VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

PLÁNOVACÍ MODUL PRO KANBOARD.ORG

PLANNING MODULE FOR KANBOARD.ORG

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **JIŘÍ HOLUBÁŘ**
AUTHOR

VEDOUCÍ PRÁCE **Ing. FRANTIŠEK ŠČUGLÍK, Ph.D.**
SUPERVISOR

BRNO 2019

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2018/2019

Zadání bakalářské práce

Student: **Holubář Jiří**
Program: Informační technologie
Název: **Plánovací modul pro Kanboard.org**
Time Scheduling Plugin for Kanboard.org
Kategorie: Web

Zadání:

1. Seznamte se s nástrojem Kanboard.org a s technologiemi PHP a SQLite.
2. Nastudujte možnosti tvorby vestavných modulů pro Kanboard.org.
3. Navrhněte modul pro vizualizaci přidělených úkolů jednotlivým uživatelům. Ve vizualizaci uvažujte zobrazení na zvolené časové ose s možností přiřazování nových úkolů.
4. Po konzultaci s vedoucím navržené řešení implementujte a otestujte. Dosažené výsledky zhodnoťte.

Literatura:

- Dokumentace Kanboard.org [online], Dostupné z: https://docs.kanboard.org/en/latest/
- Welling, L., Thomson, L.: PHP a MySQL Kompletní průvodce vývojáře, CPress, 2017
- Kroenke, D., M., Auer, D. J.: Databáze, CPress, 2014

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Ščuglík František, Ing., Ph.D.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 15. května 2019
Datum schválení: 30. října 2018

Abstrakt

V dnešní době se ve firmách často musí úkoly rozdělit mezi více lidí. Zároveň jeden člověk většinou pracuje na více úkolech současně. Proto je třeba rozumně rozvrhnout práci každého člověka pro dosažení co největší efektivity. K tomu by měl sloužit modul pro Kanboard, kterému se věnuji v této práci.

Abstract

Nowadays, in most corporations must be tasks split between more than one person. Simultaneously one person work on more tasks at the same time. That is the reason, why people must plan their work, to achieve maximum efficiency. To do that, the module for Kanboard, which I describe in this work, should be used.

Klíčová slova

Kanboard, plugin, PHP, HTML, CSS, plánování času, OOP, javascript

Keywords

Kanboard, plugin, PHP, HTML CSS, time management, OOP, javascript

Citace

HOLUBÁŘ, Jiří. *Plánovací modul pro Kanboard.org*. Brno, 2019. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. František Ščuglík, Ph.D.

Plánovací modul pro Kanboard.org

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Františka Ščuglíka Ph.D.. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jiří Holubář
14. května 2019

Poděkování

Na tomto místě bych rád poděkoval Ing. Františkovi Ščuglíkovi Ph.D. za ochotu a čas, který mi při vedení mé práce věnoval.

Obsah

Kapitola 1

Úvod

Při práci na větších projektech je často velmi důležité správně si rozvrhnout čas, aby jsme byli schopni dodržet deadline. Především při práci na více projektech současně je důležité hlídat si během dne čas strávený na konkrétním projektu. Příliš mnoho času stráveného nad jedním úkolem by mohlo zapříčinit nedostatek času na úkoly u ostatních projektů.

Vzhledem k tomu, že ve firmách často na jednom projektu pracuje více lidí a tito lidé navíc pracují na více projektech současně, bylo by téměř nemožné pro vedoucí týmů rozvrhnout práci efektivně tak, aby se vše stíhalo jak má. A to je chvíle, kdy je vhodné využít nějaký nástroj pro plánování práce.

Většina těchto nástrojů umožňuje vytvořit projekty, rozdělit je na úkoly a tyto úkoly přiřadit jednotlivým lidem. Často se ale stává, že pro potřeby firmy je třeba vytvořit rozšíření těchto nástrojů, aby vyhovovaly konkrétním požadavkům.

V této práci se budu věnovat tvorbě takového modulu, a to konkrétně pro nástroj Kanboard.

Kapitola 2

Kanban

Při vývoji softwaru je často vhodné využít nějaký model procesu vývoje, neboť bez organizace práce by mohl vzniknout chaos, který by měl negativní vliv na výsledný produkt. Ať už se rozhodneme pro kterýkoliv model procesu, je dobré jej nějakým způsobem zefektivnit. A právě pro toto zefektivnění se dá použít metoda zvaná Kanban.

Ta spočívá v tom, že si projekt rozdělíme na menší části neboli úkoly. Ty poté mohou reprezentovat štítky, ať už v reálné podobě nebo elektronické. Tyto štítky se pak umístí na Kanban tabuli, která je rozdělená na sloupce, které reprezentují stav, v jakém je určitý štítek. Díky tomu pak můžeme vidět, zda není příliš mnoho úkolů v některém z těchto stavů a pokud ano, může se například přidělit více lidí na odstranění tohoto stavu. Například může příliš mnoho úkolů čekat na specifikaci a lidé, kteří jsou za ni zodpovědní budou vědět, že je třeba se více soustředit právě na tyto úkoly.

Hlavní výhodou na tomto přístupu je, že díky rozdělení práce se mohou členové týmů soustředit pouze na jednu věc a nepracovat na více věcech najednou. Zároveň mají všichni členové týmu souhrnný přehled o tom, v jakém stavu jsou práce na projektu. Tato vizualizace může být velice užitečná například na týmových poradách, kde se takto dá dobře sledovat pokračování prací na projektu od posledního setkání. Je také dobré sledovat, jak dlouho trvá průměrně u úkolů, než se dostanou z jednoho stavu do nějakého jiného. V případě, že je tato doba příliš dlouhá, je dobré analyzovat, proč došlo k této prodlevě a tomuto problému se například vyvarovat u jiných projektů.

Každý úkol by měl mít také předem určený čas, který na něm může pracovník strávit. Zamezíme tak zaseknutí prací na úkolech, které by neměly vyžadovat více času, než je odhad na jejich realizaci. To nám umožní odhalit některé problémy již při vývoji.

Pro správné rozvržení práce je dobré mít také přehled o tom, který pracovník se věnuje kterému úkolu a kolik času na něm stráví. A právě za účelem této vizualizace vznikl modul, kterému se věnuji v této práci. Je mnohem snazší plánovat práci, když můžeme určit, že se určitý člověk bude denně věnovat určitému úkolu například jen tři hodiny a díky tomu stihne dokončit například i jiné menší úkoly.

Kapitola 3

Kanboard

Kanboard je bezplatný open source program pro správu projektů[4]. Vyznačuje se především svou jednoduchostí. Jeho uživatelské rozhraní není příliš hezké, ale díky tomu se v něm dá efektivně pracovat.

Díky jeho zaměření na jednoduchost a minimalizmus se ovšem může stát, že jeho funkčnost nevyhovuje potřebám uživatele. To se dá vyřešit instalací modulů, které rozšíří funkčnost Kanboadu. Spoustu těchto modulů se dá najít na hlavní stránce Kanboardu. A i když uživatel nenajde modul, který by mu vyhovoval, má vždy možnost si napsat vlastní.

Kanboard také nabízí více překladů, mezi nimiž je i čeština. Na to je dobré myslet například právě třeba při vývoji modulů, kde by se většina textů měla vypisovat pomocí překladové funkce.

3.1 Instalace a požadavky

Pro fungování serverové části Kanboardu je třeba mít vlastní hosting s verzí PHP 5.6.0 nebo vyšší[5]. Doporučuje je ovšem využívat nejnovější verzi PHP s co možná nejmodernější Linuxovým nebo Unixovým operačním systémem.

Kompatibilní serverové operační systémy jsou[5]:

- Alpine Linux (3.8 nebo vyšší)
- Linux Ubuntu (16.04 nebo vyšší)
- Linux Centos (7.x)
- Linux Redhat (7.x)
- Linux Debian (9)
- FreeBSD (10.x)
- Microsoft Windows 2016
- Microsoft Windows 2012 R2

Kanboard také používá databázi. Zde se doporučuje využít Sqlite v případě, že bude Kanboard využívaný pro menší projekty v menších týmech. Pro větší projekty, na kterých se podílí více lidí se doporučuje využít Mysql nebo Postgres.

Pro využívání Kanboardu ze strany klienta je třeba pouze internetový prohlížeč. Nejvíce se doporučuje využívat Google Chrome nebo Mozillu Firefox.

Instalace je poté velice jednoduchá, Stačí stáhnout zdrojový kód Kanboardu a nakopírovat jej přímo na server nebo ho naklonovat přímo z repozitáře na gitu. Poté už se stačí pouze přihlásit pod defaultním uživatelem.

3.2 Uživatelé

Jsou tři základní skupiny práv pro uživatele:

- Administrator
- Manager
- User

Administrátor má přístup ke všemu, co aplikace nabízí. Dalším typem uživatele je Manager, který může spravovat projekty a úkoly, ale už nemá přístup do nastavení aplikace. A posledním typem uživatele je User, který může vytvářet pouze soukromé projekty.

V rámci projektu poté můžou mít uživatelé tyto práva[6]:

- Projekt Manager
- Projekt Member
- Project Viewer

Project Manager může měnit nastavení projektu a má také přístup ke statistikám a reportům. Project Member může vytvářet úkoly a měnit jejich stav. A poslední skupinou je Project Viewer, který může úkoly pouze vidět, ale nemůže u nich nic měnit.

Administrátoři mají také možnost vytvářet skupiny a do nich přiřazovat jednotlivé uživatele. Jeden uživatel může být členem jedné nebo více skupin. To poté usnadní práci při udělování práv k projektům, protože Project Manager poté nemusí nastavovat práva na projekt jednotlivým lidem, ale může je nastavit celé skupině.

3.3 Projekty

Projekty se dělí na soukromé a týmové. Soukromý projekt si může vytvořit každý uživatel a nemá do něj přístup nikdo jiný kromě něj. Týmové projekty mohou vytvářet pouze Administrátoři a Manažeři. U těchto projektů už lze nastavovat práva pro jednotlivé uživatele nebo skupiny.

Ke každému projektu se dají take přiřadit swimlanes. To jsou v podstatě fáze práce na úkolech. Každý projekt má po vytvoření nastavenou defaultní swimlane, která v kanboardu obsahuje čtyři fáze.

Defaultními fázemi jsou:

- Backlog (fronta)
- Ready (připravené na zpracování)
- Work in progress (úkoly, na kterých se pracuje)

- Done (vyřízené)

Mezi těmito fázemi může uživatel libovolně přetahovat úkoly, která jsou u projektu vytvořené. To usnadňuje práci s úkoly a umožňuje mít přehled o tom, na kterých úkolech by se například mělo začít pracovat.

Kanboard také umožňuje u každého projektu vytvořit vlastní skupiny práv, které lze poté přiřazovat určitým lidem, kteří se na projektu podílí. Lze takto tedy povolit některým uživatelům vytvářet nové úkoly, ale například pouze v sekci Backlog. Zároveň lze takto také omezit u uživatelů přesouvání úkolů mezi jednotlivými fázemi. Například někteří uživatelé budou moci přetáhnout úkol ze sekce Ready do sekce Work in progress, ale už je nebudou moci přetáhnout do sekce Done.

Velice užitečné je také zpřístupnění nástěnky projektu pro veřejnost. Tato možnost vygeneruje odkazy, díky kterým se dá zobrazit nástěnka projektu s úkoly. To je dá využít například, když chceme zákazníkovi ukázat, v jaké fázi je jeho projekt, aby si mohl udělat představu o pokračování prací. Tato možnost se dá u projektu kdykoliv zakázat a link s tímto náhledem se stane neaktivním.

Je zde také možnost zobrazit nejrůznější grafy pro konkrétní projekt. Tak můžeme přehledně vidět například na kolika úkolech se v daném projektu podílel konkrétní uživatel, nebo v kterém období se na projektu nejvíce pracovalo. To může často pomoct při analýze práce na projektech. Vedoucí týmů díky tomu mohou například lépe přerozdělovat práci, takže se na projektu pracuje pravidelněji a neřeší se všechny problémy na poslední chvíli.

3.4 Úkoly

Velice důležité při práci na projektu je rozdělit si ho na menší úkoly. To umožní nejen mít lepší přehled o tom, kolik práce ještě na projektu zbývá, ale také rozdělit práci mezi více lidí. Jednotlivé úkoly se dají totiž přiřazovat určitým uživatelům a tím usnadnit plánování práce.

Jednotlivým úkolům se dá také přiřadit, kolik času by se mělo strávit jeho vypracováním. To bývá důležitý údaj, protože zákazník je například ochoten zaplatit jen určitý čas. Právě kvůli rozvržení času, který je přiřazený k úkolu, vznikl plugin pro plánování času, kterým se zabývá tato práce.

K úkolům je také možné přiřazovat konkrétní štítky. Ty usnadňují práci, neboť podle nich například můžeme filtrovat seznam úkolů. Je možné vytvořit vlastní štítky. Ty se dají vytvářet pouze pro konkrétní projekt, nebo pro všechny projekty. Štítek se dá také vytvořit přímo ve formuláři editace úkolu[7].

K úkolu se dá také přidat popis, který může pomoci člověku, který na něm pracuje, pomoci v jeho řešení. Je zde možné například vložit snímek obrazovky, kde se vyskytl problém, kvůli kterému byl úkol vytvořen.

Kanboard umožňuje graficky v grafu zobrazit, jakou dobu byl úkol ve které fázi. To se může hodit především při analýze efektivity práce a můžeme tak například identifikovat úkoly, které už příliš dlouho čekají na zpracování.

Úkoly můžeme také rozdělit na jednotlivé podúkoly, což nám opět usnadní rozdělení práce v týmu a může vést k rychlejšímu vyřešení problému, kterým se daný úkol zabývá.

Důležité je také zaznamenávání času, který se strávil řešením daného úkolu. Pracovník tak může například zjistit, kolik času mu ještě zbývá na vyřešení daného problému zbývá. To bývá důležitý údaj především u úkolů, na které je například pouze omezený rozpočet

a zaměstnavateli by se nevyplatilo, kdyby na něm zaměstnanec pracoval delší dobu, než je určený čas.

3.5 Moduly pro Kanboard

Velkou výhodou Kanboardu je snadná implementace modulů, které rozšíří jeho funkčnost. Svědčí o tom také možnost stáhnout spoustu modulů přímo na stránkách Kanboardu. Ty jsou všechny zdarma a obvykle vyvíjené komunitou. Díky tomu si může uživatel upravit vzhled a funkčnost aplikace podle svých představ. A pokud nenajde žádný modul dle svých požadavků, muže si jej vždy naprogramovat sám.

Bohužel moduly nejsou nijak schvalovány a jejich kód není kontrolován. Je tedy na uživateli, aby si ověřil, že modul dělá opravdu to, co požaduje[3].

3.5.1 Instalace Modulů

Instalace modulů je velice snadná. Stačí si stáhnout modul z webu, nebo jej případně nakopírovat z gitu. Složku s modulem poté vložíme do složky plugins. Pokud je modul vytvořený správně, měl by se vypsat v seznamu modulů v nastavení Kanboardu. Pokud modul potřebuje instalaci nějakých dalších modulů nebo čehokoliv jiného, melo by to být napsáno v popisu modulu. Některé moduly také požadují určitou verzi Kanboardu. Pokud tedy nemáme nejnovější verzi, je třeba zkontrolovat ve výpise, zda máme minimálně stejnou nebo vyšší verzi Kanboardu, než jaká je požadovaná.

Pokud vytváříme nový modul, postará se o jeho registraci soubor Plugin.php, který je umístěný v adresáři modulu. V něm můžeme také nastavit například jméno autora nebo požadovanou verzi Kanboardu.

Kapitola 4

Používané technologie

Back-end modulu je psaný v PHP jako i back-end Kanboardu. Modul je napsaný pro verzi PHP 7.2. Front-end využívá HTML a css. Pro přetahovaní úkolů k jejich uživatelům a změnu přiřazených hodin je využitá javascriptová knihovna Interact.js.

4.1 PHP

PHP je netypovaný programovací jazyk, který pracuje na straně serveru. Díky tomu rychlost back-end části programu nezáleží na výkonu počítače uživatele, ale na výkonu serveru, na kterém je umístěný Kanboard. Tento jazyk je využívaný pro vývoj webových aplikací především proto, že ho jde vkládat do HTML[12]. Při vkládání do HTML skriptů většinou začíná značkou *<?php* a končí značkou *?>*. Toho se v Kanboardu využívá především v šablonách stránek například pro výpis dat, které odeslal back-end do šablony.

Další výhodou PHP je, že podporuje jak procedurální, tak objektově orientované programování. Právě díky objektově orientovanému programování (OOP) je možné v Kanboardu vytvářet moduly bez jakéhokoliv zásahu do původní aplikace. Díky tomu je instalace modulů velice snadná, kdy v podstatě stačí nový modul zkopírovat do příslušné složky. Vzhledem k tomu, že celý Kanboard je psaný objektově, lze ho využít jako takový jednodušší framework.

4.1.1 PHP 7.2

Nová verze PHP přinesla vylepšení, která usnadní programátorovi práci.

Jsou jimi například[11]:

- hash funkce umožňuje využít Argon2i algoritmus
- nová sodium rozšíření, která jsou nyní součástí jádra PHP
- možnost použít *object* jako typ parametru a návratové hodnoty
- dědění typů parametrů u poděděných tříd
- označení některých zastaralých funkcí jako Deprecated

4.1.2 Práce s databází

Jednou z mnoha výhod PHP je jeho schopnost pracovat s velkou škálou databází. K tomu se dají využít knihovny, které jsou vytvořené pro konkrétní typ databáze (například *mysql*)

nebo knihovny nabízející větší abstrakci, jako je například *PDO*. Také díky tomu je PHP vhodné pro psaní webových aplikací, které ke svému fungování vyžadují připojení k databázi, kam si ukládají svá data.

4.1.3 Frameworky

Díky svým možnostem, je PHP jeden z nejvyužívanějších jazyků pro psaní menších i rozsáhlých webových aplikací, mezi které patří například i Kanboard. Při psaní právě těch větších aplikací se může stát, že musí programátor pro většinu elementárních operací psát téměř stejný kód například jen s menšími obměnami. Příkladem takových operací může být například ukládaní dat do databáze, kde se může operace vždy lišit pouze v drobnostech.

To je důvod, proč vznikly frameworky. Nabízí programátorovi určitou abstrakci nad operacemi. Například zajišťují připojení k databázi a práci s ní. Nemusíme díky tomu tedy při každém dotazu na databázi psát dotaz, ale stačí nám zavolat metodu, které pouze předáme informace o tom, co chceme zjistit a odkud. Zbytek už za nás zařídí právě framework.

Dostupné frameworky jsou například:

- Symfony
- Laravel
- CodeIgniter
- Cake PHP
- Zend
- FuelPHP

Na to, který framework se rozhodneme využít, by měly mít vliv především naše požadavky. Každý framework totiž nabízí jiné funkce. Nemusí například podporovat připojení k druhu databáze, kterou chceme v naší aplikaci využívat. Každý framework může mít také rozdílnou časovou náročnost pro zpracování některých operací.

Někdy se může stát, že nám žádný framework z nějakého důvodu nevyhovuje. V takovém případě bývá nejlepší napsat si vlastní framework, který bude vyhovovat naším požadavkům. Psaní takového frameworku může být sice časově náročnější, ale velice nám to usnadní práci do budoucna. Touto cestou šel i Kanboard. Díky tomu se programátor při psaní modulu nemusí zatěžovat inicializací všech potřebných věcí, připojováním k databázi a podobně. Pro většinu operací, které potřebuje provádět může pohodlně využít již vytvořené metody.

4.1.4 Dependency injection

Dependency injection (DI) je technika využívaná v objektově orientovaném programování. Objekty často využívají ke svému chodu objekty jiných tříd. Ale tyto objekty jiných tříd mohou vyžadovat pro svoji inicializaci další data, objekty a podobně. V takovém případě bychom museli objektu všechny tyto prostředky poskytnout a nemuseli bychom mít jistotu, že mají ta správná data pro svoji funkci. Toto se dá pomocí DI obejít.

V podstatě se jedná o to, že objekty tříd, které poté budou využívat jiné objekty, inicializujeme na začátku programu a uložíme si je do DI containeru. Všechny potřebné závislosti jsou zajištěny již při vkládání do tohoto containeru a když pak chceme některý

z objektů využívat, tak si ho z něj pouze vytáhneme. Tím zajistíme, že všechny objekty využívající objekt v DI containeru, pracují se stejnými daty. Příkladem takového objektu může být například objekt třídy, která se stará o připojení k databázi. Máme díky tomu jistotu, že se připojujeme ke stejné databázi, ke které se připojují ostatní objekty využívající právě ten, který slouží k tomuto připojení.

Nevýhodou tohoto přístupu ovšem je, že kód se tím stane o něco méně přehledným. Na druhou stranu se ale dá mnohem lépe orientovat v závislostech jednotlivých objektů. V podstatě by se dalo říci, že se tím zmenší počet nově vytvářených objektů v metodách tříd.

4.1.5 Model-View-Controller

Nejčastěji využívaným návrhovým vzorem u webových aplikací je Model-View-Controller. Ten rozděluje aplikaci do tří oddělených částí. Model, view a controller.

Částí, kterou vidí uživatel je View část. Tato reprezentace může spočívat například ve výpisu určitých seznamů, grafů nebo tabulek.

Controller se naopak stará o zpracování požadavků uživatele. Z těch poté vytvoří požadavky pro Model část programu. Zároveň by také měla validovat data, která od uživatele dostane.

Model část programu je jeho hlavní architekturou. Stará se o správu přijatých dat, logiku celého programu a zajišťuje dodržování určitých pravidel. Pokud je například s daty před uložením tř eba provést nějaké operace, stará se o to právě tato část programu.

U Kanboardu se toto projevuje například tak, že v šablonách jednotlivých stránek není řešená žádná logika a pouze se zde vypisují data, která byla šabloně předána. Naopak v controlleru se neprovádí žádný výpis dat, ale tato data se pouze připravují a předávají šabloně.

4.2 HTML

HTML (Hypertext Markup Language) je značkovací jazyk, který je využívaný především pro tvorbu webových stránek. Tyto stránky jsou propojené hypertextovými odkazy. Vykreslování obsahu poté probíhá na straně klientova prohlížeče. Díky tomu není třeba k jeho zobrazení žádný server. Uživatel si může stránku zobrazit nezávisle sám ve svém prohlížeči.

Jazyk HTML je tvořen množinou značek. K těmto značkám je také možné přiřazovat jejich atributy. Mezi tyto značky se poté vkládá obsah webu, který se má vykreslit.

Značky mohou být párové a nepárové. Název a attributy značky jsou uzavřené v úhlových závorkách $<$ a $>$. Na začátku značky je vždy její název a poté následují atributy[1]. Značky spolu s s obsahem, který je mezi nimi tvoří prvek webu. První značkou webové stránky by měla být značka *<html>*. Ta je pak rozdělená do tří hlavních prvků, kterými jsou *<head>*, *<body>* a *<footer>*. Tyto značky označují hlavičku, hlavní tělo a zápatí celé stránky.

Značky se dají rozdělit na tři základní skupiny[1]:

- Strukturální
- Popisné
- Stylistické

Strukturální značky určují vzhled dokumentu. Můžou označovat například nadpisy, odstavce nebo tabulky.

Popisné značky se neprojeví na vzhledu dokumentu, ale mají za úkol popsat typ obsahu dokumentu. Tyto značky jsou často užívané například roboty různých vyhledávačů. Díky nim mohou poté rozhodnout, na jaké pozici se určitá stránka zobrazí při vyhledávání určitých slov.

Stylistické značky slouží pouze pro změnu vzhledu vykreslovaného obsahu. V poslední době se od nich posupně upouští, neboť mohou být snadno nahrazené kaskádovými styly.

4.3 CSS

Kaskádové styly, neboli CSS je programovací jazyk vytvořený pro úpravu vzhledu HTML nebo XML dokumentů. Umožňuje zpřehlednit HTML kód tím, že styly jednotlivých značek se nemusí psát jako jejich atributy. Chceme li přiřadit určitý styl nějaké značce, máme více možností jak ji v css souboru identifikovat. Můžeme k tomu využít buďto jméno značky, její třídu nebo id. Pokud nemá některý element definované styly, dědí je většinou od svých rodičů. Chceme-li tedy, aby celá stránka využívala určitý font, stačí jej nadefinovat některému z hlavních elementů, a všechny elementy, které obsahuje, jej podědí.

4.4 JavaScript

Spolu s HTML a CSS tvoří hlavní technologie pro tvorbu front-end části webových aplikací. Je to objektově orientovaný, netypovaný, událostmi řízený jazyk. Jeho zpracování probíhá na straně prohlížeče, tudíž nezatěžuje server. Díky tomu může reagovat na akce uživatele, aniž by musel posílat požadavky na server.

Velice užitečná je možnost detekovat určité události v prohlížeči a na jejich základě provádět různé akce. Například při změně hodnoty v selectboxu je možné znovu načíst změněný obsah určitých elementů bez nutnosti nového načtení celé stránky. K získání tohoto obsahu se často využívá Ajaxové volání určité url.

4.4.1 Interact.js

Na internetu je možnost stáhnout spoustu knihoven pro JavaScript, které nám usnadní práci s webovým obsahem. Jednou z těchto knihoven je i Interact.js. Tato knihovna umožňuje vytvořit ze statických elementů pohyblivé. U těchto elementů je pak možné je otáčet, přetahovat, měnit jejich velikost nebo jejich tvar podle toho, které události k nim přiřadíme[2].

Instalace této knihovny do projektu je velice jednoduchá. Můžeme ji nainstalovat pomocí NPM, YARN nebo ji vložit do stránek pomocí HTML značky *<script>*. Poté již můžeme plně využívat všechny funkce této knihovny.

Elementům pak můžeme přiřadit tyto tři základní vlastnosti[2]:

- Draggable
- Dropzone
- Resizable
- Gesturable

Draggable umožní měnit pomocí kurzoru myši pozici elementu. Pro dosažení tohoto stavu stačí na element zavolat metodu *draggable*. Máme také možnost omezit pohyb elementu, například pouze na vertikální nebo horizontální pohyb. Je zde také možnost určit, co se má stát po uvolnění elementu.

Někdy chceme, aby měl uživatel možnost přesunout element pouze do určitých oblastí. Toho lze dosáhnout tak, že na cílový element použijeme metodu *dropzone*. Zde máme také možnost nastavit různá omezení. Například můžeme určit, jaké elementy mohou být do dropzone umístěny a jaké ne.

Dalším užitečným nástrojem, který tato knihovna poskytuje, je možnost měnit velikost elementu. Toho docílíme tak, že na element zavoláme metodu *resizable*. Tím docílíme toho, že element půjde roztahovat nebo smršťovat do stran. Máme také možnost omezit, kterým směrem bude možné element smršťovat, nebo limity, jak moc je možné velikost měnit. Knihovna u této akce umožňuje odchytávat tři události. Začátek změny velikosti elementu, poté samotné měnění velikosti a konec měnění velikosti.

Poslední možnost je dělaná spíše pro mobilní zařízení. Uživatel má možnost měnit velikost elementu pomocí pohybu dvou prstů.

4.5 Databáze

Kanboard umožňuje využívat tři druhy databáze.

- SQLite
- MySQL
- Postgres

Pro práci s menšími projekty postačí SQLite, ovšem při práci na větších projektech se doporučuje využít spíše MySQL nebo Postgres[5].

4.5.1 SQLite

Na rozdíl od ostatních SQL databází, nepotřebuje SQLite vlastní proces na straně serveru. Data jsou ukládána přímo do souboru na disku. To velice usnadňuje například migraci databáze. Díky tomu je dobrá volba pro menší aplikace s menším objemem dat. Pokud chce uživatel využívat SQLite na větších projektech, je dobré mít disk s rychlým čtením a zápisem dat. Dalším problémem může být také fragmentace dat, díky které se může zpomalit doba potřebná k vykonávání příkazů.

Tato databáze je jedna z nejvíce využívaných databází na světě. Je to především proto, že je zabudovaná v téměř všech smartphonech, nehledě na operační systém[13]. To je umožněno také díky tomu, že formát dat není vůbec závislý na operačním systému, který databázi používá.

4.5.2 MySQL

MySQL je relační databáze běžící na straně serveru. Je navržená tak, aby mohla využívat více vláken současně, což umožňuje vyšší rychlost databáze[9]. Každé připojení pak dostane právě jedno vlákno pro vykonání potřebných dotazů. Umožňuje také vytvářet uživatele, kteří se k ní mohou připojit. Jejich autentizace je potom prováděna na základě uživatelského

jména, hesla a hostiteli. Zároveň je také možnost využít pro autentizaci certifikáty přes připojení SSL. Po připojení uživatele server ověřuje, jestli má patřičná práva pro operaci, kterou chce vykonat.

Tato databáze je navržena tak, aby zvládala i větší databáze. Díky tomu se často využívá například pro tvorbu webových stránek a e-shopů. Také Kanboard doporučuje upřednostnit u větších projektů tento typ databáze nebo Postgres před SQLite, která je uložena pouze v jednom souboru.

Data se ukládají na disk na serveru. Zde má každá databáze svoji složku a každá tabulka svůj soubor. Proto je třeba dávat pozor při pojmenovávání tabulek a databází. Na serverech s operačním systémem Windows nezáleží na velkých a malých písmenech, zatímco na unixových ano. Metadata k tabulkám byly dříve uloženy v metadata souborech. Nyní jsou ovšem uloženy v slovníkových tabulkách[8].

Pro správu databáze a jejich dat je možné využít různé nástroje. Takovými nástroji, které běží na serveru jsou například phpMyAdmin nebo Adminer. V případě, že se chce uživatel editovat databázi a na serveru se tyto nástroje nenachází, má možnost si nainstalovat programy pro správu databáze přímo do svého počítače. Příkladem takového programu je například Workbench.

4.5.3 Postgres

Poslední databází, kterou může Kanboard využívat je Postgres. Tento objektově-relační databázový systém umožňuje jako většina jiných databázových systémů vytváření funkcí, indexů, triggerů atd. Umožňuje také tabulkám dědit vlastnosti jiných tabulek. Data se poté tváří, že jsou uložena v rodičovské tabulce.

Pro správu databáze a dat lze opět využít více nástrojů. Jedním z nich je například phpPgAdmin, který vychází z phpMyAdmin a je také určen pro použití na server, tudíž se ovládá pomocí webového rozhraní. V případě, že by chtěl uživatel pro editaci databáze využít program přímo ve svém počítači, může využít například pgAdmin[10].

Kapitola 5

Plánovací modul

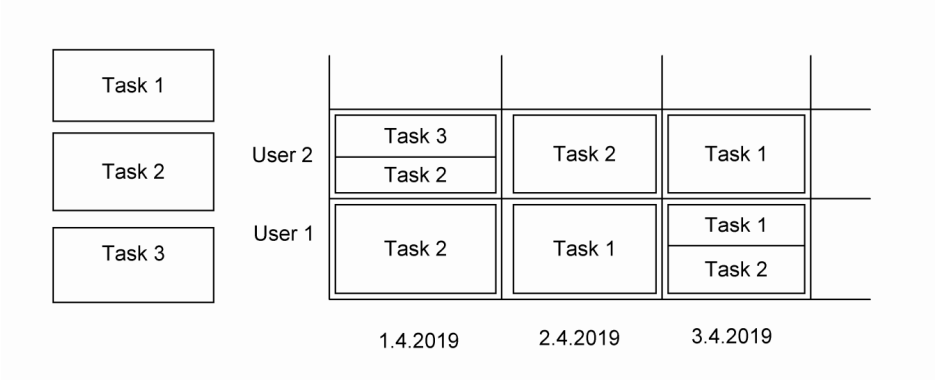
Lidé v týmech často nepracují pouze na jednom úkolu. Zároveň některé úkoly zaberou více času a není možné, aby jeden člověk věnoval po delší dobu pouze takovému úkolu. Proto vedoucí týmů, kteří mají na starost rozdělování práce mezi členy týmu, musí určit, jakou dobu má kdo strávit na určitém úkolu v určitý den. Ne všechny úkoly mají totiž stejnou prioritu.

Právě v tomto by jim měl pomoci modul, kterým se zabývám v této práci. Díky němu může vedoucí týmu pohodlně na jednom místě rozdělit práci všem zaměstnancům.

5.1 Návrh

Hlavním účelem je, aby byl co nejjednodušší na ovládání a co možná nejpřehlednější. To je vidět i návrhu 5.1, že se modul bude snažit nezahlcovat uživatele přílišným množstvím informací. Uživatel uvidí všechny vytvořené úkoly, které ještě nemají přiřazený nějaký čas ke konkrétnímu uživateli a pouhým přetažením do určitého dne uživatele mu tento úkol bude moci zadat na ten konkrétní den. U každého úkolu bude mít poté možnost jeho roztažením či zmenšením změnit počet hodin, po které se má daný uživatel tomuto úkolu věnovat během dne.

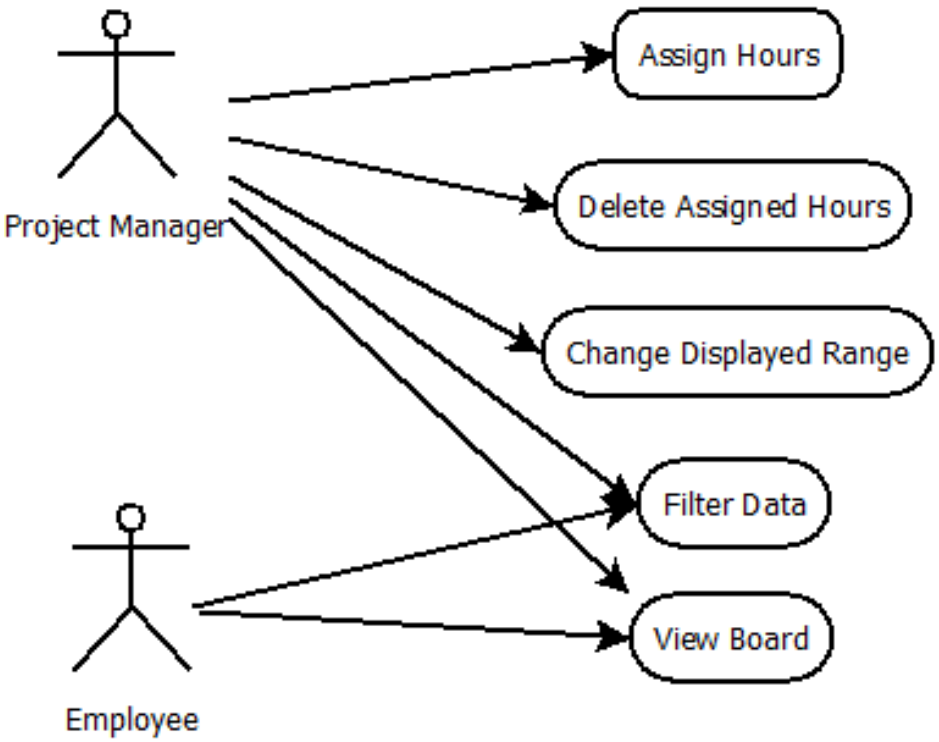
Pro zjednodušení práce si může uživatel zvolit, zda chce na časové ose zobrazit dny, týdny nebo měsíce. Po přetažení úkolu do určitého týdne se nastaví hodiny rovnoměrně do všech pracovních dní tohoto týdne. Zároveň jde také odebrat určitému uživateli přiřazené hodiny u úkolu z určitého dne, nebo hromadně, stejně jako je tomu u přiřazování.



Obrázek 5.1: Návrh základního vzhledu modulu

Všechna potřebná data by pak měla být uložena v jedné tabulce. Ta bude obsahovat návaznost na tabulku uživatelů, úkolů a projektů. Dále pak informaci o tom, kolik bylo uživateli přiřazeno hodin na určitém úkolu a také v který den.

Use case diagram pro modul je na obrázku 5.2.



Obrázek 5.2: Use case diagram modulu

Pro implementaci by se mělo využít co nejvíce tříd a metod, které nabízí Kanboard. Tím by měla být zajištěna správná práce s daty Kanboardu. Celý modul tedy bude objektově orientovaný, se zachováním adresářové struktury Kanboardu.

Ve front-end části se poté počítá s využitím HTML a CSS. Důležitou roli zde bude nejspíš hrát i JavaScript, neboť s jeho pomocí se budou dát přesouvat jednotlivé úkoly a nastavovat jim hodiny.

Kapitola 6

Implementace modulu

V této kapitole se budu zabývat samotnou implementací modulu. Pro jeho zhotovení byly využity jazyky PHP, HTML, CSS a JavaScript. Zároveň byly dodrženy pravidla pro psaní kódu, které platí obecně pro celý Kanboard, aby byla zajištěna plná kompatibilita s ním.

6.0.1 Registrace modulu

O registraci modulu se stará třída *Plugin*, která je umístěná v hlavním adresáři modulu. Zde je hlavní funkcí *Initialize()*, která vloží do určitých sekcí Kanboardu obsah šablon, definovaný v našem modulu. Konkrétně vkládá odkaz na celý modul, který je umístěn v levém sloupci celého Kanboardu. Uživatel tak má možnost přejít do modulu z téměř všech hlavních stránek Kanboardu.

Dále pak vkládá potřebné javascript a css soubory do hlavičky Kanboardu. Z bezpečnostních důvodů Kanboard neumožňuje vložit JavaScript jinak, než touto cestou. Kdyby chtěl tedy uživatel vložit js kód například uprostřed šablony, nebude mu fungovat.

Třída *Plugin* dále obsahuje metody na získání jména modulu, jména autora, verze modulu a také požadované verze Kanboardu pro správné fungování modulu. Jazykové mutace jsou pak načteny v metodě *onStartup()*. Ty jsou uloženy ve složce Locale v hlavním adresáři modulu.

Další důležitou metodou této třídy je pak *getClasses()*, která zaregistruje určené třídy do DI kontejneru. Díky tomu je pak můžeme využívat napříč celým programem bez nutnosti jejich inicializace. Tato metoda pouze vrací pole, ve kterém jsou všechny třídy, které chceme mít v DI kontejneru. Jako index se zde využívá název třídy a hodnota je název, pod kterým bude třída v dostupná v kontejneru.

6.0.2 Definice tabulek

Pro fungování modulu je třeba do databáze přidat jednu tabulku, ve které je uvedeno, kolik hodin denně má uživatel strávit na jednom konkrétním úkolu. Tato tabulka má návaznost na tabulku úkolů, uživatelů a projektů. Přidání takové tabulky je v Kanboardu velice jednoduché. Stačí do složky *Schema* umístit tři soubory, přičemž v každém je definice pro jednu z databází, kterou může Kanboard využívat. Ten pak prochází všechny složky s tímto názvem a v případě, že najde definici tabulky, která ještě není v databázi, ji vytvoří.

Tato tabulka pak obsahuje id uživatele, id úkolu, id projektu, den a hodiny, které má uživatel v ten daný den strávit prací na tomto úkolu. Časové údaje jako hodiny a den jsou

ukládány jako datový typ *int*. V případě hodin jde o celé číslo reprezentující právě počet hodin. U dnů je to pak unixová časová stopa začátku dne, pro který je záznam vytvořen.

Jako primární klíč zde slouží id záznamu. Id uživatele, id úkolu a id projektu jsou provázány pomocí cizích klíčů. Pokud tedy bude například smazán celý projekt, smažou se i záznamy v této tabulce, které jsou na něj navázané.

6.0.3 Výpis obsahu

O výpis veškerého obsahu se v Kanboardu starají controllery. Ty mají za úkol především získat všechna data, která se mají vypsat a předat je šabloně, která se již stará o celkový vzhled stránky. Každá třída, která má sloužit jako controllet by měla rozšiřovat třídu *Base-Controller*, což zajistí její použitelnost v aplikaci. Poskytne tak zároveň controlleru spoustu užitečných funkcí pro získání dat. Díky tomu stačí například pro získání momentálně přihlášeného uživatele zavolat nad třídou controlleru funkci *getUser()*, která se již postará o vše potřebné a vrátí pole dat, reprezentující tohoto uživatele.

Pro vypsání obsahu, který vrací určitá metoda controlleru, je třeba do url zadat tři parametry. Těmito parametry jsou:

- plugin
- controller
- action

Parametry plugin a controller jsou názvy modulu a controlleru, který chceme pro výpis obsahu použít. Kanboard pak sám zjistí, zda je takový modul registrovaný a zda obsahuje právě takový controller. V případě že ano, přichází na řadu parametr action. Aplikace zkontroluje, zda daný controller obsahuje metodu se stejným názvem, jako je uvedený v tomto parametru. Pokud ano, vypíše obsah, který vrací tato metoda.

Pro snadné generování těchto url je v základu Kanboardu připravená třída *url*, která obsahuje spoustu metod pro práci s odkazy. Chceme-li vygenerovat odkaz, který se přidá k obsahu levého menu, stačí zavolat metodu *link()* této třídy. Jejími parametry jsou text odkazu, který se zobrazí, název controlleru, na který bude odkaz směřovat, název metody, který bude v parametru action a nakonec pole dalších volitelných parametrů. V našem případě, kdy chceme využít controller, který se nachází v modulu, je třeba mezi tyto parametry vložit název modulu do parametru plugin. Pro vygenerování textu odkazu je dobré využít překladové funkce *t()*, která zkontroluje, zda pro zadaný výraz existuje překlad, a pokud ano, tak místo původního výrazu vypíše jeho překlad. Tato funkce vypisuje překlad slova pro jazyk, který je nastaven pro celý Kanboard.

Seznam projektů

Jako první se v modulu uživateli vypíše seznam projektů, na kterých se podílí. O výpis tohoto obsahu se stará metoda *index()*, kterou obsahuje controller našeho modulu. Pro získání seznamu projektů je využita třída *projectPagination*, jejíž metoda*getDashboardPaginator()* vrací všechny projekty uživatele, rozdělené do jednotlivých stránek. Limit projektů na jednu stranu se dá nastavit parametrem této metody.

Metoda *index()* přímo HTML, které se má vypsat na výstup. K získání obsahu pak slouží metoda *dashboard()* třídy *layout*, která je též v základu Kanboardu. Ta se volá se dvěma parametry. Prvním je název šablony i s její cestou. Zde je třeba si dávat pozor na velká a

malá písmena. Servery, které využívají operační systém Windows nerozlišují velká a malá písmena, tudíž by šablonu našli nehledě na ně. Ovšem například u unix operačních systémů by nastal problém, že by šablonu nenašel, pokud by se název souboru a složek neshodoval i ve velkých a malých písmenech. Druhým parametrem této metody je pak pole, obsahující data, která se využívají v šabloně. Klíčem je zde název proměnné, pod kterou budou data dostupná v šabloně a hodnotou jsou samotná data.

Hlavní obsah

O výpis hlavního obsahu se stará metoda *detail()*. Ta jako první kontroluje, zda nebyl odeslaný některý z formulářů, které jsou na stránce. Konkrétně se jedná o změnu časového rozsahu sloupců tabulky a filtrování úkolů nebo uživatelů. Pokud byl odeslán některý z těchto parametrů, znovu načte stránku a do url přidá hodnotu z formuláře. Toto zajišťuje metoda *redirect()* třídy *response*. Děje se to hlavně proto, že uživatel po odeslání některého z formulářů muže obnovit stránku a bez přesměrování by mu prohlížeč nabídl, zda chce znovu odeslat formulář.

Poté se vygenerují odkazy pro navigační tlačítka. Ty berou v potaz momentálně zobrazovaný časový rozsah tabulky. Pro generování těchto odkazů je využita metoda *href()* třídy *url*. Ta má tři parametry. Prvním je jméno controlleru, na který ma odkaz směšovat, druhým je pak název metody pro vykreslení odkazu a třetím je pole parametrů, které mají být přidány do url.

6.0.4 Šablony

Vzhled modulu je definován v šablonách. Ty se nachází v adresáři Template. Kvůli přehlednosti jsou pak dále rozděleny do podadresářů podle toho, jaký obsah vypisují. Jako cestu k šabloně pak stačí zadat název modulu a podadresář s názvem souboru šablony. Kvůli přehlednosti je dobré rozdělit šablony na menší části a ty poté rozdělit do jednotlivých souborů. Vyhneme se tak opakovanému psaní stejného kódu.

Pro výpis projektů slouží šablony uložené ve složce ProjectsList. Hlavní šablonou je zde *projects_list.php*, která vkládá hlavní kontejner a poté pomocí cyklu vypisuje všechny projekty uživatele. Samotný projekt je pak po vzoru Kanboardu rozdělen na další tři šablony. Pro výpis názvu projektu s odkazem na jeho detail slouží šablona *project_title.php*. Pak následují informace o projektu jako je jeho vlastní a data začátku a konce projektu. O tento výpis se stará šablona *project_detail.php*. A na konec se vypíší ikony projektu, které poskytují doplňující informace. Ty vypisuje šablona *project_icons.php*.

O vzhled hlavní obrazovky modulu se pak starají šablony uložené ve složce ProjectDetail. Hlavní šablonou je zde *table_container.php*, která vypisuje horní ovládací lištu, seznam nepřiřazených úkolů a tabulku uživatelů. Výpis nepřiřazených úkolů pak probíhá v cyklu, kde pro výpis jedné karty úkolu je využita šablona *table_task.php*.

Výpis samotné tabulky uživatelů probíhá v cyklu, kde se na řádek vypisují uživatelé. Do jednotlivých sloupců k uživateli je pak třeba vypsat již přidělené úkoly a jejich hodiny. O to se stará šablona *table_day_plan.php*, která získá data pro konkrétní časový úsek, který reprezentuje sloupec, a ty pak vypíše po jednotlivých úkolech pomocí šablony *table_task.php*.

6.0.5 Správa úkolů

K tomu, aby šlo jednotlivé karty s úkoly uchopit myší a poté přesunout do příslušného pole tabulky, slouží knihovna Interact.js. Ta ze všech karet, které mají třídu *drag-drop*,

vytvoří elementy, se kterými lze takto manipulovat. Konkrétně se jedná o úkoly, které mají zbývající nepřiřazený čas a jsou umístěné v levém sloupci vedle tabulky uživatelů. Těmto objektům je pak možné nastavit různá omezení a akce při určitých událostech. Nejdůležitější z nich je funkce *dragMoveListener()*, která se vykonává vždy při pohybu nějaké karty. Ta při pohybu vkládá do stylového atributu elementu parametr translate, který zajišťuje právě efekt pohybu.

Dalším důležitým prvkem jsou pak jednotlivá pole tabulky. Ty jsou nadefinovány díky třídě *dropzone* jako místa, kam lze umisťovat jednotlivé úkoly. Každé pole má v datových atributech uložené informace potřebné ke správnému uložení přiřazených hodin. Po vložení úkolu do pole je zachycena událost *ondrop* a následně se provede anonymní funkce, která pomocí ajaxového volání uloží daný úkol k uživateli. Url, kterou tato funkce volá směřuje na metodu *save()* controlleru modulu. Ta poté vrátí obsah stránky, která se má vykreslit. O to se poté postará funkce *html*, která vloží do hlavního elementu nový obsah a starý smaže. Díky tomu není nutné po každém uložení úkolu znovu načítat stránku.

Po vložení úkolu a načtení obsahu tabulky již karta nemá třídu *drag-drop*, tudíž ho už nejde přesouvat. Má však třídu *resize-drag*, která umožní měnit velikost elementu a tím i počet přiřazených hodin. Kvůli přehlednosti odpovídá výška štítku počtu přiřazených hodin. Zde každých 20 pixelů na výšku odpovídá jedné přiřazené hodině. Při měnění výšky je za pomoci odchytávání akce *resizemove* opět měněn vzhled elementu s využitím stylového parametru translate. Po skončení akce je díky akci *resizeend* opět ajaxově volána metoda *save()*, stejně jako při vkládání úkolu do tabulky. Opět je poté překreslen obsah hlavní tabulky.

Odstranění přiřazených hodin funguje na podobném principu jako jejich ukládání. Při kliknutí na křížek v pravém horním rohu karty je ovšem ajaxově volána url, která využívá metodu *delete()*. Zde byl ovšem problém s navázáním akce po kliknutí na elementy, které byly přidány až po načtení stránky. To jsou elementy, které jsou například vypsané při změně přiřazených hodin u některého z úkolů. U takto přidaných karet úkolů nelze akci navázat přímo na událost kliknutí na ně. To je vyřešené tak, že akce není navázána na událost kliknutí na element třídy *remove-task*, ale na jejího rodiče, který po načtení stránky již zůstává stejný. Zde se pak určí třída identifikátor podřazených elementů, na jejichž událost kliknutí bude akce navázána. V našem případě je tímto identifikátorem právě třída *remove-task*.

6.0.6 Filtr dat

Pro získání dat, jako je například seznam úkolů přiřazených k projektu, nabízí Kanboard ve svém základu spoustu metod. Ne vždy však jejich výstup odpovídá naším požadavků. Pro takové případy je zde třída *UserTaskFilter*, která má za úkol získávat data z databáze a upravovat je podle našich potřeb. Nejdůležitější metodou je zde *getTasksForProject()*, která poskytuje seznam úkolů, které ještě nemají přiřazený nějaký čas. Tato metoda má jeden parametr, což je identifikační číslo projektu, pro který je vypisována tabulka uživatelů. Metoda si pomocí třídy *db* vytáhne z tabulky úkolů, jejíž model je definován v třídě *TaskModel*, data všech úkolů, které jsou vytvořené pro daný projekt. Následně je všechny projde a jako první odstraní ty, které nejsou aktivní. Poté si pomocí metody *getHoursForTask()* třídy *userTaskModel* zjistí, kolik je pro daný úkol přiřazeno celkově hodin a porovná jej s přiděleným časem. V případě, že již nezbývá žádný čas k přiřazení k určitému uživateli, odstraní tento úkol ze seznamu. Pokud zbývá, uloží jej do pole s daty, aby se poté

mohl vypsat v seznamu nepřiřazených úkolů. Po projití všech úkolů vrací metoda zbývající metoda pole s těmi, které jsou aktivní a mají dostupný nějaký čas k přiřazení.

6.0.7 Model

Modul ukládá všechna svá data do jedné tabulky s názvem *user_task*. Třídou, která je určená pro práci právě s touto tabulkou je *UserTaskModel*. Ta obsahuje metody, které jsou třeba pro práci s daty v této tabulce. Patří mezi ně například ukládání záznamů, jejich úprava a mazání. Dále pak vyhledávání záznamů v tabulce podle určitých kritérií. Tato třída je po vzoru Kanboardu umístěná v adresáři Model a je také přidána do DI kontejneru při inicializaci modulu.

Ukládání dat

V případě, že chce uživatel někomu přiřadit úkol, je třeba tento záznam uložit do tabulky. To ovšem může být více problematické, než se na první pohled zdá. Úkol se totiž nemusí ukládat pouze do jednoho dne, ale třeba do celého týdne. V takovém případě je třeba pro každý den vytvořit jeden záznam v tabulce. Proto se pro tuto operaci využívá metoda *saveTasksToDays()*, které se kromě id uživatele, úkolu a hodin předá také časové stopy krajních dnů časového úseku, do kterého mají být data zapsána. Ta poté prochází jednotlivé pracovní dny a zapisuje k nim údaj o tom, kolik hodin má na daném úkolu uživatel strávit. Zároveň také kontroluje, zda ještě na úkolu nebyl vyčerpán přidělený čas. Pokud z toho času zbývá méně, než se snaží uživatel zapsat, zapíše se maximální možný počet hodin. Nakonec se zavolá metoda *save()* stejné třídy, která zjistí, zda stejný záznam již neexistuje jen s například rozdílným počtem hodin. Pokud ano, tak tento záznam pouze upraví a pokud ne, vytvoří nový

Mazání dat

Podobný problém, jako byl s ukládáním dat, je i s mazáním. Uživatel má totiž možnost jedním kliknutím smazat záznam z jednoho dne, ale i třeba ze všech dní jednoho týdne. Z toho důvodu se pro mazání dat využívá metoda *removeTasksFromDays()*, které se opět určí časový úsek, ze kterého je třeba data smazat. Ta pak prochází tento časový úsek den po dni a v případě, že tabulka nalezne záznam vyhovující všem podmínkám, smaže ho. Samotné mazání jednoho záznamu pak provádí metoda *removeTask*.

Práce s časem

Díky možnosti změnit časové rozmezí, pro které se zobrazuje tabulka uživatelů, je složitější práce s dny. Právě proto bylo třeba vytvořit více metod, které tuto práci usnadňují. Jednou z těchto metod je *getDateLine()*, která vrací jednotlivé časové rozsahy reprezentující jeden sloupec tabulky. Ta jako první zjistí, od kdy do kdy se má tabulka vykreslit. K tomu slouží dvě základní metody, kterými jsou *getDateFrom()* a *getDateTo()*. Ty se nejdříve podívají, zda nejsou tato data definovaná v parametrech url. Pokud ne, vezme se jako začátek časové osy první den momentálního týdne nebo měsíce. To záleží na momentálně zobrazovaném časovém rozsahu jednotlivých sloupců.

Poté se pomocí třídy *DatePeriod* prochází všechna data v daném rozsahu a ukládají se do pole jako počátek rozsahu sloupce. Jako konec tohoto rozsahu se poté bere začátek

dalšího sloupce. Ten se získá díky metodě *modify()* třídy *DateTime*, která je zabudovaná v základu PHP.

Důležité jsou pak také metody *getNextDate()* a *getPrevDate()*, které vrací počátek a konec dalšího a předchozího časového rozsahu celé tabulky. To se poté využívá pro generovaní odkazů, které slouží pro navigační šipky nad tabulkou uživatelů. Lze pak díky nim pohodlně přepínat mezi zobrazovaným časovým rozsahem dat..

Kapitola 7

Ovládání modulu

Modul byl navržen tak, aby jeho ovládání bylo co nejpřehlednější a co možná nejvíce intuitivní. Stejně jako celý Kanboard se snaží uživatele nezahlcovat přílišným množstvím informací.

7.0.1 Seznam projektů

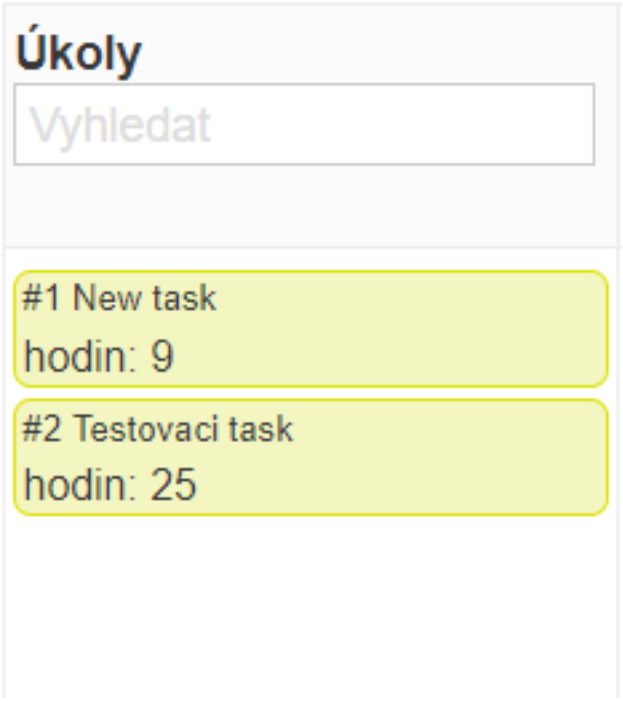
Na úvodní obrazovce modulu má uživatel možnost vidět seznam všech projektů, na kterých se podílí. Pokud nejsou nalezeny žádné takové projekty, je o tom uživatel informován vypsáním této informace na obrazovku.

Název každého projektu je zároveň odkazem na hlavní desku projektu, kde je možné přiřazovat úkoly z tohoto projektu jednotlivým uživatelům. Výpis projektů je také omezen limitem počtu projektů na stránku. To zamezí příliš dlouhému seznamu projektů na jedné stránce. V případě, že je tento limit překročen, zobrazí se pod seznamem navigace mezi jednotlivými stránkami.

7.0.2 Hlavní obsah

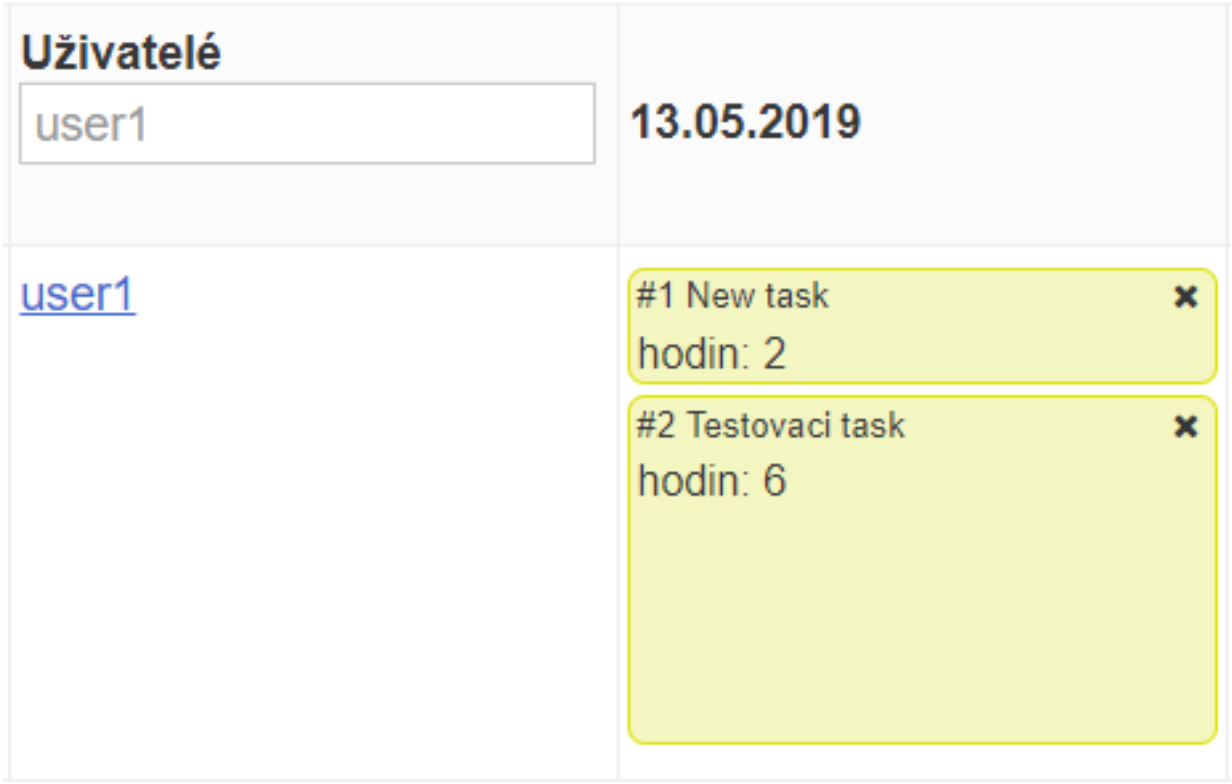
Nejdůležitější částí celého modulu je obrazovka, na které lze uživatelům přiřazovat úkoly a nastavovat počet hodin **??**, které na nich mají strávit. Hlavním prvkem je zde tabulka, kde jeden její řádek reprezentuje jednoho uživatele a jeden její sloupec reprezentuje určitý časový úsek. Uživatele lze filtrovat pomocí textového, které se nachází v horní části sloupce. Výchozí časový úsek je jeden den, ale uživatel jej má možnost změnit pomocí selectboxu nad tabulkou. V případě, že jeden sloupec reprezentuje jeden den, zobrazí se uživateli pracovní dny současného týdne. V případě týdnů a měsíců se zobrazí momentální plus pět následujících. Mezi takto zobrazeným obdobím má uživatel možnost přepínat za pomoci navigačních šipek, umístěných nad tabulkou uživatelů.

Další důležitou částí je sloupec vlevo od tabulky uživatelů. Ten obsahuje všechny aktivní úkoly z projektu, u kterých ještě zbývá přiřadit nějaký čas. Informace o tom, kolik hodin je nepřiřazených, je vypsána pod názvem úkolu. Příklad takového seznamu je na obrázku 7.1.



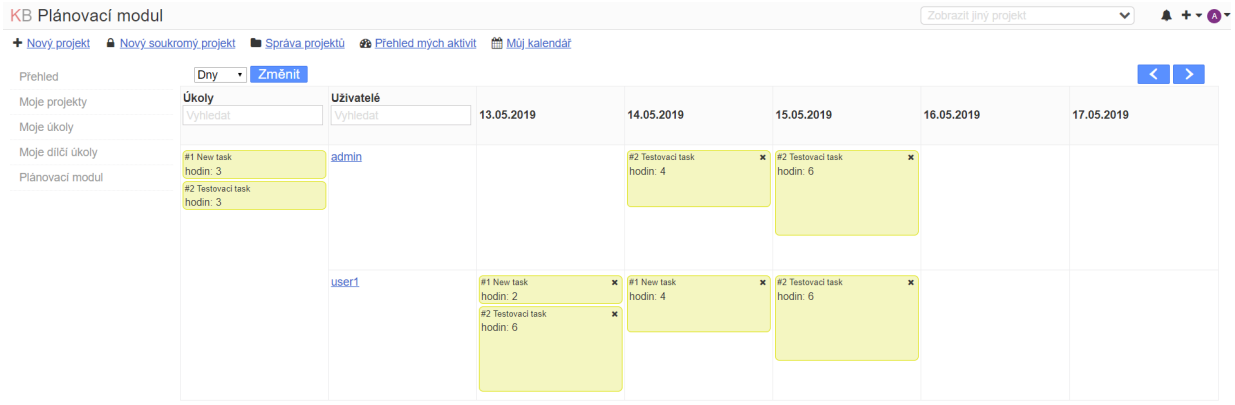
Obrázek 7.1: Příklad seznamu úkolů

Tyto úkoly jsou reprezentované štítky, které se dají filtrovat podle názvu, pomocí textového pole nad nimi. V případě, že chceme přiřadit hodiny práce na úkolu ke konkrétnímu uživateli, stačí štítek prostě přetáhnout do příslušného pole v tabulce. Po přetažení je možné štítek roztahovat pomocí jeho spodního okraje a tím měnit počet hodin, které ma uživatel na tomto úkolu strávit v daný den. Uživatel může mít v jeden den přiřazených maximálně 8 hodin práce na úkolech. Pokud pak chceme zrušit uživateli přiřazené hodiny, stačí pouze kliknout na křížek v pravém horním rohu štítku. Příklad jednoho dne uživatele, ve kterém má přiřazené všechny hodiny na více úkolech, je na obrázku 7.2.



Obrázek 7.2: Přiřazené úkoly k uživateli

Některé úkoly ale mohou mít delší přidělený čas a bylo by nepraktické je přiřazovat ke každému dni zvlášť. K tomu právě slouží možnost přepnutí časového rozsahu jednoho sloupce selectboxem nad tabulkou uživatelů. Pokud je pak úkol přiřazen například do jednoho týdne, přiřadí se do každého pracovního dne čtyři hodiny práce na tomto úkolu. I zde je pak možnost určovat, kolik hodin má uživatel strávit denně na úkolu. Tyto hodiny se pak mění u každého dne stejně, pokud tedy nejsou vyčerpány přidělené hodiny na úkolu, nebo pokud uživatel nemá u některého dne již přidělených osm hodin. U jiného než denního zobrazení se u úkolů zobrazuje průměrný počet přidělených hodin v časovém úseku, který představuje ten konkrétní sloupec tabulky.



Obrázek 7.3: Hlavní obrazovka modulu

Kapitola 8

Testování

Testování funkčnosti modulu probíhalo tak, že jsem se snažil nasimulovat jeho používání v menší firmě. Vytvořil jsem tedy testovací projekty, ke kterým jsem vytvořil testovací úkoly s určitým přiřazeným časem. Dále pak několik testovacích uživatelů, ke kterým jsem pak přiřazoval tyto úkoly. Jednoduchost modulu dělá práci s ním uživatelsky přívětivou. U většího počtu úkolů a uživatelů je velice užitečná možnost filtrovat je pomocí jejich názvů. Také odstranění neaktivních úkolů a těch, kterým již nelze přidělit žádný čas, velice zpřehlednilo jejich seznam.

Po výkonnostní stránce jsem nezaznamenal žádný problém ani při větším počtu projektů a úkolů. Ale vzhledem k tomu, že jsem modul testoval pouze s omezeným množstvím dat, není vyloučeno, že by se tyto problémy nemohli objevit. Zejména pak ve firmách, kde mají desítky projektů a stovky úkolů. To by ovšem odhalilo až testování v ostrém provozu.

Kapitola 9

Rozšíření

Modul by se dal rozšířit o mnoho dalších funkcí. Zde se budu věnovat návrhu některých z nich. Často by se mohlo ovšem jednat o rozšíření, která by byla specifická pro konkrétní požadavky firmy.

9.0.1 Nastavení modulu

Obecné nastavení modulu by mohlo uživatelům umožnit upravit si základní funkce modulu tak, aby vyhovovali jejich požadavkům. Mohla by zde být například možnost, jak velký časový rozsah má tabulka zobrazovat. Nyní se zobrazuje pět časových kroků od počátečního data, aby byla tabulka přehledná. Někomu by ovšem mohlo vyhovovat zobrazit například sva týdny v jedné tabulce.

Dále by pak šlo například nastavovat velikost dostupných časových kroků. Uživatel by pak mohl například zobrazovat v jednom sloupci rozsah třeba tří dnů. To by ovšem záleželo na požadavcích uživatele, proto by bylo dobré, aby si uživatel tyto rozsahy přidával dynamicky.

9.0.2 Přiřazování úkolů

Kanboad umožňuje přiřadit ke konkrétnímu úkolu konkrétního uživatele. Modul na toto ovšem nebere ohled, protože umožňuje přiřadit hodiny práce na jednom úkolu více uživatelům. Dalším rozšířením by pak mohlo být, že by se při přiřazení hodin uživateli zároveň i přiřadil úkol. Zde by byl ovšem problém právě s tím, že by se pak modul snažil přiřadit uživatele k úkolu, který již jednoho má. Musela by se tedy upravit práce s tímto přiřazováním v celém Kanboardu. To by mělo být možné udělat bez zásahu do kódu samotného Kanboardu pomocí přetěžování tříd. Po takové úpravě by pak ale bylo třeba pečlivě projít celý Kanboard a najít, kde všude je potřeba provést změny.

Přetížení základních tříd Kanboardu by také mohlo způsobit nefunkčnost dalších modulů. Uživatel by pak musel každý instalovaný modul projít a zanalyzovat, zda mu nebude možnost přiřadit více uživatelů k jednomu úkolu bránit v jeho činnosti.

Kapitola 10

Závěr

Vytvářením modulu jsem získal spoustu nových zkušeností, především pak s prací v Kanboardu. Ze začátku může být zdlouhavé zjišťování, jak vlastně Kanboard a jeho moduly fungují. K tomu ovšem pomůže samotná dokumentace Kanboardu, ve které jsou popsané základní funkce. Bohužel mnoho věcí zde ale nenajdeme a proto je pak třeba procházet samotný kód Kanboardu a potřebné funkce hledat. Ale poté, co se člověk ve všem zorientuje, je práce velice pohodlná a spousta věcí jde snadno udělat za pomoci základních tříd Kanboardu.

Výsledný modul tedy odpovídá původním požadavkům na jeho funkčnost. To ovšem neznamená, že se při jeho používání nemůžou vyskytnout připomínky a návrhy, jak jej vylepšit. Ne všechny původně navržené věci musí plně vyhovovat uživatelům, kteří jej budou využívat. I tak by ovšem měl velice usnadnit práci projektovým manažerům, vedoucím týmů a lidem, kteří se starají o přidělování práce ve firmách.

Literatura

[1] HTML značky [online]. [cit. 8.5.2019].
URL
https://www.w3.org/TR/html/syntax.html#writing-html-documents-elements

[2] Interact.js [online]. [cit. 8.5.2019].
URL https://interactjs.io/docs/

[3] Kanboard moduly [online]. [cit. 7.5.2019].
URL https://kanboard.org/plugins.html

[4] Kanboard [online]. [cit. 7.5.2019].
URL https://kanboard.org/

[5] Kanboard požadavky [online]. [cit. 7.5.2019].
URL https://docs.kanboard.org/en/latest/admin_guide/requirements.html

[6] Kanboard uživatelé [online]. [cit. 7.5.2019].
URL https://docs.kanboard.org/en/latest/user_guide/users.html

[7] Kanboard úkoly [online]. [cit. 7.5.2019].
URL https://docs.kanboard.org/en/latest/user_guide/tasks.html

[8] MySQL metadata[online]. [cit. 9.5.2019].
URL https://dev.mysql.com/doc/refman/8.0/en/data-dictionary.html

[9] MySQL [online]. [cit. 9.5.2019].
URL https://dev.mysql.com/doc/refman/8.0/en/features.html

[10] PgAdmin[online]. [cit. 9.5.2019].
URL https://www.pgadmin.org/

[11] PHP 7.2 [online]. [cit. 8.5.2019].
URL https://www.php.net/releases/7_2_0.php

[12] PHP [online]. [cit. 7.5.2019].
URL https://www.php.net/manual/en/intro-whatis.php

[13] SQLite [online]. [cit. 8.5.2019].
URL https://www.sqlite.org/index.html