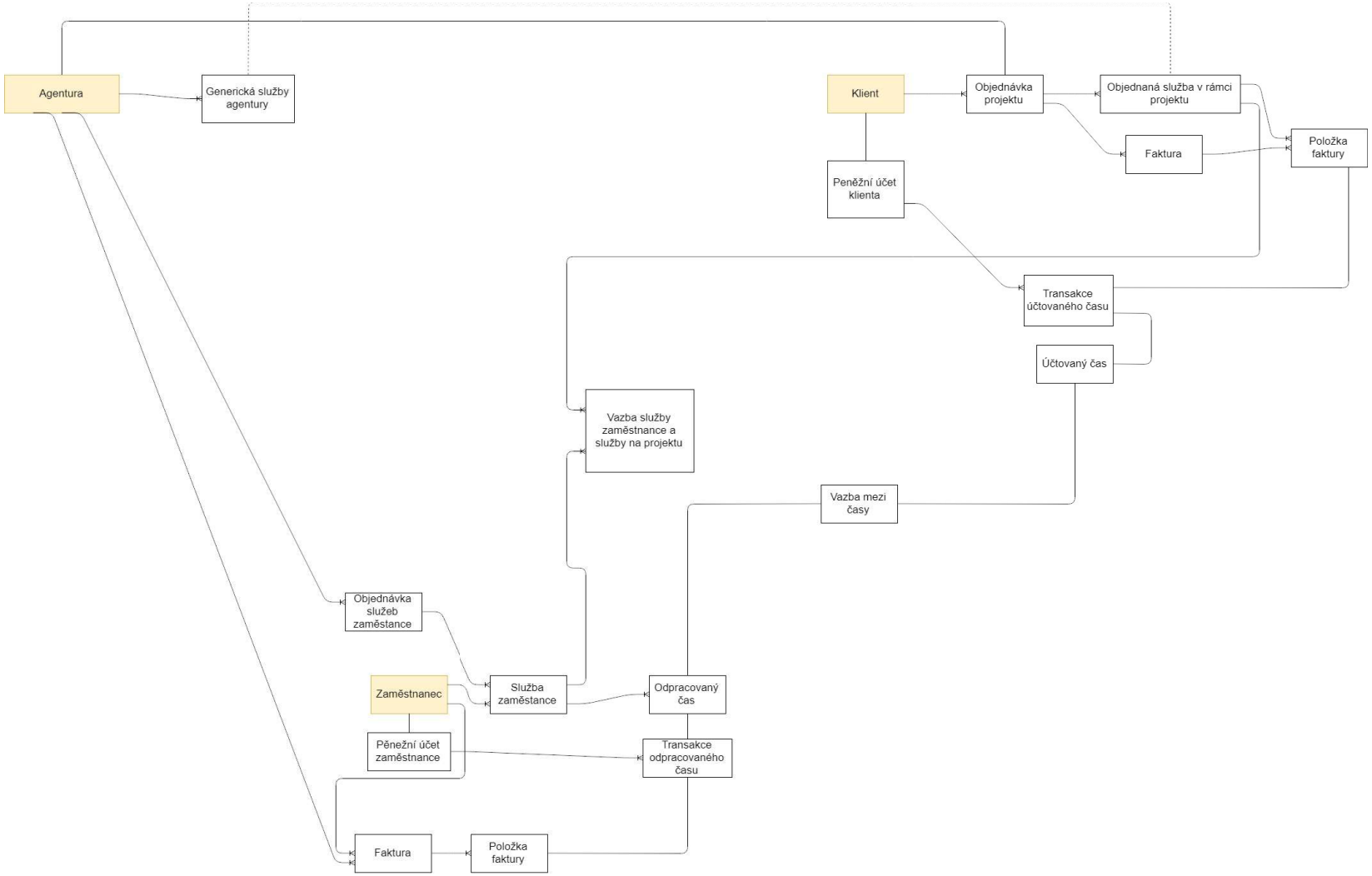
- Dotazník použitý pro průzkum.
- Výsledky dotazníkového průzkumu v .xlsx dokumentu.
- Diplomová práce *Webová aplikace pro administraci projektů* ve formátu PDF.
- Implementace aplikačního rozhraní v jazyce Angular.
- Zdrojové kódy aplikace.
- Úpravy provedené na databázovém modelu.
- Audio nahrávka rozhovoru pro získání zpětné vazby.

Příloha B **Srovnání aplikací**

Aplikace	Trial verze	Způsob platby	Varianta a cena			
Costlocker	30 dní	uživatel/měsíc	12 - 20 USD dle četnosti plateb			
Toggl	30 dní	uživatel/měsíc	Free 0 USD	Starter 9 USD	Premium18 USD	Enterprise 49 USD
Basecamp	30 dní	neomezeně uživatelů/měsíc	99 USD za měsíc neomezeně uživatelů			
Budgeta	30 dní	neomezeně uživatelů/měsíc	Basic 0 USD	Advanced 75 USD	Pro 200 USD	Enterprise 500 USD
TMetric	30 dní	uživatel/měsíc	Free 0 USD	Professional 4/5 USD	Business 6/7 USD	

Aplikace	Desktop aplikace			Doplňky do prohlížeče	Mobilní aplikace		API	Měření času	Reporting
	Windows	Mac	Linux		Android	iOS			
Costlocker	✓	✓		✓			✓	✓	✓
Toggl	✓	✓	✓	✓	✓	✓	✓	✓	✓
Basecamp	✓	✓		✓	✓	✓	✓	✓	✓
Budgeta							✓		✓
TMetric	✓	✓	✓	✓	✓	✓	✓	✓	✓

Příloha C **Architektura přeprodeje služeb**



Příloha D **Přepis rozhovoru pro získání zpětné vazby**

[Petr Žďárský] Takže tohle je rozhovor na získání zpětné vazby aplikace Companeero. Je tu se mnou Martin Pultzner, který bude zastupovat firmu Prague Labs s.r.o.; jen chci, Martine, dodat, jestli souhlasíš s nahráváním…?

[Martin Pultzner] Ano, souhlasím. Dobrý den.

[Petr Žďárský] Asi je potřeba, abychom si nějak prošli tu aplikaci a zhodnotili si výsledek toho, jak to celé dopadlo. Když začneme od systémové administrace, tak vlastně na úplně úvodní stránce máme přehled těch agentur, které jsou subscribeované v systému, a ty jednotlivé agentury se dají editovat, jak můžeme vidět při našem procházení…

[Martin Pultzner] … I ikonka se tam dá měnit, to bylo super.

[Petr Žďárský] Přesně tak.

[Martin Pultzner] Je tam vždycky… co se týče DPH (VAT rate), tak to tak dát třeba nějakým procentem. Jestli by to šlo.

[Petr Žďárský] Jo, to by určitě šlo. Chápu, že zadávat to v desetinným místě není úplně uživatelsky přívětivé.

[Martin Pultzner] Ale jinak, a to je vlastně to samé, je to potom u lidí, to znamená u klientů a u employees, jestli případně mají VAT rate, ale jinak asi super.

[Petr Žďárský] OK. Když se přesuneme dál k subscriptions, což je vlastně předplatné těch agentur, tak – jak jsme si už definovali – tak momentálně tam není žádná definice business modelu, a tudíž tahle sekce je pouze připravená a umožňuje jen definovat, že nějaká ta agentura má nějaké to předplatné… takže je to pouze hrubý nástřel…

[Martin Pultzner] Jasně… od-do, objednáno…. To vypadalo v pohodě… Asi nic moc dalšího se teď momentálně nedá ani vymyslet. Já si myslím, že to splňuje asi perfektně to, co má.

[Petr Žďárský] Když se potom přesuneme dál, tak máme vlastně přehled uživatelů v celé aplikaci – to jsou systémoví administrátoři. Opět jak vidíme, tak je to jednoduchá tabulka s možnostmi vytvoření uživatelů a definování nějakých metainformací, jako jsou uživatelská jména a fotky… Tak nevím, jestli je tady potřeba k tomu něco dodávat…?

[Martin Pultzner] Asi ne… Jenom tady, když to vidím, tak možná takový feedback na (a to se asi prolíná celým tím systémem, je určitě hrozně jednoduchý to udělat), že já teda vždycky ve všech systémech mám tendenci, když vidím jméno někoho nebo jméno společnosti, jméno uživatele a tak podobně, tak rovnou na něj jakoby kliknout, případně kliknu na tu ikonku – a tady to jakoby nejde a vždycky si musím nějak vpravo vybrat tu akci. Chápu, proč to je, protože těch akcí je někdy víc, ale asi by bylo dobré k tomu vždycky přiřadit nějakou defaultní akci, ideálně asi vždycky to nějak editovat, nebo se dostat do toho editačního pohledu tak, aby člověk mohl vždycky kliknout na tu největší část toho… ono to i pro ten touch… pro ten dotykový display je super, že si potom vždycky kliknu jenom na toho člověka a hned mi to otevře ten pohled. A to je vlastně všude… Ale to je asi jednoduché změnit, předpokládám…?

[Petr Žďárský] Jo, to určitě nebude problém. Potom když se přesuneme dál, tak to už jsou opět části systému, které jsou spíš pro to budoucí použití… jako jsou definice šablon notifikací, které systém bude odesílat. Momentálně ty notifikace v systému umožní definovat a jsou tam i definované vlastně nějaké variables, které automaticky ten systém plní. Ale jelikož momentálně ještě nemáme nikoho naboardovaného, kdo by to používal, a ten systém by něco odesílal, tak je to teď pouze připravené.

[Martin Pultzner] Jasně, jo, to vypadá v pořádku.

[Petr Žďárský] Je tady vlastně i ta další tabulka těch odeslaných notifikací – jelikož ještě nic neodesíláme, tak momentálně žádná data z databáze nenačítá… Další částí té systémové administrace – tady máme nastavení aplikace, kde si konkrétně můžeme definovat dynamické texty, které pak budou sloužit k překladu celé aplikace v případě, že by aplikace měla někdy takovéhle ambice, takže opět jsou to jen nějaké jednoduché tabulky a formuláře, kde se vybere jazyk a ten text se vytvoří a později referencuje v samotné aplikaci. Poslední tabulkou v systémové administraci je…

[Martin Pultzner] Mně se tam líbil ten editor, ten je super.

[Petr Žďárský] Tak to jsem rád (smích). Poslední tabulkou v té systémové administraci mám vlastně systémové nastavení, ve kterém můžeme definovat různé proměnné systému, jako je třeba adresa obrázkového serveru anebo výchozí noreply adresu, ze které budou odesílané vlastně ty notifikace. Opět je tam formulář, jak můžeme vidět, a proto se tam vždycky definuje nějaká hodnota nebo ji je možné editovat.

[Martin Pultzner] Jo, to vypadá jasně a srozumitelně… To tam ještě můžou zadat fotku, že jo?

[Petr Žďárský] Jo, tu fotku si můžeš upravit, když si vlezeš na svůj profil…

[Martin Pultzner] Vlastně i odsud se dá dostat…

[Petr Žďárský] Přesně tak a tam vlastně vybereš obrázek. Když se přesuneme do samotného dashboardu aplikace, což je teda ta část, kde to budou používat ty agentury, tak bych asi začal tím, že aplikace umožňuje si definovat nějaké services, což je nějaký jednoduchý ceník služeb, který ta agentura nabízí. A tady u těch služeb je možný definovat, jestli tu služby jsou billované jako man day, man hour, nebo mají fixní sazbu.

[Martin Pultzner] Jo, o tomhle jsme se bavili i s kolegama, že tohle je hodně důležité, že tam systém umí pracovat jak s man days, tak s hodinovýma sazbama, protože ono se to hodně míchá mezi klientama… a taky občas pracovníkama, takže tohle je důležité. A taky je tam možnost mít ty services v různých měnách, což je pro nás taky hrozně důležité, takže to určitě je jedna z nejdůležitějších součástí, a to vypadá dobře.

[Petr Žďárský] Tak to jsem rád. Když se teď přesuneme o kousek dál, konkrétně bych se teď podíval na HR té samotné aplikace, což je vlastně správa kandidátů a jejich zdrojů. Když to vezmeme nejdřív od těch zdrojů, tak vlastně ten systém nám tady umožňuje definovat nějaké ty HR zdroje, jako jsou třeba portály Jobs nebo StartupJobs, a pod tím je vidět nějaká metrika toho, jak jsou jednotlivé zdroje úspěšné, a díky tomu je vidět, na jaké zdroje se třeba soustředit, které platformy fungují.

[Martin Pultzner] Jo, to se mi hrozně líbí, ten graf tam, za to velká pochvala – já mám grafy rád a vím, že Pavel, co mi říkal, tak Pavel má taky grafy rád, takže to je… to, že tam je vidět, kolik těch kandidátů přišlo z jakého zdroje, tak to je super. Víc takových grafíků (smích)!

[Petr Žďárský] Pak jen k těm samotným kandidátům, jak jsme se už dřív bavili… Vlastně ti jednotliví kandidáti můžou mít tagy, takže je možné je tagovat, jaké mají schopnosti, jestli to jsou backenďáci, frontenďáci nebo jestli mají zájem o full-time nebo externě a tak dále…

[Martin Pultzner] Tady se jenom zeptám, mně to nebylo úplně jasné – ty tagy se berou/generujou tak, že když tam zadám nějaký text, který ještě není… jako dosavadní tag, tak se založí potom jako nový tag, že ho můžu používat, nebo je někde nějaká jednotná správa tagu?

[Petr Žďárský] To zní jako skvělá možnost, nad kterou jsem neuvažoval, ale ta správa tagů tam je, když klikneš na Aplikace a Settings, tak tady je tabulka, kde si můžeš definovat ty nové tagy.

[Martin Pultzner] Aha! OK, jasně, rozumím. Aspoň v tom nebude nepořádek, to si myslím, že stejný přístup máme i v administraci v jiné společnosti, takže to chápu. A k těm employees… k těm kandidátům, můžu ještě jeden feedback? O tom jsme se bavili s Honzou a vlastně my teď pro hireovací proces používáme Trello, a tam je to vlastně stylem karet, které se přesouvají z jednoho sloupečku do dalšího podle toho, v jakém stavu ten kandidát je, a vlastně tady zatím jsou dva stavy – s těm stavama tam už počítáš a vlastně tam je buď není hired, anebo je hired. Tak jestli by šlo (a bylo by to podle mě hodně důležité v tom celém HR procesu) mít přehled, v jaké fázi toho onboarding procesu nebo toho hireovacího procesu se nachází. Jenom typicky – třeba tam máme první sloupeček „lidi k pozvání na pohovor“, to znamená nám chodí životopisy – třeba deset, patnáct za týden – a ty si štosujeme v tom sloupečku „pozvat na pohovor“. A jednou za týden nebo za dva týdny se na to někdo podívá a protřídí to na ty relevantní/nerelevantní, a ti, co jsou relevantní, se přesunou „pozvat na pohovor“. Ti, co jsou pozvaní a přijdou, jsou v prvním kole, pak je druhé kolo a pak je třeba „odmítnul“ nebo „jsme ho chtěli a odmítnul“ nebo „on chtěl, ale my jsme ho nechtěli“ atp. To znamená, že potom se to dá i krásně filtrovat v budoucnosti a mít i nějaký HR zdroj, že si vyfiltruju v podstatě jenom ty, co jsou třeba… co jsme odmítli, co jsme neodmítli, chtěli jsme je na hireovat, ale odmítli třeba nás – tak je znovu oslovit. Ale jakoby mít potom i v tomhletom nějaký přehled, v jaké fázi toho procesu je… Takže to v nějakých budoucích funkcích. Teď to úplně stačí, ale aby to mohlo fungovat potom úplně přesně, třeba jako to máme teďka

nastavený – to by bylo úplně perfektní, kdyby tam těch příznaků „hired“, „not hired“ bylo víc, třeba šest, sedm, ale to asi taky není problém…?

[Petr Žďárský] To není problém. A mělo by to spíš být jako tabulka, nebo jako kanban styl jako v Trellu?

[Martin Pultzner] Mně je to asi víceméně jedno. Honzovi se líbí kanban styl… Tady, kdyby to „not hired“ nebo ten stav byl třeba klikatelný a že bych si v tomhle seznamu rychle mohl vybírat, že „byl pozván“, a nemusel bych vlastně nějakým způsobem otevírat profily těch lidí, tak by to bylo úplně super… Pokud to jde…?

[Petr Žďárský] Jasně, to určitě půjde zařídit.

[Martin Pultzner] Paráda (smích). Doufám, že toho feedbacku není moc?

[Petr Žďárský] Ne, každý feedback je dobrý (smích). Když se přesuneme k empoyees, co jsou zaměstnanci, kteří pracují ve firmě, tak opět se tady dívám na nějakou tabulku, kde máme metainformace (kontaktní údaje, jméno, fotku), a potom, když se přesuneme na nějaký detail toho zaměstnance, tak jako první, co vidíme, je tabulka „Services“, což, jak jsme si již dřív při definici toho projektu bavili, je, že by bylo dobré, aby každý ten zaměstnanec mohl mít několik služeb, které mají různé sazby. Takže tady je možné (opět je to stejný model jako u toho ceníku té celé agentury), že každý zaměstnanec má vlastní ceník služeb, který se potom váže na ty jednotlivé práce k těm projektům.

[Martin Pultzner] Jo, to jsem zkoušel a super, že se to dá přidávat snadno, hvězdičkovat dokonce, to je super.

[Petr Žďárský] A potom jednoduché grafíky, vlastně měsíční zobrazení, jak ten uživatel kolik měří času, kolik mu bylo vyplaceno. A taky přehled těch posledních aktivit, které naměřil v time trackeru.

[Martin Pultzner] Jasně, tam to všechno vypadalo v pohodě.

[Petr Žďárský] A když to teď ještě vezmeme z té druhé strany – začneme klientama. Tak opět je to nějaká definice firem, kde se můžou definovat standardní informace jako adresa, účetní informace jako VAT rate, DIČ a různé další info.

[Martin Pultzner] A tady jsem k tomu… jo, jo, OK. Ještě teda jedna věc k těm zaměstnancům, jestli můžu – když jsem tam spravoval nějaké věci a lidi, tak potom jsem narazil na to, že tu sekci „Billing“ nemůžu editovat.

[Petr Žďárský] To je pravda. Tohleto vlastně editovalné není kvůli tomu, že v tom současném modelu tak, jak to bylo navržené, bychom si tyhle informace museli ukládat, že jsou platné v nějakém čase, což momentálně ten systém neumí, a mohlo by se stát, že kdyby někdo měsíc měřil nějaký čas s nějakou sazbu s nějakým nastavením DPH a my bychom to v tom průběhu změnili, pak by vlastně došlo k vybillování, tak by se najednou mohlo stát, že místo toho, abychom vyúčtovali tisíc korun, bychom vyúčtovali tisíc euro.

[Martin Pultzner] Jasně, jo, jo. To znamená, že by se tam muselo udělat něco, že jakákoli změna bude až od nového měsíce nebo nějaká taková věc.

[Petr Žďárský] Třeba. Anebo, když by tohle byl jako požadavek, že je to potřeba měnit, tak to můžeme posunout z té úrovně toho employee na tu jednotlivou službu.

[Martin Pultzner] Jo, ono většinou to není potřeba měnit, ale napadá mě typicky to, co se nám teď stalo, že jeden člověk (externista), který fakturuje, předtím nebyl plátce DPH, tak se teď plátcem DPH stal od prvního, to znamená, že by se to muselo měnit třeba od prvního v měsíci, že je registrován v DPH, anebo třeba… a je to, myslím, to samé… Nic ostatního asi není potřeba měnit, ale jinak ta registrace k DPH, no. Případně pokud se vláda naštve a najednou nám zvýší DPH nebo sníží, tak se to vlastně u všech bude muset změnit.

[Petr Žďárský] Jo, v tom případě myslím, že by dávalo smysl to posunout na tu úroveň té jednotlivé služby toho zaměstnance, protože ta je závislá v čase, takže vlastně by to ten zaměstnanec od nového měsíce trackoval pod jinou svoji službu.

[Martin Pultzner] Jasno, jo, OK, to by asi šlo. A ještě druhá věc: k té adrese, která tam teď už je v profilu toho uživatele, tak ještě přidat billing adresu – ona může být odlišná od adresy trvalého bydliště, kam třeba může zasílat nějaké písemnosti, ale billing adresa, na kterou se vystavují faktury, ta může být jiná… že by to mohlo být takhle odlišené, vlastně tam přidat to samé, co je v poli „Address“, tak přidat ještě „Billing Address“.

[Petr Žďárský] Jo, to určitě není problém.

[Martin Pultzner] OK, jo, to je všechno. Jinak opravdu tam těch informací je spousta a všechno ostatní fungovalo, je to super.

[Petr Žďárský] A předpokládám, že to samé je i pro klienty, na které koukáme, že můžou mít jiné sídlo společnosti, než je billing adresa.

[Martin Pultzner] Přesně tak, to se stává hodně často, to tady mám taky.

[Petr Žďárský] Když se teď posuneme k projektům… jen co se to načte… U náhledu projektů je to opět tabulka, kde jsou vypsané všechny aktuálně běžící projekty, a po kliknutí na detail tady vidíme nějaké rychlé informace, jaký je aktuální balanc té společnosti, která si ten projekt zadala, metainformace, v jakém je to stavu a odkdy do kdy ten projekt běží, a také nějaké účetní informace ohledně toho, kolik už bylo klientovi vybillováno, jaké jsou náklady a jaká je celá revenue toho projektu.

[Martin Pultzner] Jasně, graf – taky super (smích).

[Petr Žďárský] No a co je tady vlastně možné, tak každý ten projekt má definované nějaké svoje worky, jak tomu říkáme, a ty jsou založené buď na šablonách toho ceníku, anebo je to definované vždycky pro ten vlastní projekt. A opět, jak můžeme vidět, je to vždycky definované za nějakou částku a potom typ billingu, jestli je to man day, man hour nebo nějaká fixní sazba… Vlastně na tyhle služby, na tyhle worky se potom vážou ty služby těch jednotlivých zaměstnanců, což nám tím pádem dochází k mapování jedna ku jedné, a vím, že tohle Honza trochu oponoval, že by ho nebavilo potom, když přijmeme nového zaměstnance, že by musel naklikat všechny ty vazby, že tenhle člověk může trackovat na tyhle worky.

[Martin Pultzner] Jo, že říkal, že by měl mít možnost trackovat na všechno.

[Petr Žďárský] Ale tam jsme, když se to navrhovalo, tak jsme nikdy nedošly k nějakému závěru. Vlastně druhá oponentura byla, že by bylo dobré mít tohle limitované, že ne každý může měřit pro všechno.

[Martin Pultzner] Jasně, určitě každý z těch… V ideálním světě by to měl být ten druhý přístup, ten, co je tam teď implementovaný, to znamená, že člověk nemůže měřit a

ani se mu nenabídne měření v projektu, ke kterému nemá přístup, aby se nestávaly chyby. Takže když máš možnost billovat jenom dva, tři projekty, tak si nemusíš vybírat z nějakého dlouhého seznamu, a to mi přijde dobré. Samozřejmě to má konotace v tom, že ten manager musí ještě předtím, než se začne billovat, tohle všechno nastavit. Ale nevím, jestli to jde udělat… Je pravda, že Honza to vždycky nastavuje. Nevím, jestli to jde nějakým způsobem provázat tak, aby, když ten člověk začne na tom projektu billovat, tak se pošle nějaká konfirmace, že „Skutečně chcete billovat na tomhle projektu, ke kterému nemáte přístup?“ nebo něco podobného, aby v případě nutnosti, že ten manager to nemůže nějak assignovat, aby ten člověk mohl i tak billovat… Ale to si myslím, že už je příliš složité, myslím, že bych to takhle asi nechal, to mi přijde lepší kontrola. Prostě to závisí opravdu na tom managerovi, komu dá přístup. Tohle mi přijde asi lepší než otevřít všem všechno, pak by z toho mohly vznikat chyby, které vznikají (teď co používáme ten trackovací software, tak tam občas lidi natrackujou do něčeho špatného). Takže asi bych to takhle nechal, to mi přijde jako dobrý přístup.

[Petr Žďárský] OK. Když budeme koukat dál, tak tady máme i nějaký grafík toho, kolik bylo účtováno tomu klientovi a jaké byly náklady na tohoto klienta, což vlastně odráží ty platby zaměstnanců či externí výdaje… což ty výdaje tam jdou přidat i manuálně, jako kdyby to byly nějaké jednorázové věci jako pořízení hostingu nebo něco takového.

[Martin Pultzner] Jo, to vím, že jsme probírali minule, a to je důležité. Tahle část vypadá hodně dobře, že to je použitelné. A vlastně všechno, co by to mělo umět, tak to umí… včetně těch grafů (smích), což je dobře.

[Petr Žďárský] Když půjdeme na další funkcionalitu, což je, jak už jsi zmiňoval, time tracker, kde docházelo k integraci téhle vybudované aplikace s tím současným řešením, co se používá. Jen to tady otevřu… Je to opět nějaká tabulka, která každých třicet vteřin… tenhle systém je aktualizovaný tím současným řešením, takže ta data jsou skoro absolutně aktuální, což si můžeme i teď vyzkoušet, že když já začnu něco měřit, tak se nám to tu za chvíli objeví.

[Martin Pultzner] Jo, to jsem si i zkoušel a objevovalo se to hezky.

[Petr Žďárský] Což mě pak přivádí k další funkcionalitě, a to že ten model, jak je tady nastavený, je, že vždycky ten časový záznam, který ten zaměstnanec vyměří, tak je vlastně zdvojený. A je to jeden čas, který je účtovaný klientovi, a jeden čas, který je účtovaný zaměstnancem. A tohle je možné rozdělit potom, tyhle dva časy rozvázat, že třeba klientovi se to upraví na míň nebo na víc, případně se to dá nastavit, jakože tenhle čas vůbec není proplacený.

[Martin Pultzner] No, tohle je velice častá věc, to i s Honzou a hlavně s Pavlem, kdy neustále měníme sazby pro jednoho klienta… nebo spíš billovaný čas, tak to je důležité, to je super, že to tam je.

[Petr Žďárský] Tak, my se teď přesuneme k tomu billingu, který je rozdělený na dvě části, a to je billing zaměstnanců a billing klientů. Mám tady takovou tabulku, kde se vždycky zobrazují už aktuálně vybillované časy jednomu nebo druhému. Pomocí tlačítka můžeme jednoduše vygenerovat nějaký report, který odráží, kolik se má tomu člověku proplatit, v případě klienta mu nafakturovat za ten čas, který je v tom systému naměřený. To, co je u těch reportů možné, je výpis těch aktivit, jako jaká to byla doba a hodnota, a pak je vlastně jedním tlačítkem nastavit, že tahle částka byla proplacena zaměstnanci nebo od klienta.

[Martin Pultzner] Jo, samozřejmě než bude ta faktura tak komplexní jako třeba v tom iDokladu, který používáme, tak to bude chvilku trvat, ale ta základní funkcionalita tam je, takže použitelné to určitě může být. Akorát na ty faktury je potom potřeba doplnit ty fakturační údaje, co se týče IČa, DIČa toho, kdo to vystavuje – Prague Labs jakoby. Že teď je tam vlastně adresa a další údaje tam jsou v pořádku; jenom aby se tam propisovalo i to ID, ale to možná není, protože to není v profilu vyplněné… je možné?

[Petr Žďárský] Na těch fakturách? To je dobrá otázka, to bych se musel podívat, proč se to tam nezobrazuje.

[Martin Pultzner] Tady vpravo nahoře by to mělo být… IČO a DIČ.

[Petr Žďárský] To určitě dává smysl… co mě ještě napadá – vím, že jsme se kdysi bavili o tom, jak přiidávat bonusy zaměstnancům, což ten současný systém nepokrývá, jak jsme se už kdysi bavili, protože my momentálně tam můžeme zadat ten bonus, který

jde přímo z účtu Prague Labs, ale to se pak nedělí mezi ty projekty, případně klienty. Takže to je asi věc, na které taky můžeme v budoucnu zapracovat.

[Martin Pultzner] Jo, teď se to vlastně dělá manuálně, že účetní to rozpočítává poměrově podle počtu hodin, které jaký zaměstnanec trávil na kterém projektu, tak ten bonus rozpočítává podle toho. Takže teď je to supermanuální, pokud se to tímhle zautomatizuje, tak tím líp.

[Petr Žďárský] Jo, myslím, že je určitě reálné to do toho systému doplnit… No, tak to byl celý průchod té aplikace. Nevím, jestli máš na svém listu ještě něco, co jsme neprobrali…?

[Martin Pultzner] Já se podívám, co tady… Jo, tady jsem měl poznámku, že aby se ukázaly ty expenses u daného projektu i v tom grafíku a byly zaúčtované – je potřeba je nějak zaúčtovat, což jsou dva pohledy. Jeden je, jestli to neukazovat ještě před zaúčtováním, abychom věděli, jak jsme na tom na projektu. Ale přijde mi možná, že ten přístup, co tady je, je správnější, že nejdřív to musíš jakoby schválit, zaúčtovat a pak se to tam objeví. To znamená, když to bude schvalovat někdo jednou měsíčně společně s tím, jak se dělají odměny, tak to si myslím, že je dostatečné. Asi je to jenom o přístupu, stejně jako v nastavování těch projektů, takže si myslím, že tohle je asi v pohodě.

[Petr Žďárský] Jako teoreticky bychom tam mohli třeba přidat nějaké procesy, že ten zodpovědný manager bude jednou denně notifikován, aby prošel ty včerejší záznamy a…

[Martin Pultzner] Jo, jednou denně by bylo asi docela… nereálné (smích), ale tak jednou týdně nebo jednou za dva týdny si myslím, že to reálné je. Ale jinak co tady všechno máme, tak… tak tady mám, že jsme prošli. Jediné, co ještě u employees… teda u candidates – ještě tak v profilu toho kandidáta by bylo užitečné přidat tam nějakou možnost přidávat životopis. Buď nahrát přímo do systému, anebo tam postnoust aktivní link třeba na Google Drive. Teď na tom Trellu se to dává přímo do té dané karty v Trellu, že si to pak člověk nemusí hledat. Pak vlastně když máme s tím kandidátem pohovor, tak vždycky dáváme link na ten konkrétní životopis a na ten konkrétní profil na tu danou kartu, takže kdyby šlo dělat něco podobného, tak by to bylo super, že bychom ten životopis mohli nahrát přímo do toho Companeera.

[Petr Žďárský] Jo, to je určitě validní připomínka. Vím, že uložiště souborů bylo i jednou z funkcí, o kterých jsme uvažovali, ale bohužel tam teď momentálně není.

[Martin Pultzner] Zatím to můžeme udělat tak, že by se to dalo jako…

[Petr Žďárský] Můžeš si to přidávat do té poznámky, a ta se potom objeví v tom logu, který je pod tím uživatelem.

[Martin Pultzner] Jasně, tak v tom případě to je z mých připomínek všechno, jinak všechno ostatní funguje velice dobře, musím říct.

[Petr Žďárský] Tak to jsem rád. Tak jo, já myslím, že to je pro naše hodnocení toho, jak ta aplikace dopadla, asi všechno, a já Ti děkuju za rozhovor.

[Martin Pultzner] Taky děkuju.