UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

GROUP ASSISTANT – SYSTÉM NA PODPORU PRÁCE V SKUPINE

BAKALÁRSKA PRÁCA

2021
ANNA REBEKA SOJKA

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

GROUP ASSISTANT – SYSTÉM NA PODPORU PRÁCE V SKUPINE

BAKALÁRSKA PRÁCA

Študijný program:	Aplikovaná informatika
Študijný odbor:	Informatika
Školiace pracovisko:	Katedra didaktiky matematiky, fyziky a informatiky
Školiteľka:	Doc. RNDr. Ľudmila Jašková, PhD.

2021

ANNA REBEKA SOJKA

98907330

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Anna Rebeka Sojka
Študijný program: aplikovaná informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Group Assistant - systém na podporu práce v skupine
Group Assistant - a system supporting group work

Anotácia: Nástroj na podporu práce v skupine poskytne rôzne možnosti pre rôzne typy členov skupiny na základe nastavených používateľských práv.
Každý člen bude mať vlastný profil a podľa používateľskej role budú každému oprávnenému dostupné funkcie :
shvaľovanie resp. zamietanie úloh a ich stavu, posielanie pozvánok do skupiny pomocou e-mailu, plánovanie stretnutí, formulár pre zápis zo stretnutia, plánovanie udalostí, zadávanie úloh jednotlivým členom skupiny s termínom dokončenia, odovzdávanie úloh so súkromným komentárom hodnotiaceho/nadriadeného, e-mail notifikácie pre úlohy/stretnutia, udalosti, správy hromadné, súkromné medzi členmi (vrátane príloh), zoznam dôležitých kontaktov a grafický prehľad a štatistika plnenia úloh jednotlivými členmi.
Ako vývojový nástroj zvažujeme použiť Laravel framework.

Cieľ: Autor navrhne a vytvorí webovú aplikáciu Group Assistant - nástroj na zabezpečenie, uľahčenie a zefektívnenie komunikácie, či plánovania úloh v rámci ľubovoľne veľkej skupiny.

Vedúci: doc. RNDr. Ľudmila Jašková, PhD.
Katedra: FMFI.KDMFI - Katedra didaktiky matematiky, fyziky a informatiky
Vedúci katedry: prof. RNDr. Ivan Kalaš, PhD.

Dátum zadania: 30.09.2020

Dátum schválenia: 06.10.2020

doc. RNDr. Damas Gruska, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Čestné prehlásenie

Čestne prehlasujem, že bakalársku prácu „Group Assistant – systém na podporu práce v skupine“

som vypracovala samostatne s použitím uvedenej literatúry a pod vedením mojej školiteľky.

V Bratislave 30. 5. 2021 .....

Anna Rebeka Sojka

Poďakovanie

Ďakujem mojej školiteľke, doc. RNDr. Ľudmile Jaškovej, PhD, za jej odborné vedenie, pomoc, ochotu, čas, všetky konzultácie a usmernenia.

Tiež chcem poďakovať svojej rodine a priateľom za ich neustálu podporu a trpezlivosť.

Abstrakt

Cieľom tejto bakalárskej práce bolo vytvoriť webovú aplikáciu, ktorá bude slúžiť ako systém na podporu práce v skupine („group assistant“). Výsledná aplikácia umožňuje používateľovi vytvárať a spravovať skupiny, úlohy a udalosti. Taktiež poskytuje niekoľko možností na komunikáciu medzi členmi jednotlivo alebo v rámci celej skupiny. Vyznačuje sa najmä možnosťou pre používateľa zjednotiť všetky povinnosti zo všetkých skupín na jednom ľahko dostupnom mieste, pričom zachováva súkromie všetkých zainteresovaných členov.

Implementácia tejto webovej aplikácie bola realizovaná pomocou Laravel frameworku pre backend, Vue.js frameworku pre frontend a MySQL databázy, v spolupráci s menšími prídavnými knižnicami.

V záverečnej fáze vývoja prešla aplikácia testovaním niekoľkými skupinami používateľov. Na základe spätnej väzby z tohto testovania som aplikáciu úspešne doladila.

Kľúčové slová: webová aplikácia, práca v skupine, Laravel framework, Vue.js framework, MySQL databáza

Abstract

The goal of this bachelor thesis is to create a web application, which will serve as a group assistant. The finalized application allows the user to create and manage groups, assignments, and events. Furthermore, it provides several options for communication in-between group members – individually and all together. Its key trait is that the application unites all the user's responsibilities from all groups in one easily accessible place, while it is preserving the privacy of all members concerned.

The implementation of this web application was realized using the Laravel framework for the backend, the Vue.js framework for the frontend and the MySQL database, with help of smaller additional libraries.

In the final stage of development, the application was tested by several groups of users. Based on the feedback from this testing, I have successfully finalized the application.

Key words: web application, group work, Laravel framework, Vue.js framework, MySQL database

Obsah

Zoznam obrázkov 10
Úvod 11
1 Východiská 12
1.1 Vymedzenie základných pojmov 12
Webová stránka 12
Webová aplikácia 12
Databáza 13
Jazyk PHP 13
PHP Frameworky 13
1.2 Použité technológie 13
Framework Laravel 8 13
Framework Vue.js 15
Pusher Channels 15
Framework Tailwindcss 16
1.3 Prehľad podobných aplikácií 16
Aplikácie dostupné na internete 16
2 Návrh 24
2.1 Používateľské rozhranie 24
Navigácia 24
Panel skupiny (ľavý panel) 25
Panel členov skupiny (pravý panel) 25
Hlavný panel a domovská stránka 25
2.2 Databázový model 26
Users 27
Groups 27
Invites 27
Group whiteboard posts 28
Assignments 28
Assignments files 28
Assignments comments 28
Events 28

Event comments 28
Chatrooms 29
Messages 29
2.3 Používateľské role 29
Vytváranie nových skupín 30
Správa skupiny 30
Opustenie skupiny 30
Nástenka skupiny a jej príspevky 30
Úlohy 30
Udalosti 31
Štatistika plnenia úloh 31
2.4 Architektúra aplikácie 31
Backend 31
Frontend 31
2.5 Požiadavky na server 32
3 Implementácia 33
3.1 MVC model 33
Sekvenčný diagram 33
Model 34
View 36
Controller 36
3.2 Architektúra aplikácie 37
Middleware a web.php 37
Vue.js komponenty a app.js 38
Axios 39
Odosielanie e-mailov 40
Eventy a broadcasting 40
Premenné prostredia (.env) 41
4 Testovanie 42
Záver 43
Prílohy 44
Literatúra 45

Zoznam obrázkov

Obrázok 1: Výber účelu pre Teamhood [8] 17
Obrázok 2: Tabuľka úloh Teamhood [8] 18
Obrázok 3: Editovanie úloh Teamhood [8] 19
Obrázok 4: Registrácia MeisterTask [9] 19
Obrázok 5: Nástenka MeisterTask, [9] 20
Obrázok 6: Projekt MeisterTask, [9] 21
Obrázok 7: Časovač Clockify [10] 22
Obrázok 8: Nástenka Clockify [10] 22
Obrázok 9: Tímy Clockify, [10] 23
Obrázok 10: Návrh GUI 24
Obrázok 11: Konečná verzia GUI 25
Obrázok 12: Databázový model 26
Obrázok 13: Use-case diagram 29
Obrázok 14: Sekvenčný diagram 33
Obrázok 15: Model – pole $fillable 34
Obrázok 16: Model – vzťahy 35
Obrázok 17: Komponent ../profile/show.blade.php 36
Obrázok 18: Definícia funkcie newAssignmentNotifyUsers 37
Obrázok 19: Cesty týkajúce sa používateľských profilov 38
Obrázok 20: Vue.js cesty 38
Obrázok 21: Vue.js mená 38
Obrázok 22: Axios GET požiadavka [16] 39
Obrázok 23: Axios POST požiadavka 39
Obrázok 24: Axios PATCH požiadavka 39
Obrázok 25: Axios DELETE požiadavka 40
Obrázok 26: Súbor channels.php 41

Úvod

Informatika a digitálne technológie dnes umožňujú ľuďom prekonávať obrovské vzdialenosti v priebehu pár sekúnd, a tak spájať ľudí po celom svete. Na rebríčkoch najnavštevovanejších internetových stránok a masovo obľúbených aplikácií si držia nespochybniteľné miesta sociálne siete. Používatelia majú široké spektrum výberu aj čo sa týka aplikácií, ktoré napomáhajú organizácií každodenného života – ako práce jednotlivca, ale aj tímov. Toto obdobie poznačené pandémiou ochorenia COVID-19 ukazuje viac ako kedykoľvek predtým dôležitosť dostupnosti softvérov ako je Microsoft Teams, ktorý sme využívali na predmetoch počas dištančnej výučby. Veľakrát som sa stretla aj s aplikáciami, ktoré nepovažujem za dostatočne intuitívne pre menej skúsených používateľov a sama som potrebovala dohľadať, ako robiť aj jednoduché úkony. Čo mne osobne najviac chýbalo bolo zjednotenie všetkých mojich úloh, všetkých udalostí zo všetkých oblastí mojich povinností. Všetky aplikácie, ktoré som našla, spĺňali len čiastočne to, čo som hľadala a vždy chýbala podstatná časť. Facebook dokáže manažovať udalosti, ale stráca sa aspekt súkromia. Microsoft Teams zjednocuje tímy (súkromne), ale neposkytuje rýchly prehľad všetkých nadchádzajúcich povinností pre daný deň. Teamhood je webová aplikácia, ktorá ponúka prehľad povinností, ale chýba komunikácia medzi členmi (čet, skupinový čet, komentáre a podobne). Rozhodla som sa preto v rámci mojej bakalárskej práce vytvoriť webovú aplikáciu, systém na podporu práce v skupine („group assistant"), ktorý som nazvala „Radar". Cieľom bolo, aby táto aplikácia umožnila používateľovi zjednotiť jeho povinnosti na rýchlo dostupnom mieste, komunikovať s členmi skupiny, vytvárať úlohy a udalosti pre jednotlivých členov a pri tom všetkom zachovať súkromie daného používateľa a jeho spolupracovníkov v skupine.

V prvej kapitole sa nachádza popis technológií, ktoré som použila pri tvorbe tejto webovej aplikácie a analýza podobných existujúcich riešení.

Druhá kapitola je venovaná plánovaniu implementácie, teda obsahuje návrh používateľského rozhrania, databázový model a diagram používateľských rolí.

Tretia kapitola popisuje technickú stránku práce – implementáciu. Nachádza sa v nej popis architektúry kódu a vysvetlenia riešení zložitejších problémov.

Posledná, štvrtá kapitola obsahuje výsledky testovania a ich prínos. Zdrojový kód aplikácie, ktorú som vytvorila, sa nachádza v prílohe.

1 Východiská

V tejto kapitole sa nachádza vymedzenie základných pojmov, ako sú napríklad jazyk PHP alebo PHP framework. Ďalej sú vymenované použité technológie, teda aj frameworky, ako Laravel a Vue.js. Posledná časť je venovaná prehľadu podobných existujúcich webových aplikácií.

1.1 Vymedzenie základných pojmov

Táto časť obsahuje definície kľúčových pojmov, ktoré v práci opakovane používam.

Webová stránka

Webová stránka je online dostupné miesto na sieti, najmä na internete, sprístupňované prostredníctvom webového prehliadača a využívajúce hypertextový prenosový protokol alebo jeho zabezpečenú verziu. Webová stránka tvorí jednu vizuálnu obrazovku webového sídla, aj ak je zložená z viacerých rámov [1].

Webová aplikácia

Webová aplikácia je taká aplikácia, ktorú nie je nutné inštalovať na zariadení používateľa (počítač, tablet, smartfón) a je možné ju spustiť z ktoréhokoľvek zariadenia pomocou webového prehliadača, pretože je spustená na strane servera. Vzhľadom k tomu, že je potrebný len prehliadač, sa webová aplikácia niekedy tiež nazýva ako ľahký klient [2].

Výhody webových aplikácií:

- nemusia sa inštalovať,
- používateľ nemusí nič aktualizovať (aktualizácia prebieha na serveri),
- používateľ potrebuje iba webový prehliadač,
- dáta sú uchovávané a zálohované na serveri a sú prístupné odkiaľkoľvek.

Nevýhody webových aplikácií:

- vyžadujú pripojenie na internet,
- niekedy môže byť pomalší tok dát a práce s aplikáciou (rýchlosť je závislá na kvalite pripojenia),
- možné bezpečnostné riziko úniku dát v prípade nedostatočne zabezpečenej služby zo strany poskytovateľa.

Databáza

Databáza je štruktúrovaný súbor dát uložený na pamäťovom médiu. Databázy sú logicky, podľa určitých pravidiel, štruktúrované dáta a to vrátane systému ich vzájomných vzťahov a väzieb. Ide o bázu dát, ktorú tvorí jedna alebo viac tabuliek [3].

Jazyk PHP

PHP je serverový skriptovací jazyk a výkonný nástroj na vytváranie dynamických a interaktívnych webových stránok.

PHP je široko používanou, bezplatnou a účinnou alternatívou voči konkurencii, akou je napríklad Microsoft ASP.

PHP 7 je stabilná verzia, ktorá je zároveň aj medzi požiadavkami na server zabezpečujúcimi spoľahlivý chod mojej aplikácie (viď 1.3) [5].

PHP Frameworky

PHP frameworky (PHP Frameworks) sú frameworky napísané v programovacom jazyku PHP. Frameworky sú akousi nadstavbou jazyka PHP - obsahujú predpripravené celky, z ktorých sa aplikácia skladá rýchlejšie než využitím čistého jazyka PHP [4].

1.2 Použité technológie

Táto časť obsahuje popis technológií, ktoré som si zvolila pre vývoj aplikácie. Zároveň zdôvodňujem tento výber, vzhľadom na rozsiahle možnosti, ktoré sú aktuálne dostupné s prihliadnutím na ich podporu a spoľahlivosť pre potenciálne širšiu verejnosť a s podmienkou, že ide o open-source licenciu.

Pre voľne dostupný Apache web server som používala XAMPP, populárnu voľbu pre Windows, pozostávajúcu z MySQL, PHP a Perl. Textový editor, v ktorom som pracovala, bol Visual Studio Code.

Framework Laravel 8

Laravel je open-source PHP framework, ktorý vytvoril Taylor Otwell v roku 2011 [13]. Aktuálne je zverejnená verzia číslo 8, pomocou ktorej som pracovala na tvorbe aplikácie. Ponúka najmä rozsiahlu podporu pre programovanie backendu, ale aj istú časť frontendu. Avšak pre vývoj dynamickej webovej aplikácie sa v praxi kombinuje väčšinou s JavaScript frameworkom Vue.js alebo prípadne s HTML a frameworkom Angular. Pre frontend som zvolila Vue.js, keďže som sa dočítala z viacerých dokumentácií [6, 14, 15], že je pre tento účel najvhodnejší vďaka svojej kompatibilite s frameworkom Laravel.

Ale tento framework je silným nástrojom pre vývoj aj samostatne. Ponúka užitočné funkcionality, ktoré teraz bližšie popíšeme.

- **MVC model a Eloquent ORM**

  Architektonický vzor MVC model rozdeľuje kód do troch typov tried: Model, View a Controller. Vďaka tomu sa aj pre ďalšieho programátora kód javí ako čitateľnejší a prehľadnejší, keďže je potrebné dodržiavať konvencie.

  - **Model** (Laravelov objektovo-relačný mapovač Eloquent) reprezentuje dáta z databázovej tabuľky.
  - **View** slúži na separovanie kódu pre používateľské rozhranie (vďaka čomu aj celý HTML kód je členený do viacerých komponentov).
  - **Controller** je možné označiť ako handler (či „manažér“) konkrétneho Modelu. Napríklad GroupController (či „manažér skupiny“) bude vytvárať, ukladať, editovať, mazať skupinu alebo skupiny z databázy a taktiež bude vracať triedy View, ktoré si používateľ vypýta ako profil skupiny, editačný formulár pre úpravy skupiny a podobne.

- **Artisan**

  Laravel prichádza aj s vlastným rozhraním v príkazovom riadku. Pre zobrazenie všetkých funkcií, ktoré vie Artisan vykonať, slúži príkaz **php artisan list.** Každý príkaz má aj pomocný popis, ktorý sa zobrazí po pridaní reťazca „help“ pred kľúčové slovo. Teda príkaz pre zobrazenie návodu pre **php artisan migrate** (príkaz pre vytvorenie databázy na základe kódu), by vyzeral nasledovne **php artisan help migrate.** Medzi populárne príkazy patrí aj **php artisan tinker,** ktorý spustí pre Artisan vlastné REPL (Read-Eval-Print-Loop) konzolové programovacie prostredie, ideálne pre izolované testovanie jednoduchších príkazov.

  Pomocou takýchto Artisan príkazov sa väčšinou vytvárajú všetky triedy a nástroje projektu, keďže vďaka prepínačom sa navzájom prepájajú súvisiace triedy a upravujú vstupy pre štandardné funkcie. Taktiež sa vďaka tomuto spôsobu intuitívnejšie dodržujú konvencie, keďže sú triedy a s nimi aj funkcie predpripravené.

- **Query Builder**

  Vytvára databázové dopyty pomocou vlastných metód. Teda programátor nemusí písať príkazy priamo v SQL, pričom je chránený pred SQL injection útokmi (druhá strana využíva nedostatočne alebo úplne neošetrené príkazy a vkladá vlastný SQL kód).

- **Migrations**

  Vďaka migráciám je možné vytvoriť všetky tabuľky databázy (vrátane integritných obmedzení a podobne) a pohodlne ich upravovať, zálohovať, presúvať a znovu využívať v ďalšom projekte.

Existuje aj mnoho ďalších funkcionalít, s čím sa spája hlavná výhoda frameworku Laravel a tou je jeho podrobne vypracovaná mnohostranná dokumentácia a oficiálne video tutoriály, ktoré sú adresované rovnako začiatočníkom, ako aj skúseným programátorom.

Framework Vue.js

Vue je framework, ktorý sa sústreďuje na vytváranie používateľských rozhraní. Na rozdiel od iných frameworkov je Vue od základu navrhnutý tak, aby bol použiteľný samostatne, ale aj v kombinácií s inými knižnicami alebo existujúcimi projektmi. Taktiež je vhodný pre sofistikované jednostránkové aplikácie (Single-Page Applications), ak sa používa v kombinácii s modernými nástrojmi a podpornými knižnicami [6]. Pri tvorbe mojej aplikácie som ho využila v kombinácii s Laravelom, kde poslúžil pri programovaní frontend časti kódu a zabezpečil potrebné dynamické vlastnosti. Vďaka Vue už nebolo potrebné zbytočné opätovné načítavanie stránky, teda zmeny v databáze sa okamžite zobrazili používateľovi. Dokonca v spolupráci s hosting službou Pusher umožnila kombinácia Laravel ako backend a Vue.js ako frontend zobrazovať zmeny na stránke v reálnom čase, ako je to napríklad nevyhnutné pre čet alebo vhodné pre sekciu komentáre.

Podobne ako framework Laravel disponuje Vue detailnou dokumentáciou a rastúcou popularitou.

Pusher Channels

Pusher Channels, teda Pusher kanály, poskytujú komunikáciu v reálnom čase medzi servermi, aplikáciami a zariadeniami. Kanály sa používajú pre tabuľky v reálnom čase, zoznamy používateľov v reálnom čase, mapy v reálnom čase, hry pre viacerých hráčov a pre mnoho ďalších typov komponentov vyžadujúcich neustálu aktualizáciu používateľského rozhrania.

V prípade, že nastane v systéme zmena, Pusher umožní aktualizovať webové stránky, aplikácie a zariadenia [7]. Podobne je to napríklad v prípade udalosti na stránke. Keď používateľ pridá nový komentár, tak sa táto udalosť vysiela zvyšným používateľom v danej skupine a následne sa zobrazí bez opätovného načítavania stránky.

Framework Tailwindcss

CSS framework Tailwindcss [12] som sa rozhodla použiť aj z toho dôvodu, že som s ním už mala dobré skúsenosti, ale najmä pre jeho veľkú mieru prispôsobivosti a jeho podrobne spracovanú dokumentáciu. Obsahuje bohatú množinu tried s intuitívnymi editovateľnými názvami (napríklad pre nastavenie margin-top elementu, sa napíše mt-2) a vyznačuje sa aj veľmi dobrou podporou a kompatibilitou.

1.3 Prehľad podobných aplikácií

Pri hodnotení aplikácií uvedených v tejto časti práce som vychádzala z vlastných skúseností s webovými aplikáciami, ktoré však boli nepochybne ovplyvnené aj štúdiom na FMFI UK. Pozorovala som tiež nových používateľov digitálnych technológií. Čítala som aj niekoľko recenzií a zohľadnila kritériá, ktoré sú popísané v článku „7 najlepších aplikácií pre manažment tímu v roku 2020“ (7 Best Team Management Apps for 2020) [11].

Kritériá, ktoré považujem za smerodajné a chcela som ich implementovať aj do svojej práce, sú nasledovné.

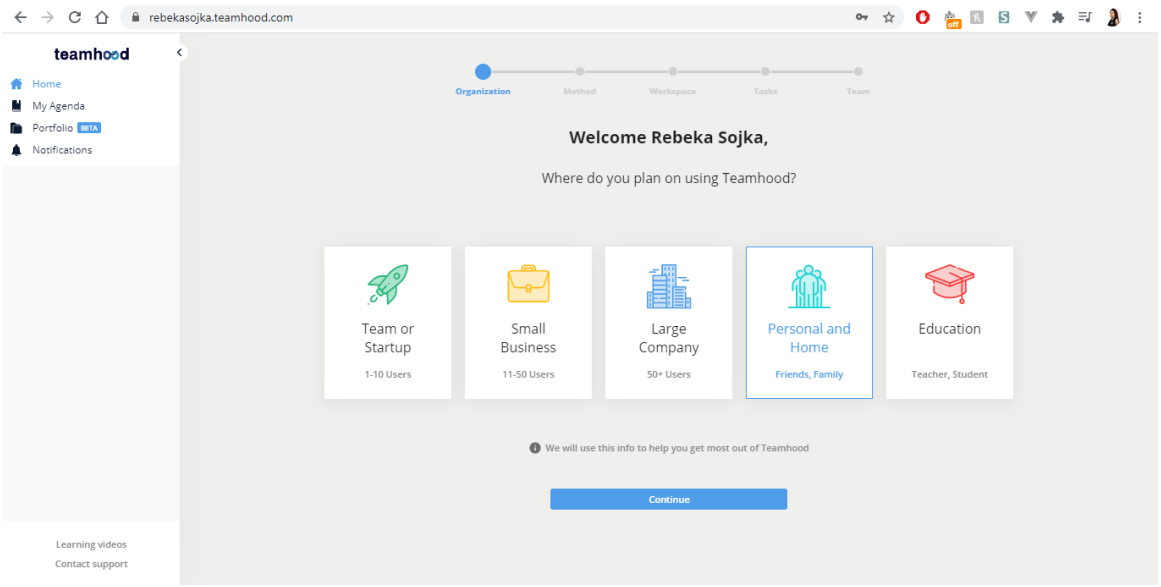
- **Jednoduché používanie:** Ako jednoduché je používanie aplikácie? Môže každý v tíme využiť všetky funkcie, ktoré ponúka?
- **Výnimočné vlastnosti:** Čo dokáže daná aplikácia, čím sa odlišuje od iných aplikácií? Sú tieto ďalšie funkcie užitočné pre členov tímu? Vyhovuje aplikácia špecifickým potrebám tímu?
- **Implementácia a integrácia:** Je aplikácia kompatibilná s programami, ktoré už členovia tímu používajú? Je kompatibilná s počítačmi, smartfónmi a inými zariadeniami všetkých členov tímu?
- **Spektrum využitia:** Čo všetko aplikácia podporuje? Poskytuje napríklad primerané úložisko dokumentov? Môže si každý člen tímu vytvoriť účet a ľahko ho používať?

Aplikácie dostupné na internete

Teamhood

Teamhood [8] je platená webová aplikácia pre rôzne typy skupín. Pomáha tímom prevziať úplnú kontrolu nad svojimi projektmi. Ponúka ľahké zostavovanie a aktualizovanie plánov, sledovanie ich stavu a celkového postupu práce.

Po vytvorení konta si aplikácia pýta od používateľa informáciu, na aké účely ju plánuje používať a aké typy projektov plánuje vytvárať, ako je vidieť na obrázku číslo 1.



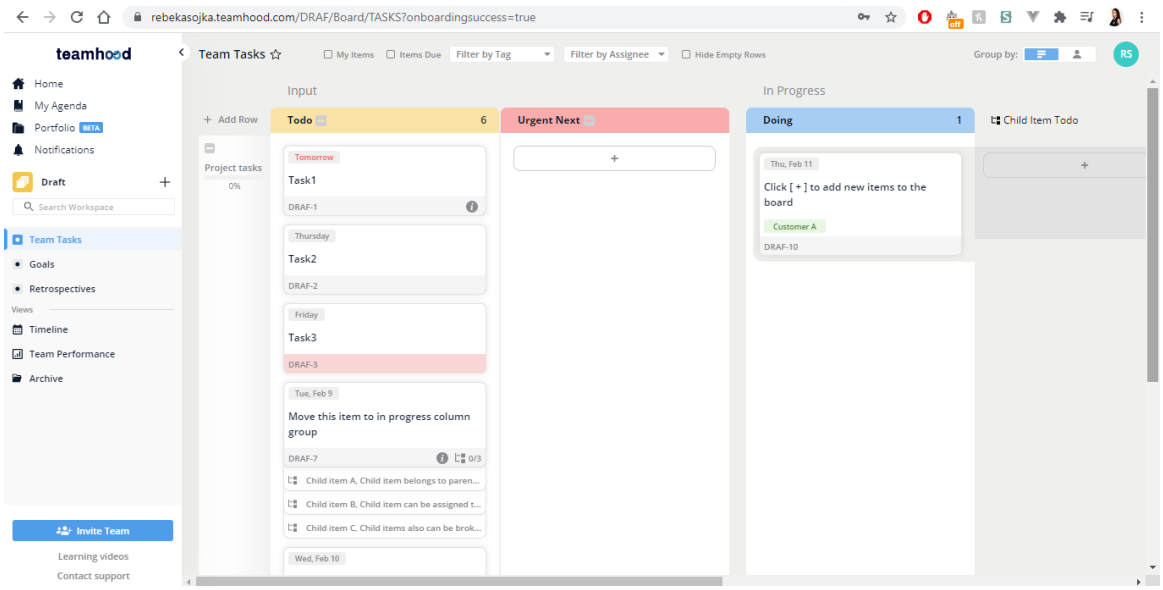
Obrázok 1: Výber účelu pre Teamhood [8]

Potom príde ponuka rozposlať pozvánky pomocou e-mailu a vytvoriť prvé úlohy. Nakoniec sa zobrazí rozpis naplánovaných úloh. Ľavý panel rozhrania slúži na navigáciu medzi projektami, ako vidno na obrázku číslo 2.

Aplikácia ponúka :

- organizačnú tabuľku – stĺpce predstavujú proces a riadky sú na oddelenie tímov, produktov alebo projektov,
- časovú os – pre nastavenie plánu projektu a zaznačenie závislostí,
- dashboard – pre zobrazenie celkového pokroku a odhalenie oneskorených úloh,
- prácu so súbormi na jednom mieste – umožňuje zdieľanie aj editovanie,
- zadelenie úloh členom tímu – administrátor skupiny ovláda celý pracovný priestor a môže podľa potreby prideľovať práva ostatným členom,
- „Moja agenda“ – zobrazenie všetkých úloh prihláseného používateľa na jednom mieste,
- pracovná záťaž – štatistika vykonanej práce pre jednotlivých členov,

a mnoho ďalších nástrojov na prispôsobenie.



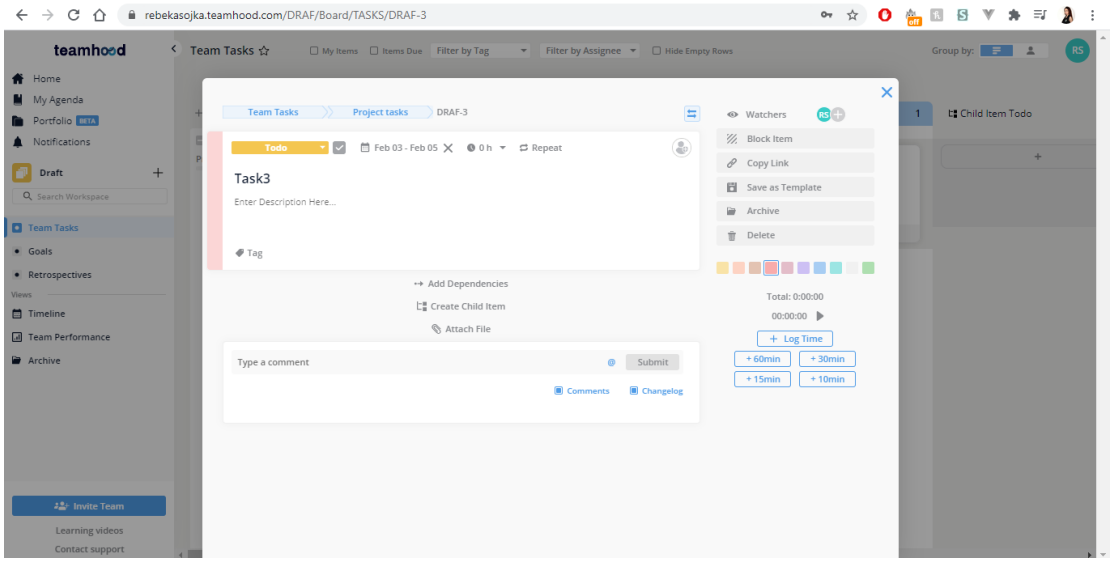
Obrázok 2: Tabuľka úloh Teamhood [8]

Pozitíva :

- multifunkčnosť využitia,
- silný výber funkcionalít,
- miera prispôsobivosti (ktorú je vidieť aj na obrázku číslo 3)
- práca so súbormi na jednom mieste,
- zobrazenie všetkých úloh používateľa na jednom mieste,
- štatistika pracovnej záťaže jednotlivých používateľov, vďaka ktorej sa dá vyhnúť nerovnomernému rozdeľovaniu úloh.

Negatíva :

- hoci množstvo funkcií vnímam ako silné pozitívum, trpí tým používateľské rozhranie, ktoré pôsobí veľmi neprehľadne,
- neintuitívne ovládanie – aplikáciu by som neodporučila menej skúseným používateľom, čo pokladám za veľmi silné negatívum, keďže tento typ aplikácie je smerovaný širokej verejnosti a pokiaľ by som ju chcela použiť na prepojenie viacerých skupín (rodina, práca, hobby), je vysoká pravdepodobnosť, že by samotnú prácu spomalila a zhoršila.

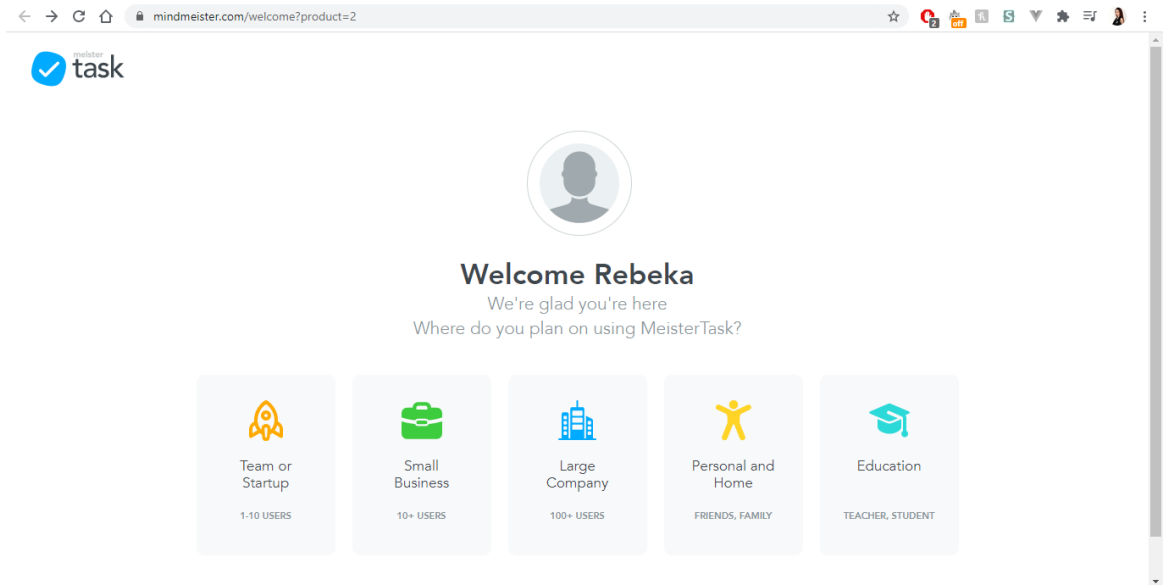


Obrázok 3: Editovanie úloh Teamhood [8]

MeisterTask

Webová aplikácia MeisterTask [9] (základná verzia je bezplatná) pre organizáciu skupiny a jej projektov sa odvoláva na svoj čistý dizajn, kompatibilitu aj so smartfónmi a inteligentnými hodinkami a intuitívne ovládanie.

Po registrácií si používateľ vytvára svoju prvú skupinu a upresňuje, na aké účely plánuje používať MeisterTask, ako je ukázané na obrázku číslo 4.



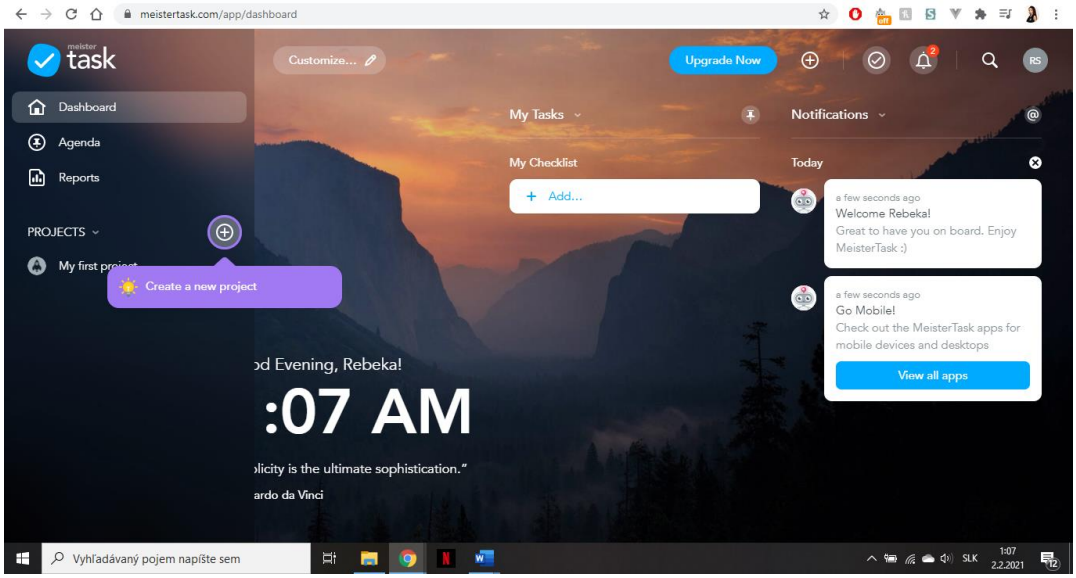
Obrázok 4: Registrácia MeisterTask [9]

Aplikácia ponúka nasledujúcu funkcionalitu.

- Pre projekty: vytváranie sekcií pre prehľadnosť, filtrovanie (podľa prideleného používateľa, tagu, supervízora, zadaného termínu dokončenia, stavu alebo plánu), zobrazenie časovej osi, automatizácia

opakovaných krokov, limitovanie úloh pre jednotlivého používateľa (pre spravodlivé rozdelenie a dodržiavanie termínov).

- Pre úlohy: pridelený používateľ, supervízor, zadané termíny odovzdania, sledovanie času stráveného na úlohe, kontrolný zoznam podúloh, odovzdávanie príloh, tagy, vzájomné závislosti úloh.
- Pre prihláseného používateľa: vlastná nástenka, na ktorú môže pripevniť ľubovoľné komponenty pre sledovanie (nástenka je na obrázku číslo 5).
- Pre analýzu: postup plnenia úloh projektu, časové záznamy.
- Pre administrátora skupiny / projektu: zadeľovanie rolí a povolení, organizácia skupín a tímových projektov.
- Pre komunikáciu: komentáre s notifikáciami, označenie používateľa s notifikáciou, označenie skupiny s notifikáciou.



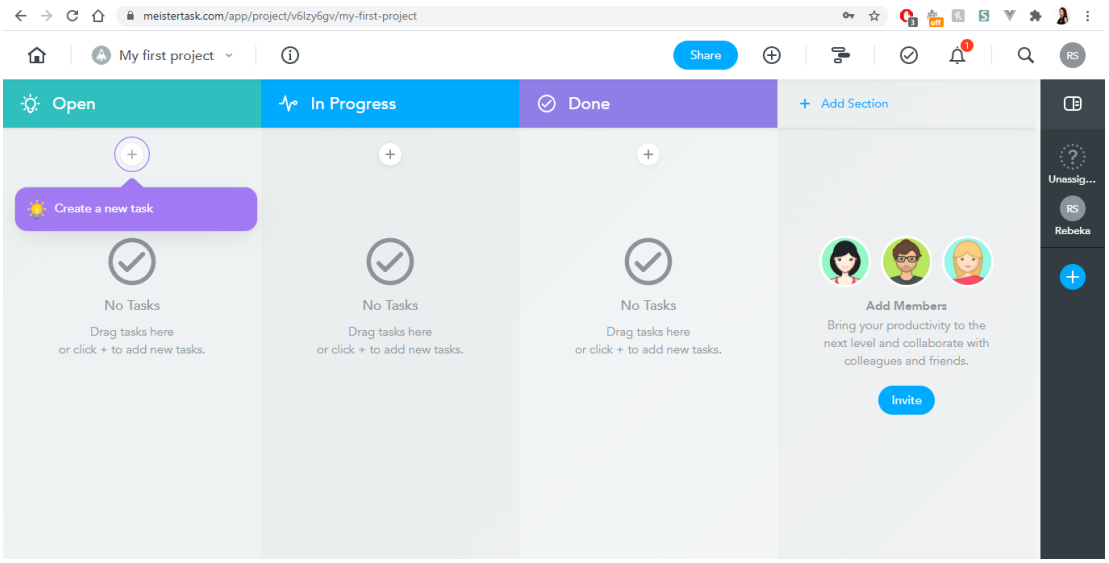
Obrázok 5: Nástenka MeisterTask, [9]

Pozitíva :

- dôraz na používateľské rozhranie a dizajn (ako je vidieť na obrázku číslo 6),
- jednoduchší, ale stabilný výber funkcionalít,
- vysoká miera prispôsobivosti zobrazenia,
- ponuka video návodov priamo v aplikácií pre náročnejšie úkony,
- kompatibilita aplikácie pre IOS aj MS Windows,
- multiplatformová aplikácia,
- zobrazenie všetkých úloh používateľa na jednom mieste,
- automatizácia opakujúcich sa úloh,
- sekcia pre komentáre s notifikáciami,
- rozmiestnenie ikon štandardným spôsobom, čo napomáha intuitívnemu ovládaniu,
- filtrovania a možnosť značenia tagmi.

Negatíva :

- napriek snahe o maximálne zjednodušenie sa, podľa môjho názoru, ešte stále dá hovoriť o presýtenom používateľskom rozhraní, najmä hneď po registrácií, kde sa predpokladá istá skúsenosť s webovými aplikáciami,
- po prvom prihlásení je potrebné vykonať niekoľko nastavení pre praktické zobrazovanie (napríklad vlastných pridelených úloh) nástenky používateľa.

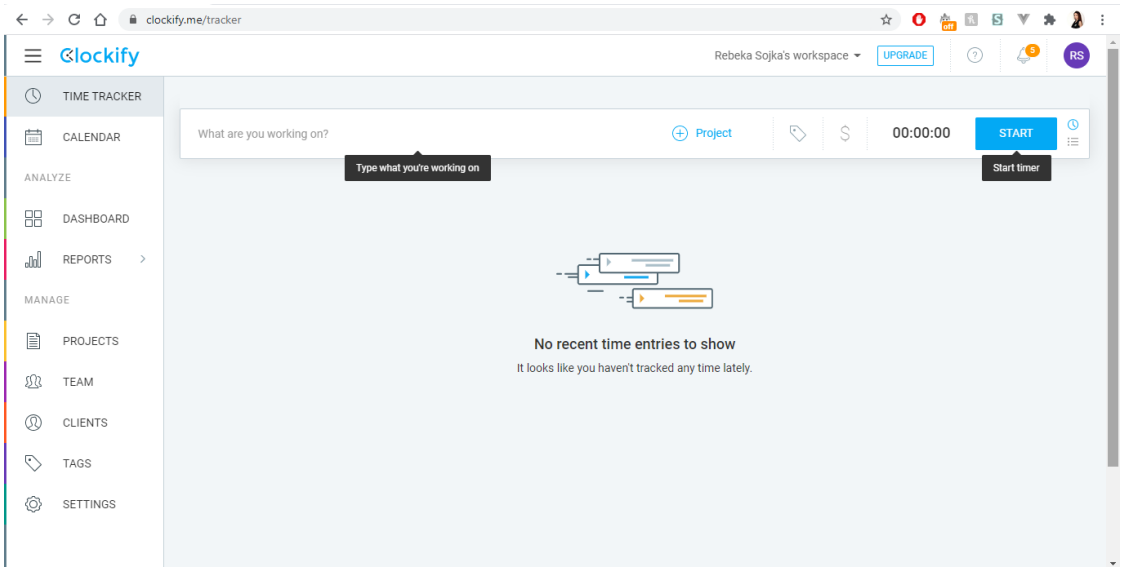


Obrázok 6: Projekt MeisterTask, [9]

Clockify

Webová aplikácia Clockify [10] je zameraná na meranie pracovného času. Podobne ako MeisterTask, aplikácia ponúka časť funkcionalít bezplatne, ale v porovnaní s MeisterTask ich sprístupňuje viac.

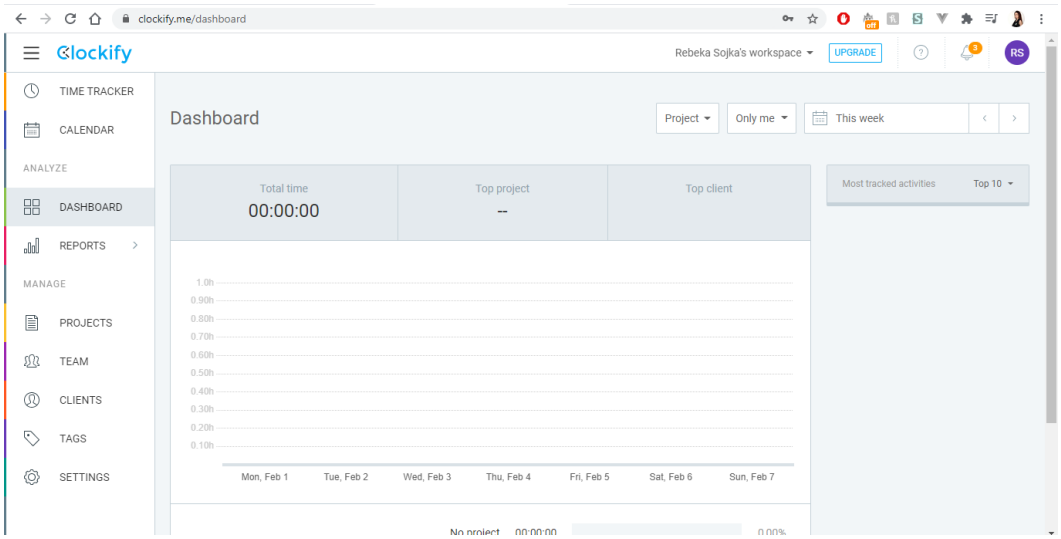
Po registrácií sa používateľovi okamžite zobrazí domovská obrazovka s časovačom, ktorá je na obrázku číslo 7. Navigácia na ľavom paneli obsahuje prepojenia na domovskú stránku s časovačom, kalendár, prehľad práce, analýzu (graf práce s možnosťou filtrovania), projekty, tímy, klientov, tagy a používateľské nastavenia.



Obrázok 7: Časovač Clockify [10]

Aplikácia ponúka:

- časovač – pomocou priameho merania alebo manuálneho vkladania s možnosťou vyčíslenia hodinovej mzdy,
- prezenčnú listinu – značenie aktivít, poznámok, výpočet odpracovaných hodín, šablóny pre nasledujúce týždne,
- nástenku (nachádzajúcu sa na obrázku číslo 8) – prehľad vlastnej práce a práce tímu, top aktivity, viditeľné grafy, aktuálny status kto pracuje na čom,
- analýzu – prehľad podľa dní, aktivít, a používateľov s možnosťou exportu do formátu PDF, CSV a Excel,
- projekt – pre sledovanie stráveného času na projektoch, pokroku a rozpočtu,
- tím (je vidieť na obrázku číslo 9)– neobmedzené množstvo členov, mzdy na hodinu, používateľská rola manažéra.



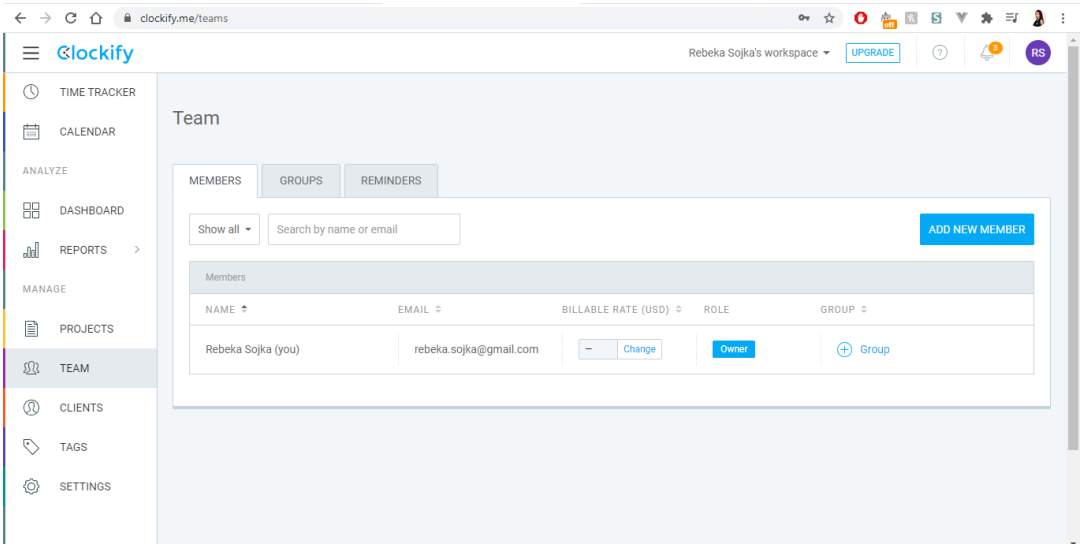
Obrázok 8: Nástenka Clockify [10]

Pozitíva:

- jednoduchý konzistentný dizajn,
- prehľadné používateľské rozhranie,
- intuitívne ovládanie,
- vhodné pre širokú verejnosť,
- ponúka veľké množstvo funkcionalít,
- kompatibilita aplikácie pre IOS, MS Windows aj Linux,
- multiplatformová aplikácia,
- tutoriál videá.

Negatíva:

- pre lepšiu čitateľnosť v znevýhodnených podmienkach alebo pre znevýhodnených používateľov, by som zvolila kontrastnejšie farby, prípadne – farby majú tendenciu splývať, ako písmo s pozadím v hlavičke tabuľky na obrázkoch číslo 8 aj 9.



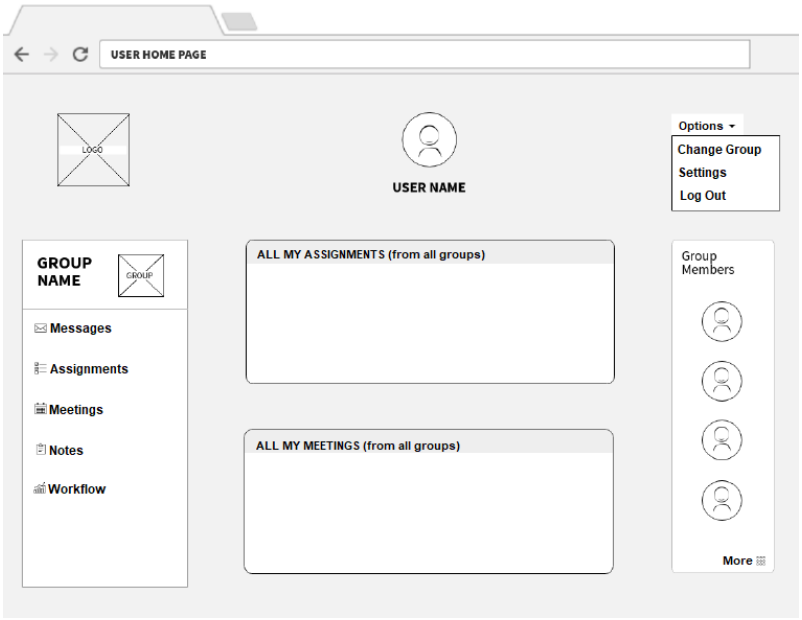
Obrázok 9: Tímy Clockify, [10]

2 Návrh

Táto kapitola obsahuje náčrt používateľského rozhrania. Ďalej vysvetľujem databázový model aplikácie a pomocou UML use-case diagramu predstavujem používateľské role. Posledná podkapitola opisuje využitie frameworkov Laravel a Vue.js pre backend a frontend.

2.1 Používateľské rozhranie

Pri tvorbe aplikácie som venovala veľkú pozornosť aj návrhu používateľského rozhrania, ktoré môže byť pre používateľa rozhodujúce. Nevyhnutnými podmienkami boli: rýchla navigácia, intuitívne umiestnenie komponentov a čistý dizajn (vyhýbajúc sa presýteniu tlačidlami a iným rušivým elementom). Náčrt, ktorý sa nachádza v tejto podkapitole (obrázok 10), vznikol ešte pred samotnou tvorbou aplikácie. V nasledujúcom popise sú vymenované klady tohto návrhu, ale taktiež odôvodnené zmeny voči finálnej podobe. Na konci podkapitoly je zobrazená finálna verzia používateľského rozhrania (obrázok 11).



Obrázok 10: Návrh GUI

Navigácia

Jednotlivé možnosti „Options“ boli pôvodne dostupné až po rozkliknutí. Ako komfortnejšie riešenie som zvolila alternatívu samostatných tlačidiel na horizontálnom navigačnom paneli (ako vidno na obrázku 11). Taktiež zmenu aktuálne zobrazovanej skupiny bolo nevyhnutné urýchliť. Vo finálnej verzii to môže používateľ vykonať pomocou bývalého elementu „Options“ (HTML select option tag). Namiesto možností, ktoré vidno na obrázku, sa mu však zobrazia mená skupín, medzi ktorými môže jednoducho prepínať.

Panel skupiny (ľavý panel)

Prvým tlačidlom na paneli je tlačidlo so znakom „+“ , ktoré slúži na vytvorenie novej skupiny. Ľavý panel ďalej obsahuje avatar skupiny, názov skupiny a prepojenia na podstránky skupiny: nástenka, úlohy, udalosti. Pre administrátora je tu naviac štatistika pracovnej záťaže a tlačidlo s ikonou ceruzky pre editovanie údajov skupiny. Poznámky a čet správy sú premiestnené z dôvodu urýchlenia prístupu v dolných opozitných rohoch okna.

Panel členov skupiny (pravý panel)

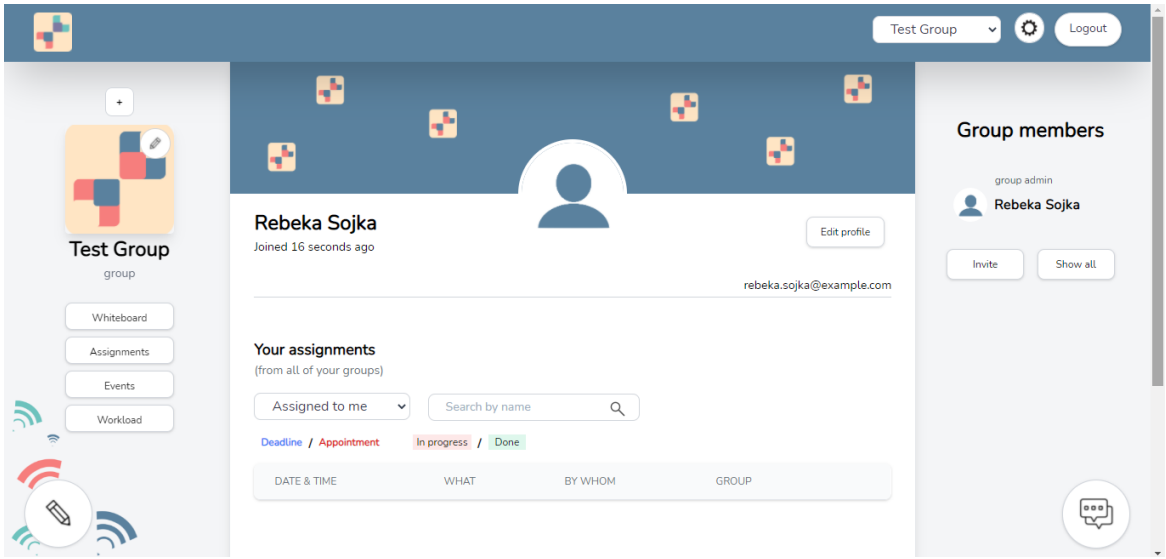
V priestore pravého panelu sa zobrazuje päť používateľov. Prvým používateľom je spravidla administrátor, aby ho používateľ nemusel viackrokovo vyhľadávať. Zostávajúci štyria používatelia sú náhodne zvolení zo zoznamu členov skupiny. Na konci panelu sú dve tlačidlá :

1. „Show all“ (pôvodne „More“) pre zobrazenie všetkých členov skupiny,
2. „Invite“ pre administrátora (možnosť pozvať nových členov) alebo „Leave“ pre bežného člena (možnosť opustiť skupinu).

Hlavný panel a domovská stránka

Za jednu z hlavných výhod tejto aplikácie považujem obsah domovskej stránky, ktorá sa zobrazuje vždy po prihlásení na hlavnom paneli. Pod svojím profilom (meno, avatar, banner, e-mail) tu používateľ nájde všetky svoje úlohy a všetky svoje udalosti zo všetkých skupín. Teda sa od neho nevyžaduje žiadne ďalšie vyhľadávanie, čím sa významne šetrí používateľov čas.

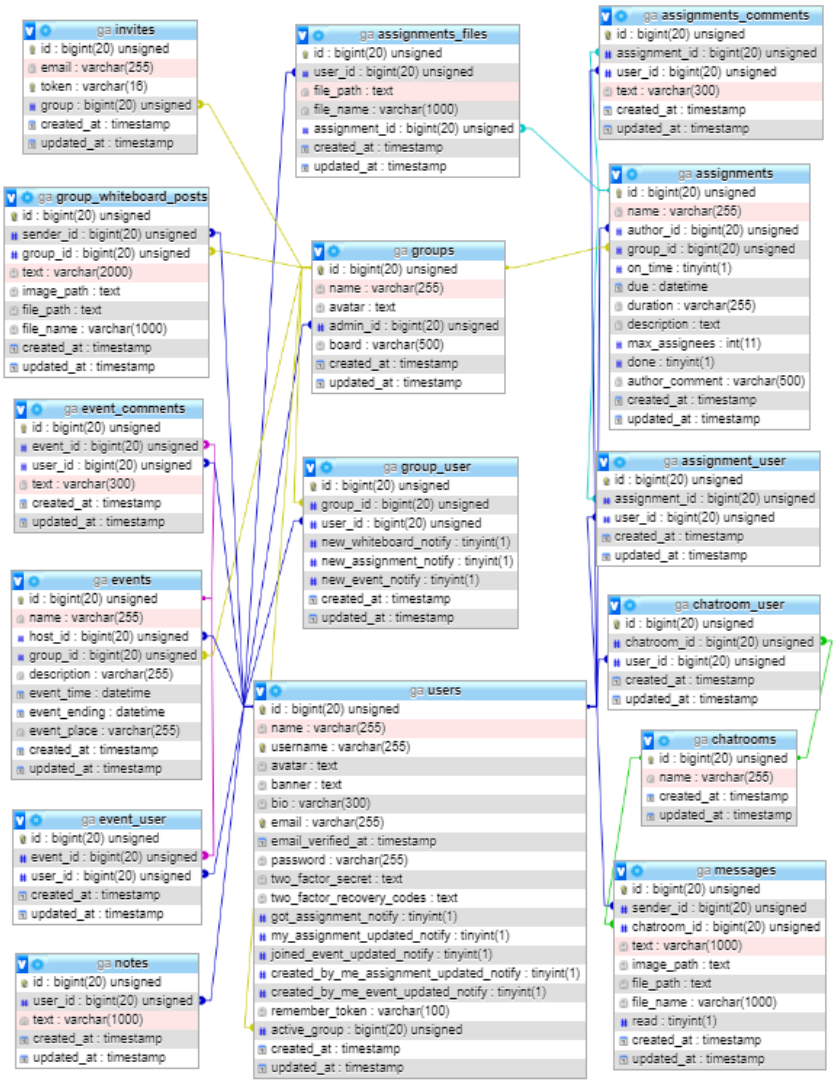
Na obrázku 11 je finálna verzia používateľského rozhrania aplikácie.



Obrázok 11: Konečná verzia GUI

2.2 Databázový model

V tejto podkapitole je vysvetlená finálna verzia relačného databázového modelu, ktorý som pre túto aplikáciu vytvorila a popis jednotlivých tabuliek.



Obrázok 12: Databázový model

Na obrázku 12 sa nachádzajú tabuľky, ktorých význam úzko súvisí s logikou stavebných základov aplikácie. Nenachádzajú sa tu pomocné tabuľky frameworku Laravel, slúžiace napríklad na sledovanie, ktoré tabuľky sa nachádzajú v databáze (tabuľka **migrations** obsahujúca uskutočnené migrácie) a podobne. Nasleduje popis jednotlivých tabuliek.

Users

Tabuľka **users** obsahuje základné údaje o používateľoch a ich kontách. O každom z nich tabuľka okrem základných údajov (meno, heslo, e-mail, avatar a pod.) obsahuje aj údaje o tom, kedy si vytvoril profil a či si želá dostávať upozornenia na e-mail v prípade že:

- mu bola zadaná úloha,
- bola vykonaná zmena ohľadom úlohy, ktorá mu je zadaná,
- bola vykonaná zmena ohľadne udalosti, ktorej sa plánuje zúčastniť.

Dôležitým údajom, ktorý sa v tejto tabuľke nachádza, je stĺpec *active_group*. Na základe toho sa používateľovi po prihlásení zobrazujú najmä dáta tej skupiny, ktorú aktuálne sleduje. Používateľské rozhranie je navrhnuté tak, aby sa toto nastavenie dalo bez zdržania meniť na domovskej stránke.

Groups

Táto tabuľka obsahuje základné údaje o skupine:

- kto je jej administrátor,
- kedy bola vytvorená,
- špecifiká, napríklad avatar skupiny.

Pre zapamätanie vzťahu, ktorí používatelia do skupiny patria, slúži pivotná tabuľka **group_user**. V tej je zároveň, pre úsporu času a lepšiu čitateľnosť kódu, uvedená aj informácia, či si daný používateľ želá dostať e-mailovú notifikáciu v prípade, že sa v rámci tejto skupiny stane jedno z tu uvedených:

- pribudol nový príspevok na nástenke,
- pribudla nová úloha,
- pribudla nová udalosť.

Invites

Pozvánky do skupín prichádzajú používateľom výhradne na e-mailovú adresu. Tým sa zabezpečí zachovanie ukrytej existencie profilov mimo skupinu. Táto tabuľka uchováva o každej pozvánke tieto údaje: do ktorej skupiny pozvánka patrí, komu bola určená (e-mailová adresa) a bezpečnostný token, ktorý po využití pozvánky kontroluje, či ide o oprávnenú osobu.

Group whiteboard posts

V tejto tabuľke sa uchovávajú jednotlivé príspevky na nástenke skupiny. Môžu obsahovať text, ale aj obrázok a súbor.

Assignments

Tabuľka **assignments** uchováva úlohy, ktoré nie sú zadané žiadnemu (voľné úlohy) alebo jednému a viac členom. Obsahuje stĺpec *max_assignees*, ktorý môže používateľ pri tvorbe novej úlohy nastaviť, a tak kontrolovať maximálny počet riešiteľov. Uchovávajú sa aj ukončené úlohy. Či je riešenie úlohy uzavreté, je uvedené v stĺpci *done*. Každá úloha nesie aj údaj o dátume a čase. Či ide o hraničný dátum „deadline" alebo sa má úloha vykonať práve v tom uvedenom čase, teda ide o „appointment", informuje stĺpec *on_time*, ktorý nadobúda hodnoty 1 a 0. Ak sa v stĺpci nachádza 1, to znamená, že úloha sa má vykonávať v tom čase. Naopak, ak je v stĺpci 0, tak úloha má nastavený deadline. Tiež je možné nastaviť dobu trvania úlohy (stĺpec *duration*) a autor môže pridať a meniť súkromný komentár pre riešiteľov (*author_comment*). Pre priradenie riešiteľov existuje pivotná tabuľka **assignment_user**.

Assignments files

Ku každej úlohe je možné pripojiť viacero súborov, či už z dôvodu zadania, alebo riešenia. Tieto súbory sa uchovávajú v tabuľke **assignments_files**.

Assignments comments

Každá úloha poskytuje aj priestor na verejnú diskusiu prostredníctvom sekcie pre komentáre, ktoré sa uchovávajú v tabuľke **assignments_comments**.

Events

Pre vytvorenie udalosti sa vyžaduje niekoľko povinných údajov: názov udalosti, miesto a čas jej konania. Používateľ, ktorý ju vytvoril, sa automaticky stáva jej organizátorom (uchováva sa v stĺpci *host_id*) a účastníkom (svoju účasť môže neskôr zrušiť). Ďalej je možné pridať popis udalosti (stĺpec *description*) a tiež dátum a čas jej ukončenia (*event_ending*). Ktorý používateľ sa udalosti plánuje zúčastniť, sa zaznamenáva v pivotnej tabuľke **event_user**.

Event comments

Rovnako ako pri úlohách, aj pri udalostiach je vyhradený priestor na diskusiu pomocou komentárov. Tie je možné prečítať z tabuľky **event_comments**.

Chatrooms

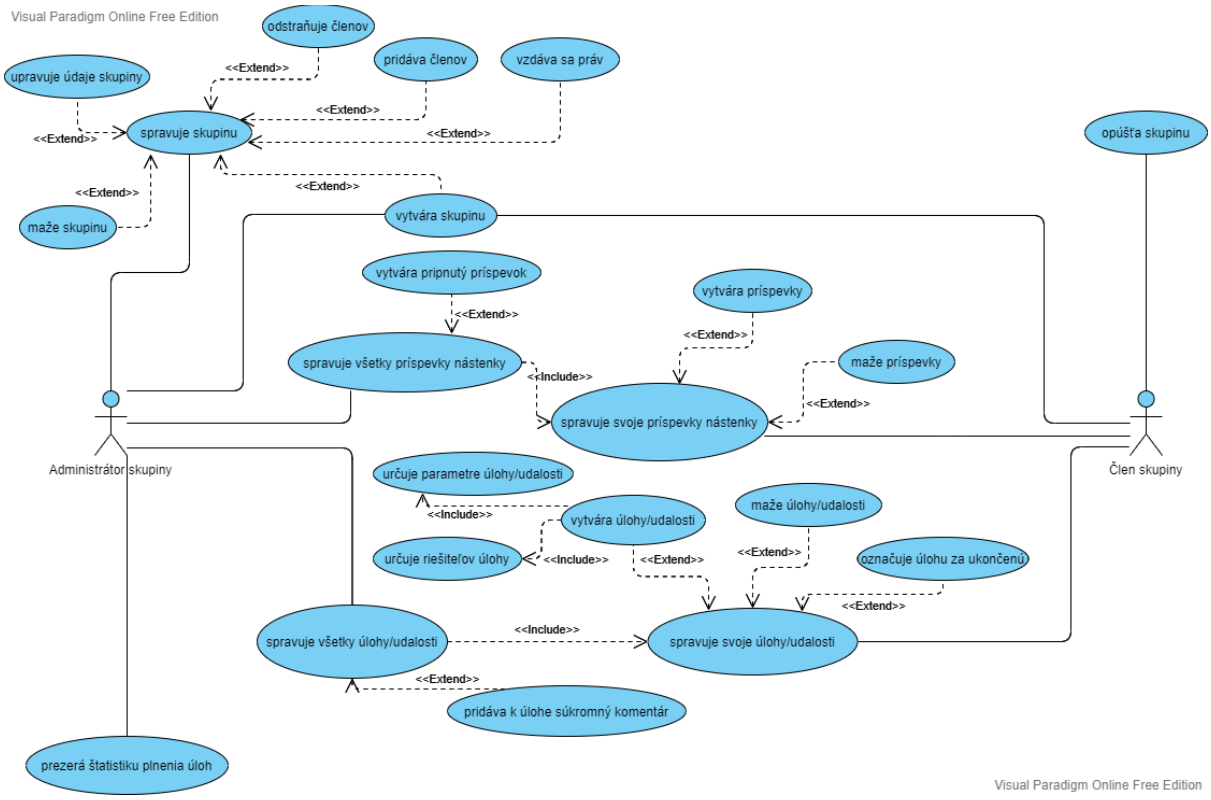
Okrem verejných priestorov na komunikáciu v reálnom čase (bez opätovného načítavania), ako je nástenka skupiny a komentáre pod jednotlivými udalosťami a úlohami, ponúka aplikácia aj súkromné správy medzi používateľmi. Četové miestnosti sa uchovávajú v tabuľke **chatrooms**. Jednotlivé správy v tabuľke **messages**, (ktorých text je šifrovaný – jedna z podmienok pre neskoršie testovanie) a vzťahy medzi používateľmi a miestnosťami sa nachádzajú v **chatroom_user**.

Messages

Súkromné správy môžu obsahovať text, obrázky do 5 MB (zobrazené) a tiež je dovolené priložiť súbor do veľkosti 25 MB. Všetky údaje je možné prečítať z tabuľky **messages**. Text správy je v databáze zašifrovaný a odšifrovaný až pred zobrazením používateľovi.

2.3 Používateľské role

Používateľské role v tejto podkapitole predstavím prostredníctvom UML use-case diagramu (obrázok 13) s vysvetľujúcim popisom.



Obrázok 13: Use-case diagram

Aplikáciu môže používať len prihlásený používateľ. V rámci každej skupiny vystupujú dva typy používateľov: administrátor skupiny a člen skupiny (teda „bežný člen skupiny“ bez práv administrátora). V nasledujúcich odsekoch popíšem jednotlivé funkcie aplikácie z pohľadu oboch typov používateľov za pomoci diagramu na obrázku 13.

Vytváranie nových skupín

Vytvoriť skupinu môže každý používateľ. Po vytvorení skupiny sa jej autor automaticky stáva jej administrátorom.

Správa skupiny

Administrátor spravuje skupinu. Teda má prístup k jej špecifickým nastaveniam, ako sú: názov skupiny, avatar skupiny a údaj o tom, kto je administrátor. Okrem týchto nastavení môže administrátor, ako jediný, pozývať do skupiny nových členov alebo existujúcich členov zo skupiny odstrániť. Skupinu môže administrátor taktiež natrvalo vymazať spolu so všetkými jej dátami.
Ku žiadnym z týchto funkcionalít bežný používateľ nemá prístup.

Opustenie skupiny

Skupina nemôže byť bez administrátora. Teda opustiť skupinu môže len jej bežný člen. Pokiaľ chce administrátor skupinu zanechať, musí v prvom kroku svoje práva odovzdať inému členovi. To je možné zmenou nastavení danej skupiny. Potom sa stáva jej bežným členom a môže skupinu opustiť.

Nástenka skupiny a jej príspevky

Každá skupina má vyhradený priestor pre skupinovú konverzáciu, kde je možné zdieľať aj obrázky a súbory. Každý používateľ môže svoj vlastný príspevok vytvoriť aj zmazať. Administrátor skupiny môže vymazať každý príspevok – bez ohľadu na autora. Nástenka má aj priestor na zvýraznený „pripnutý“ príspevok, ktorý môže editovať len administrátor.

Úlohy

Novú úlohu môže vytvoriť a zadať ľubovoľným členom skupiny každý používateľ. Vlastné úlohy, ktoré používateľ vytvoril, môže editovať, mazať alebo označiť za ukončené. Pokiaľ je úloha voľná (teda počet riešiteľov je menší, než je ich nastavený maximálny počet), tak sa môže ktorýkoľvek používateľ prihlásiť ako ďalší nový

riešiteľ. Administrátor, ktorý má vyššie práva, môže spravovať nie len ním vytvorené, ale všetky úlohy: teda každú môže editovať, mazať alebo označiť ako ukončenú.

Udalosti

Rovnako, ako úlohy, udalosti môže vytvoriť každý používateľ. Editovať a mazať môže bežný člen len tie udalosti, ktoré sám vytvoril. Administrátor môže zasahovať do všetkých udalostí. Na udalosť je pozvaný každý člen skupiny – potvrdiť svoju účasť môže ktorýkoľvek používateľ.

Štatistika plnenia úloh

Pre lepšie informované rozhodnutia ponúka aplikácia administrátorovi (výlučne jemu) možnosť zobraziť štatistiku plnenia úloh. Táto funkcia graficky a percentami informuje o pracovnej záťaži jednotlivých členov. Vďaka tejto štatistike sa potom administrátor skupiny môže lepšie rozhodovať komu pridať, resp. nepridať ďalšiu zodpovednosť.

2.4 Architektúra aplikácie

Pre vývoj aplikácie som sa rozhodla využiť dva rôzne frameworky: Laravel pre backend a Vue.js pre frontend. V tejto podkapitole je objasnené ich využitie pri tvorbe mojej aplikácie.

Backend

Komunikáciu s databázou som zabezpečila pomocou Laravel frameworku, čo znamená pomocou jazyku php. Na tejto úrovni sa realizujú dopyty, validujú sa vstupy a ukladajú sa do databázy. Ďalej sa vyhodnocujú cesty a presmerovania v rámci aplikácie. Za pomoci objektov triedy Migration sa ukladá databázový model. Významnou zodpovednosťou, ktorú manažuje backend, je aj kontrola autentifikácie prihláseného používateľa.

Frontend

Reaktívnosť frameworku Vue.js som považovala za neodmysliteľnú súčasť behu aplikácie. Reaktívny framework je taký, ktorý udržiava používateľské rozhranie zosynchronizované s dátami. Tak sa akákoľvek zmena zobrazuje okamžite bez opätovného načítania. Vďaka tomu sa v aplikácií nachádzajú aj funkcionality typu čet a komentáre, ktoré sa odohrávajú v reálnom čase – pričom ide o prichádzajúce aj odchádzajúce dáta.

2.5 Požiadavky na server

Nevyhnutné požiadavky na server:

- Podpora Laravel aplikácie (PHP >= 7.3, BCMath PHP Extension, Ctype PHP Extension, Fileinfo PHP Extension, JSON PHP Extension, Mbstring PHP Extension, OpenSSL PHP Extension, PDO PHP Extension, Tokenizer PHP Extension, XML PHP Extension) [14]
- Podpora MySQL databázy
- Podpora SMTP

3 Implementácia

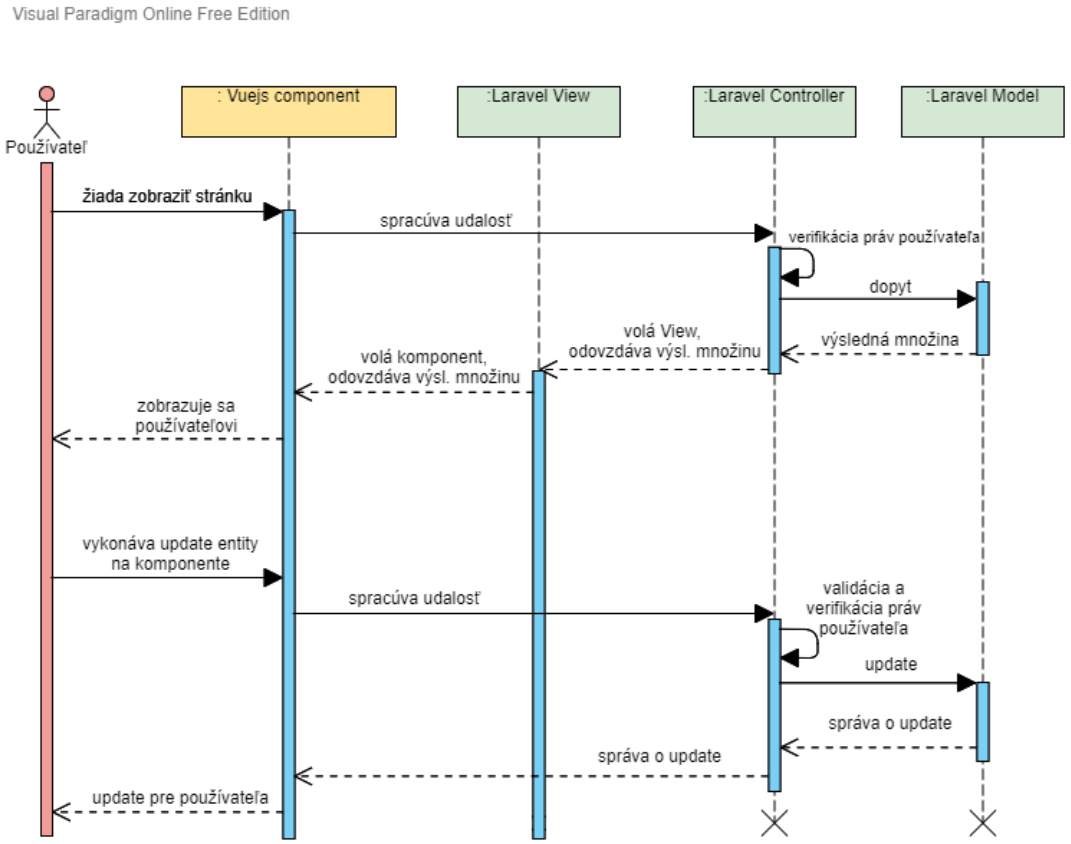
Táto kapitola sa venuje podrobnému opisu implementácie MVC modelu a architektúre aplikácie.

3.1 MVC model

V tejto podkapitole je špecifikovaná implementácia jednotlivých častí MVC modelu, ktoré sú: Model, View a Controller. Ich využitie spoločne s Vue.js je znázornené sekvenčným diagramom.

Sekvenčný diagram

Tento UML diagram (obrázok 14) je zovšeobecnením všetkých scenárov, kde používateľ žiada zobraziť podstránku aplikácie a následne vykoná zmenu, ktorú je potrebné zapísať do databázy (napríklad edituje svoj profil). Ide o proces vzájomného prijímania a odovzdávania dát medzi MVC modelom frameworku Laravel a Vue.js komponentom.



Obrázok 14: Sekvenčný diagram

V tomto scenári používateľ klikol na odkaz a bol presmerovaný na podstránku aplikácie. Táto žiadosť sa z frontendu presmeruje na príslušný Controller, ktorý

pomocou súvisiaceho Modelu vykoná databázový dopyt. Ten sa vo forme výslednej množiny pošle do požadovaného View, ktorý ju odovzdá aj Vue.js komponentu. Ten sa, ako súčasť View, zobrazí používateľovi.

Ďalej používateľ vykoná update (napríklad zmena prihlasovacích údajov). Vue.js komponent túto udalosť pošle Controlleru, kde sa pomocou Modelu vykoná update do databázy. Z nej príde správa o vykonaných zmenách. Tá je odovzdaná priamo na frontend, teda Vue.js komponent, ktorý informuje používateľa, že akcia je ukončená.

Model

Objekt triedy, ktorá plní funkciu modelu, reprezentuje riadok tabuľky. Spôsob využitia popíšem na nasledujúcom príklade triedy Group extends Model.

```php
/**
 * The attributes that are mass assignable.
 *
 * @var array
 */
protected $fillable = [
    'name',
    'avatar',
    'admin_id',
    'board'
];
```

Obrázok 15: Model – pole $fillable

Jedno zo základných nastavení Modelu je jeho pole „fillable“ (obrázok 15). Ide o zoznam atribútov (stĺpcov tabuľky), ktoré je možné upraviť pomocou jedného databázového dopytu. V opačnom prípade takúto zmenu Laravel nepovolí a nové hodnoty sa neuložia.

Model je prostredníkom medzi dátami v databáze a dátami, s ktorými pracuje aplikácia. Obsahuje metódy pre spracovanie hodnoty v stĺpci alebo pre získanie iných objektov typu Model, medzi ktorými je v databáze vzťah.

Laravel uvádza v dokumentácií [14], ako vyjadriť rôzne vzťahy medzi objektami pomocou funkcií. Z nich som využila:

- **One To One**
  - Reprezentuje v entitno-relačnom diagrame vzťah 1:1
  - „Čet má jednu poslednú správu“
    return $chat->**hasOne**(Message::class)->latest()
- **One To Many**
  - Reprezentuje v entitno-relačnom diagrame vzťah 1:n

- „Udalosť patrí jednej skupine a zároveň skupina má veľa udalostí“
  return $event->**belongsTo**(Group::class)

  return $group->**hasMany**(Event::class)

- **Many To Many**
  - Reprezentuje v entitno-relačnom diagrame vzťah n:n
  - „Používateľ patrí do veľa skupín a zároveň skupina má veľa používateľov“
    return $user->**belongsToMany**(Group::class)->withPivot(...)

    return $group->**belongsToMany**(User::class)->withPivot(...)

```php
public function getAvatarAttribute($value){
    return asset($value ? 'storage/'.$value : '/img/default_group.png');
}

public function users(){
    return $this->belongsToMany(User::class)->withPivot('new_whiteboard_notify', 'new_assignment_notify', 'new_event_notify');
}

public function hasUser($user) {
    return $this->users->contains($user);
}

public function events(){
    return $this->hasMany(Event::class);
}
```

Obrázok 16: Model – vzťahy

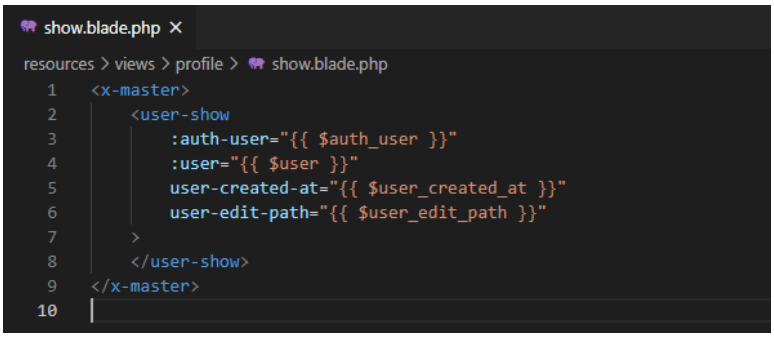
Na obrázku 16 sú ďalšie funkcie triedy Group extends Model.

- Funkcia *getAvatarAttribute($value)* spracúva nepovinný údaj o skupine – vlastný avatar skupiny.
- Funkcia *users()* vráti všetkých používateľov, ktorí patria do danej skupiny. Vzťah „používateľ patrí do skupiny“ je uchovávaný v pivotnej tabuľke. Keďže ide o vzťah „many to many“ (každá skupina môže mať viacero používateľov a každý používateľ môže byť vo viacerých skupinách), tak v tejto funkcií využívam vzťah belongsToMany a chcem získať aj dáta z pivotnej tabuľky – preto *withPitvot(...)*.
- Funkcia *hasUser($user)* vykoná na pozadí databázový dopyt, ktorým získa používateľov skupiny a potom skontroluje či používateľ v argumente funkcie do danej skupiny patrí.
- Posledná funkcia na obrázku *events()* vyjadruje vzťah One To Many v zmysle „skupina má veľa udalostí“.

View

Časť aplikácie, ktorá sa zobrazuje používateľovi vo frameworku Laravel, zabezpečuje šablónovací systém Blade, ktorý spravuje všetky View s príponou .blade.php. Ten by bez reaktívneho Vue.js komponentu vyžadoval po každej zmene opätovné načítanie. Môže existovať samostatne alebo sa môže skladať z viacerých vnorených komponentov. Obsahuje html kód s načítavaním súborov (JavaScript a CSS). V mojej aplikácií využívam ako kostru stránky master.blade.php, do ktorého vkladám jednotlivé komponenty podľa aktivity používateľa. Vďaka tomu sa minimalizuje duplicita kódu a zvyšuje sa jeho čitateľnosť. Jednotlivé blade komponenty, týkajúce sa špecifického obsahu aplikácie, niekedy obsahujú len príkaz pre vloženie Vue.js komponentu spolu s dátami, ktoré má zobraziť.

Napríklad komponent ../profile/show.blade.php na obrázku 17 je vložený do html kostry master.blade.php (<x-master>) a obsahuje volanie <user-show> Vue.js komponentu. Tomuto komponentu odovzdáva hodnoty, ktoré dostal z Controlleru.



Obrázok 17: Komponent ../profile/show.blade.php

Controller

Podľa konvencie sa píše [Množné číslo názvu Modelu]Controller (napríklad UsersController), podľa Modelu (resp. entity), ku ktorému patrí. Controller má aj verifikačnú úlohu. V každej metóde kontroluje, či má používateľ právo vykonať zmeny alebo zobraziť požadované dáta. Všetky viackrokové databázové dopyty sú chránené vhodným nastavením izolovanosti daných databázových transakcií. Okrem verifikácie práv, vykonáva aj validáciu dát, ktoré chce používateľ vložiť do tabuľky.

Podľa konvencií sú definované tieto funkcie:

- *index()* – zavolá View, kde sa zobrazí vo vhodnej forme viacero riadkov danej tabuľky (napríklad zoznam členov skupiny),
- *create()* – zobrazí formulár na vytvorenie nového záznamu,
- *store()* – uloží novovytvorený záznam do tabuľky,
- *show()* – zavolá View, kde sa zobrazí vo vhodnej forme konkrétny riadok tabuľky (napríklad profil používateľa),

- *edit()* – zobrazí formulár na editovanie existujúceho záznamu,
- *udpate()* – uloží editovaný záznam do tabuľky,
- *destroy()* – vymaže existujúci záznam z tabuľky.

Podľa povahy projektu pribúdajú aj ďalšie pomocné funkcie, napríklad funkcia *newAssignmentNotifyUsers(Assignment $assignment)* na obrázku 18. Táto funkcia po vytvorení novej úlohy pošle e-mail všetkým používateľom, ktorí majú nastavené notifikácie.

```php
public function newAssignmentNotifyUsers(Assignment $assignment){
    $notify_members = $assignment
                        ->group
                        ->users()
                        ->select('email')
                        ->where('new_assignment_notify', true)
                        ->where('users.id', '!=', auth()->user()->id)
                        ->get();
    $notify_assignees = $assignment
                            ->users()
                            ->select('email')
                            ->where('got_assignment_notify', true)
                            ->where('users.id', '!=', auth()->user()->id)
                            ->get();

    Mail::to($notify_members)
            ->send(new NewWhiteboardEventAssignmentMail($assignment->group, 'assignments', $assignment))
    ;
    Mail::to($notify_assignees)
            ->send(new FromAllGroupsNotificationMail($assignment->group->name, 'You have a new assignment', 'assignments/' . $assignment->id))
    ;
    return;
}
```

Obrázok 18: Definícia funkcie newAssignmentNotifyUsers

3.2 Architektúra aplikácie

V tejto podkapitole popisujem, ako fungujú prepojenia v rámci aplikácie, verifikácia prihlásenia používateľov, registrácia Vue.js komponentov, axios požiadavky, odosielanie e-mailov, vytváranie a broadcasting udalostí a nastavenie premenných prostredia.

Middleware a web.php

Všetky cesty, prepojenia, sú registrované v súbore web.php. Tu sa nachádzajú inštrukcie, ktorý Controller má reagovať na ktoré URL cesty, a ktorú funkciu má zavolať. Je tu nastavený aj Middleware – odkazy môžu a nemusia do neho patriť. Tie, ktoré nie sú chránené, môže navštíviť aj neprihlásený používateľ. Naopak tie odkazy, ktoré patria do Middleware skupiny, sú výlučne pre prihlásených.

Príklad ciest, ktoré sa týkajú používateľských profilov, sú na obrázku 19.

```php
Route::get('/profile/{user:username}', [UsersController::class, 'show'])->name('profile');
Route::get('/profile/{user:username}/edit', [UsersController::class, 'edit']);
Route::post('/profile/{user:username}/edit', [UsersController::class, 'update']);
Route::get('/profile/{user:username}/events/all', [UsersController::class, 'getAllUserEvents']);
Route::get('/profile/{user:username}/events/joined', [UsersController::class, 'getUserJoinedEvents']);
Route::get('/profile/{user:username}/assignments/all', [UsersController::class, 'getAllUsersAssignments']);
Route::get('/profile/{user:username}/assignments/mine', [UsersController::class, 'getUsersAssignments']);
Route::get('/{user:username}/groups', [UsersController::class, 'getAllUsersGroups']);
Route::get('/groups/{group:id}/all-members', [UsersController::class, 'index']);
Route::get('/{user:username}/active-group', [UsersController::class, 'getActiveGroup']);
Route::get('/profile/{user:id}/settings', [UsersController::class, 'showSettings']);
Route::post('/profile/{user:id}/settings', [UsersController::class, 'updateSettings']);
Route::delete('/profile/{user:id}/delete-profile', [UsersController::class, 'destroy']);
```

Obrázok 19: Cesty týkajúce sa používateľských profilov

Vue.js komponenty a app.js

V súbore app.js sú registrované všetky Vue.js komponenty. Každý komponent má svoju cestu a svoje meno, ktorého výskyt v blade súbore je automaticky odkazom na správny Vue.js komponent. Na obrázku 20 sú všetky Vue.js komponenty aplikácie a ich mená sú na obrázku 21.

```
1  import Vue from 'vue';
2  import navbar from './components/navbar/navbar.vue';
3  import messagesLink from './components/navbar/messages-link.vue';
4  import groupSelection from './components/navbar/group-selection.vue';
5  import groupPanel from './components/group-panel.vue';
6  import emptyGroupPanel from './components/empty-group-panel.vue';
7  import membersPanel from './components/members-panel.vue';
8  import userShow from './components/users/user-show.vue';
9  import userEdit from './components/users/user-edit.vue';
10 import userSettings from './components/users/user-settings.vue';
11 import groupUsers from './components/users/group-members.vue';
12 import inviteUsers from './components/users/invite-members.vue';
13 import userEvents from './components/users/user-events.vue';
14 import userAssignments from './components/users/user-assignments.vue';
15 import notes from './components/notes/note-index.vue';
16 import events from './components/events/group-events.vue';
17 import eventShow from './components/events/event-show.vue';
18 import eventEdit from './components/events/event-edit.vue';
19 import eventsTable from './components/events/events-table.vue';
20 import eventComments from './components/comments/event-comments.vue';
21 import assignments from './components/assignments/group-assignments.vue';
22 import assignmentShow from './components/assignments/assignment-show.vue';
23 import assignmentFileUpload from './components/assignments/assignment-show-file-upload.vue';
24 import assignmentEdit from './components/assignments/assignment-edit.vue';
25 import assignmentsTable from './components/assignments/assignments-table.vue';
26 import assignmentComments from './components/comments/assignment-comments.vue';
27 import chatroomsIndex from './components/chatrooms/chatrooms-index.vue';
28 import chatroomsShow from './components/chatrooms/chatrooms-show.vue';
29 import groupEdit from './components/groups/group-edit.vue';
30 import groupWhiteboard from './components/groups/group-whiteboard.vue';
31 import groupStatistics from './components/groups/group-statistics.vue';
```

Obrázok 20: Vue.js cesty

```
Vue.component('navbar', navbar);
Vue.component('messages-link', messagesLink);
Vue.component('group-selection', groupSelection);
Vue.component('group-panel', groupPanel);
Vue.component('empty-group-panel', emptyGroupPanel);
Vue.component('members-panel', membersPanel);
Vue.component('user-show', userShow);
Vue.component('user-edit', userEdit);
Vue.component('user-settings', userSettings);
Vue.component('group-members', groupUsers);
Vue.component('invite-members', inviteUsers);
Vue.component('user-events', userEvents);
Vue.component('user-assignments', userAssignments);
Vue.component('note-index', notes);
Vue.component('group-events', events);
Vue.component('event-show', eventShow);
Vue.component('event-edit', eventEdit);
Vue.component('event-comments', eventComments);
Vue.component('events-table', eventsTable);
Vue.component('group-assignments', assignments);
Vue.component('assignment-show', assignmentShow);
Vue.component('assignment-show-file-upload', assignmentFileUpload);
Vue.component('assignment-edit', assignmentEdit);
Vue.component('assignment-comments', assignmentComments);
Vue.component('assignments-table', assignmentsTable);
Vue.component('jw-pagination', JwPagination);
Vue.component('chatrooms-index', chatroomsIndex);
Vue.component('chatrooms-show', chatroomsShow);
Vue.component('group-edit', groupEdit);
Vue.component('group-whiteboard', groupWhiteboard);
Vue.component('group-statistics', groupStatistics);
```

Obrázok 21: Vue.js mená

Štruktúra Vue.js komponentu:

- <template> obsahuje HTML kód,
- <script> obsahuje JS kód (spracovanie dát a reaktivita aplikácie),
- <style> obsahuje CSS kód.

Axios

Axios je JavaScript balíček, ktorý slúži na správu HTTP požiadaviek. Využila som ho vo funkciách v <script> časti Vue.js komponentov na posielanie väčšiny požiadaviek z frontendu na backend – ako napríklad pri odosielaní vyplneného formuláru.

Používané príkazy [16] :

- GET požiadavka (obrázok 22)

```
axios.get(' url spracuje web.php ')
  .then(function (response) {
    // ak úspešne vykonané
    console.log(response);
  })
  .catch(function (error) {
    // ak nastala chyba
    console.log(error);
  })
  .then(function () {
    // vykoná sa vždy
  });
```

Obrázok 22: Axios GET požiadavka [16]

- POST požiadavka (obrázok 23, vytvorenie udalosti)

```
axios
    .post("/events", this.groupEventsFields)
    .then((response) => {
        this.groupEventsFields = {};
        this.createNewEvent = false;
        this.newEventCreated = true;
        this.savedEvents.unshift(response.data);
    })
    .catch((error) => {
        console.log(error.message);
    });
```

Obrázok 23: Axios POST požiadavka

- PATCH požiadavka (obrázok 24, update údajov úlohy)

```
axios.patch("/assignments/" + this.assignment.id + "/edit",
    this.shownAssignment).then(response => {
    window.location.href = "/assignments/" + this.assignment.id;
}).catch(error => {
    console.log(error.message);
});
```

Obrázok 24: Axios PATCH požiadavka

- DELETE požiadavka (obrázok 25, zmazanie skupiny)

```
axios
    .delete("/groups/" + this.group.id + "/destroy")
    .then((response) => {
        window.location.href = "/dashboard";
    })
    .catch((error) => {
        if (error.response.status == 422) {
            this.errors = error.response.data.errors;
            console.log(this.errors);
        }
        console.log(error.message);
    });
```

Obrázok 25: Axios DELETE požiadavka

Odosielanie e-mailov

Každý používateľ má možnosť si nastaviť na základe akých zmien bude dostávať e-mailové upozornenia. Podľa obsahu sú e-maily rozdelené do niekoľkých samostatných tried, ktoré dedia od triedy Mailable. Tieto triedy obsahujú funkciu *build()*, kde sa nastavuje predmet správy a volí sa blade súbor, ktorý je šablónou textu správy. Odosielanie sa spúšťa v Controlleri pomocou príkazu *Mail::send(new MenoMailableTriedy(...))*, ktorý následne s využitím SMTP protokolu odošle e-mail.

Eventy a broadcasting

Keďže sa v aplikácií nachádzajú prvky ako sekcia pre komentáre, skupinová konverzácia a súkromné správy, bolo potrebné vytvoriť k týmto dejom Eventy (teda Udalosti), aby sa mohli diať v reálnom čase. Tieto Eventy potom boli ostatným používateľom zdieľané pomocou právami podmieneného broadcastingu. To znamená, že udalosť mali právo zachytiť len oprávnení používatelia, ako členovia danej skupiny alebo konverzácie. Eventy majú preto implementovanú vlastnosť „ShouldBroadcast“ a v metóde *broadcastOn()* ju vysielajú pomocou nového súkromného kanálu. Tieto kanály a prístupy ku nim sú spravované v Laravel súbore channels.php, ako je vidieť na obrázku 26.

Obrázok 26: Súbor channels.php

Jednotlivé vysielania v rámci kanálov sú podporované pomocou Pusher Channels (viď 1.2 Využité technológie).

Premenné prostredia (.env)

Uviesť premenné prostredia je nevyhnutnou podmienkou pre beh aplikácie. Nastavujú sa v .env súbore, kde ich Laravel hľadá. Medzi nimi sú napríklad:

- APP_KEY : jedinečný náhodný 32 znakový reťazec, Laravel ho vygeneruje pri vytváraní aplikácie,
- APP_URL : odkaz na webovú aplikáciu,
- DB_CONNECTION : typ spojenia s databázou, napr. mysql, psql, a podobne,
- DB_DATABASE : meno databázy,
- BROADCAST_DRIVER : nastavenie konkrétneho driveru (napr. pusher) umožňuje broadcasting eventov, a tak reakcie v reálnom čase,
- PUSHER_APP_ID : pripojenie na službu Pusher (spolu s PUSHER_APP_KEY, PUSHER_APP_SECRET a PUSHER_CLUSTER),
- MAIL_HOST : poskytovateľ e-mailovej služby,
- MAIL_FROM_ADDRESS : adresa, z ktorej sa budú odosielať e-maily.

4 Testovanie

Aplikáciu testovali dve skupiny ľudí a aj jednotlivci. Mesiac ju využívala 10 členná skupina žiakov, ktorí sa pravidelne stretávajú každý týždeň. Na pár dní si ju vyskúšala aj iná 7 členná skupina študentov. Spätnú väzbu som dostala aj od jedného marketéra a jedného informatika.

Testovanie odhalilo niekoľko chýb:

- preklepy v textoch,
- pri načítavaní súborov chýbalo prednastavenie požadovaných formátov; prehliadač ukazoval všetky,
- chýbala kontrola práv používateľa pre prístup k poznámkam,
- v niektorých prípadoch sa nepodarilo uložiť používateľské nastavenia,
- nefunkčné odosielanie e-mailov (vznikol problém s prístupom do Gmail konta, z ktorého sa e-maily posielajú).

Niektorí testeri uviedli, že nebolo jasné:

- čo robiť po prvej registrácií,
- ako sa dostať k nastaveniam skupiny,
- názov „Whiteboard" bol pre niektorých mätúci – očakávali kreslenie,
- štatistika pracovnej záťaže – chýbal vysvetľujúci popis,
- prečo sa nezobrazila daná vytvorená udalosť v tabuľke – v tabuľke sa zobrazujú len udalosti, ktoré prebiehajú alebo sa ešte len budú konať, je možné spätne zaznačiť udalosť, ktorá sa už stala, ale pre jej zobrazenie treba kliknúť na tlačidlo „Load older", ktoré zobrazí staršie záznamy.

Všetky nájdené chyby som odstránila. Nejasnosti som individuálne posúdila a riešila najmä doplňujúcimi vysvetľujúcimi textami.

Testeri mali pozitívne ohlasy ohľadne:

- prehľadnosti používateľského rozhrania,
- rýchlemu prístupu k najpoužívanejším funkciám,
- možnosti vidieť všetky úlohy a udalosti na jednom mieste zo všetkých skupín,
- veľkého skupinového četu,
- variability vlastných nastavení e-mailových notifikácií pre každú skupinu zvlášť,
- škály voliteľných polí formulárov pri tvorbe nových úloh,
- rozsahu administrátorských práv.

Testovanie bolo určite neodmysliteľnou súčasťou práce. Okrem odhalených chýb mi prinieslo veľa smerodajných rád a pravidiel do budúcna. Veľmi pozitívne ma prekvapila otázka vyššie spomínanej skupiny žiakov, či by mohli používať aplikáciu aj budúci školský rok. Celkovo všetci testeri chválili najmä nápad „súkromnej sociálnej siete", kde má používateľ všetko na jednom mieste, ale zachováva si istú anonymitu voči ostatným používateľom.

Záver

Cieľom práce bolo vytvoriť webovú aplikáciu, ktorá by uľahčila a napomáhala organizovať prácu multifunkčne pre tímy spolužiakov, krúžky študentov, kolegov v práci, ale aj domácnosti a iné skupiny ľudí. Aplikáciu, ktorá zjednotí všetko na jednom mieste pre jedného používateľa bez toho, aby mal výhrady voči narušeniu jeho súkromia (napríklad pracovný kruh používateľa nemá dosah na jeho rodinný kruh ľudí, ako to naopak býva na iných sociálnych sieťach).

Vzniku samotného Návrhu predchádzalo veľa samoštúdia. Nápad, akú aplikáciu by som chcela vytvoriť som mala už dlhší čas, najmä vzhľadom na aktuálnu situáciu vo svete. Chýbala mi technológia, ktorú by som použila na implementáciu. Počas bakalárskeho štúdia som nadobudla cenné znalosti o jazykoch PHP, JavaScript a PostgreSQL, o ktorých som sa chcela naučiť ešte viac. Vybrala som si Laravel framework pre backend, ktorý som sa učila od úplných základov čítaním dokumentácie [14] a sledovaním oficiálnych tutoriálov. Bola to veľmi hodnotná skúsenosť, keďže tento framework má jedinečnú architektúru (ako napríklad MVC model) a množstvo konvencií, ktoré som sa učila dodržiavať. Prvej verzii aplikácie chýbala reaktivita, preto som hľadala JavaScript framework pre frontend. Nakoniec som sa učila od základov aj Vue.js čisto čítaním dokumentácie [6]. Dovtedy som sa nestretla ani so žiadnym CSS frameworkom ako je TailwindCSS, ktorý som pri tvorbe priebežne študovala z dokumentácie [12].

Po samoštúdiu som sa stretla s viacerými problémami aj pri tvorbe Návrhu. Niekoľko databázových modelov sa po dlhšej úvahe ukázalo nepoužiteľnými alebo zbytočne komplikovanými. Napríklad bolo riešením ukladať dáta priamo do pivotnej tabuľky, čím sa predišlo viacerým nadbytočným databázovým dopytom.

Počas implementácie som dlho pracovala najmä na komplexnejších databázových dopytoch, prepájaní frontendu a backendu a nastavení izolovanosti databázových transakcií. Najťažší problém bol pre mňa implementovať funkcionality v reálnom čase. Opäť som študovala novú technológiu Pusher Channels; ako ju implementovať do aplikácie a ako vytvárať, kontrolovať, počúvať a autorizovať Eventy (udalosti) na základe akcií používateľa.

Na aplikácií by sa určite dalo pracovať aj ďalej. Rozšíriť ju napríklad o voliteľne veľké skupinové čety, dať možnosť používateľovi vybrať obdobie, za ktoré počíta štatistiku pracovnej záťaže alebo pridať automatickú detekciu hypertextových odkazov a možnosť voľby jazyka.

Táto bakalárska práca mi priniesla veľa nových vedomostí a skúseností: ako pracovať s frameworkom, vytváranie zložitejších databázových modelov a dopytov, návrh používateľského rozhrania, inštalácia aplikácie na server, ale aj komunikácia s testermi. Videla som celkom jasne, že tieto uplynulé roky štúdia boli veľmi dôležité a ako sa na ich základoch môžem učiť aj naďalej.

Prílohy

- Zdrojový kód aplikácie
- Súbor README.TXT obsahujúci inštrukcie k inštalácií aplikácie
- Odkaz na adresu aplikácie:
  https://radar-assistant-mxf55.ondigitalocean.app/

Literatúra

[1] Zákon o informačných technológiách [online] [cit. 24.01.2021].
Dostupné na: 95/2019 Z.z. - Zákon o informačných technológiách ... - SLOV-LEX (slov-lex.sk)

[2] Wilmington. Webová aplikácia [online] [cit. 25.01.2021].
Dostupné na: https://managementmania.com/sk/webova-aplikacia-web-application

[3] Wilmington. Databáza [online] [cit. 25.01.2021].
Dostupné na: https://managementmania.com/sk/databaza

[4] Wilmington. PHP frameworky [online] [cit. 26.01.2021].
Dostupné na: https://managementmania.com/sk/php-frameworky

[5] PHP Tutorial [online] [cit. 26.01.2021].
Dostupné na: https://www.w3schools.com/php/

[6] Introduction; What is Vue.js? [online] [cit. 27.01.2021].
Dostupné na: https://vuejs.org/

[7] Pusher Channels overview [online] [cit. 27.01.2021].
Dostupné na: https://pusher.com/docs/channels

[8] Teamhood [online] [cit. 01.02.2021].
Dostupné na: https://teamhood.com/

[9] MeisterTask [online] [cit. 01.02.2021].
Dostupné na: https://www.meistertask.com/

[10] Clockify [online] [cit. 02.02.2021].
Dostupné na: https://clockify.me/

[11] Jason Mario. 7 Best Team Management Apps for 2020 [online]
[cit. 03.02.2021].
Dostupné na: https://blog.weekdone.com/best-team-management-apps/

[12] Tailwindcss [online] [cit. 03.02.2021].
Dostupné na: https://tailwindcss.com/

[13] Laravel History [online] [cit. 26.02.2021].

Dostupné na: https://www.w3schools.in/laravel-tutorial/history/

[14] Laravel 8 Documentation [online] [cit. 26.02.2021].

Dostupné na: https://laravel.com/docs/8.x

[15] Fisayo Afolayan. Why should you use Vue.js when using Laravel [online] [cit. 26.02.2021].
Dostupné na: https://blog.pusher.com/why-vuejs-laravel/

[16] axios [online] [cit 24.04.2021]
Dostupné na: https://www.npmjs.com/package/axios