VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

OSOBNÍ KALENDÁŘ PHP

PERSONAL CALENDAR PHP

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **MATEJ SADLOŇ**
AUTHOR

VEDOUCÍ PRÁCE — **MARTIN KOLÁŘ, M.Sc.**
SUPERVISOR

BRNO 2017

Abstrakt

Táto práca sa venuje problematike manažovania času. Jej cieľom je vytvorenie webovej aplikácie fungujúcej ako pomôcka pre jednoduchšiu a prehľadnejšiu prácu s osobným časom. Pre tento účel využíva postupy manažovania času ako napríklad kalendáre, To-Do listy, delenie zložitejších problémov na časti formou projektov či spätné sledovanie využitia času. Aplikácia je vytvorená za pomoci PHP frameworku Symfony a objektovo-relačného mapovania, ktoré zabezpečujú PHP knižnica Doctrine s využitím MySQL databázy. Pre lepší vzhľad, dynamickosť a prácu s aplikáciou sú použité doplňujúce technológie ako CSS alebo JavaScript.

Abstract

This thesis aims to time management. The goal is to design and implement web application for simpler and easy-to-use managing of personal time. For this, the application uses calendars, To-Do lists, dividing of bigger task into subtasks or analyzing of spent time. The PHP framework named Symfony was used for implementation of the application with use of object-relation mapping, that is provided by the PHP library Doctrine and uses MySQL database. The CSS and JavaScript technologies were used for better graphical design and dynamism of work with application.

Kľúčové slová

manažovanie času, kalendár, To-Do list, webová aplikácia, objektovo orientované programovanie, PHP, Symfony, Doctrine, Composer

Keywords

time management, calendar, To-Do list, web application, object oriented programming, PHP, Symfony, Doctrine, Composer

Citácia

SADLOŇ, Matej. *Osobní Kalendář PHP*. Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Martin Kolář, M.Sc.

Osobní Kalendář PHP

Prehlásenie

Prehľasujem, že som túto bakalárskou prácu vypracoval samostatne pod vedením pána Koláře Martina, M.Sc. . Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Matej Sadloň
11. mája 2017

Obsah

Kapitola 1

Úvod

V dnešnej dobe je životný štýl mnohých ľudí veľmi uponáhľaný. Každý z nás zažíva dni kedy musí vykonať mnoho činností a často sa k niektorým z nich nakoniec ani nedostane pretože nestíha. Následne sa cítime pod tlakom a vystresovaný čo má vplyv na vaše zdravie. Práve z tohto dôvodu sa v poslednej dobe začala venovať väčšia pozornosť problematike manažovaniu času. To nám ponúka nástroje a metódy ako efektívne využívať náš osobný čas, a tak vylepšovať naše pracovné výkony alebo rezervovať časť nášho času na relax bez toho aby sme sa museli obávať, že zase niečo nestihneme.

Účelom tejto práce je vytvorenie webovej aplikácie využívajúcej práve niektoré z týchto nástrojov tak, aby umožňovala komplexné manažovanie osobného času pre jej užívateľov. Mala by umožňovať prihlásenie sa pomocou užívateľského účtu, ktorý bude zároveň obsahovať a uchovávať všetky údaje zadané užívateľom. Pre správu časovo obmedzených udalostí bude využívať tzv. kalendár udalostí, ktorý ich bude zobrazovať a zároveň bude umožňovať ich správu. Tieto udalosti bude možné rozdeľovať do kategórií, a tým ich odlišovať. Taktiež bude možné ich zaradenie k niektorému z projektov. Tieto projekty tvoria ďalšiu časť funkcionality výslednej aplikácie a ich hlavným účelom bude zoskupovanie jednotlivých menších úloh pod jeden celok, ktorý predstavuje zložitejší problém. Pre menšie, časovo neobmedzené alebo neurčené úlohy bude aplikácia implementovať funkcionalitu tzv. To-Do listov, ktoré predstavujú zoznam týchto úloh zoskupených do určitého celku, ako príklad si môžeme uviesť položky nákupu zoskupené do jedného zoznamu, ktorý predstavuje samotný nákup.

Aplikácia by taktiež mala poskytovať doplňujúce informácie, ktoré by užívateľ využil pri bežnom plnení svojich naplánovaných udalostí. Väčšina ľudí pri plnení úloh zohľadňuje napríklad údaje o vonkajšom prostredí. Z tohto dôvodu sa informácia o počasí javí ako veľmi užitočná v prípade, že má užívateľ naplánované udalosti konajúce sa v exteriéry.

Teoretická časť práce slúži ako sprievodca popisujúci proces vytvorenia aplikácie spĺňajúcej vyššie uvedený popis, a to od základných krokov, ktorými je nahliadnutie do problematiky manažovania času a návrhu aplikácie, až po jej implementáciu, testovanie a návrh prípadných rozšírení, ktorým sú venované posledné kapitoly tejto práce.

Kapitola 2

Manažovanie času

Manažovanie času (z ang. time management) má viacero definícii. Najpoužívanejšia definícia vraví, že manažovanie času je proces organizovania a plánovania ako čo najefektívnejšie rozdeliť čas medzi jednotlivé aktivity [28], taktiež bolo definované termínmi ako vyváženosť, flexibilita a kontrola nad časom (Lakein, 1973) alebo tiež ako určitý návyk získaný za pomoci odhodlania a praxe (Soucie, 1986). [4]

Hlavným dôvodom prečo získalo manažovanie času v posledných rokoch toľko pozornosti je najmä uponáhľaný život moderných ľudí. Tí musia každý deň vykonávať veľa najrozličnejších činností, ktoré medzi sebou často kolidujú. Bez ďalších nástrojov sa stáva problematickým rozhodovať sa medzi jednotlivými činnosťami a správne ich organizovať, na úkor čoho sa komplikuje dosahovanie osobných cieľov, sú preferované menej podstatné činnosti, a najmä vzniká plytvanie našim osobným časom. Toto sa odzrkadľuje zvýšeným stresom, ktorý má následne vplyv aj na náš zdravotný stav. Ako už bolo spomenuté manažovanie času nám napomáha v organizovaní a plánovaní týchto činností, čo nám prináša veľa benefitov, ktorými je napríklad väčšia produktivita a efektivita, lepšia reputácia v zamestnaní, menšia miera pociťovaného stresu, prináša viacej možností na osobný rast, a hlavne nám umožňuje jednoduchšie dosiahnuť naše osobné a kariérne ciele.[28]

Na to, aby sme mohli efektívne využívať nástroje spojené s manažovaním času je potrebné, aby sme vedeli odhadnúť koľko času nám zaberie daná činnosť. Ďalej musíme vedieť čo chceme robiť, čo musíme robiť a do akého dátumu je to potrebné danú činnosť vykonať. V minulosti taktiež vznikli rôzne debaty na tému aké schopnosti sú potrebné pre efektívne manažovanie času. Napríklad Shipman vo svojom diele Effective time-management techniques for school administrators (1983) identifikoval šesť princípov pre efektívne manažovanie času. Medzi tieto princípy patrí uvedomovanie si samého seba, správne rozdelenie času, vytýčenie si gólov a priorít, zvyšovanie svojej osobnej efektivity, plánovanie času pre aktivity a taktiež plánovanie času na relax.[4]

Ak by sme sa však bližšie zamerali na literatúru zaoberajúcu sa problematikou manažovania času, tak by sme dostali sedem schopností alebo správaní potrebných pre jeho efektívne využitie. Tieto schopnosti zahŕňajú analýzu času, plánovanie, vytyčovanie cieľov, určovanie priorít, rozvrhovanie, organizovanie a v neposlednom rade zakladanie nových a vylepšovanie existujúcich návykov spojených s manažovaním času.[4] Tieto schopnosti sú následne potrebné pri určovaný jednotlivých cieľov a ich priorít. Až po ich špecifikovaní môžeme využiť niektorú z metód manažovania času. Problematike určovania cieľov a ich priorít, a taktiež nadväzujúcim metódam budú venované nasledujúce podkapitoly.

2.1 Určovanie cieľov

Určovanie cieľov je základnou činnosťou využívanou pri manažovaní času. Je tomu tak najmä preto, lebo ich následné dosahovanie má na nás veľký psychologický dopad, keďže nám dáva pocit úspechu, a zároveň nám poskytuje pohľad na to, čo všetko sme doteraz dokázali, a tým nás poháňa vpred. Pokiaľ, ale nie sú ciele určené správne, nie je možné ich efektívne rozdelenie a následné dosahovanie ačo má často za následok nechuť a nezáujem pokračovať v práci ďalej. Práve z tohto dôvodu by mali všetky naše ciele spĺňať tzv. SMART kritéria, v ktorých jednotlivé písmená zastávajú nasledujúci význam:

- *Specific* - určenie špecifickej oblasti, ktorú chceme zlepšiť
- *Measurable* - určenie alebo aspoň odhadnutie ukazovateľa pokroku
- *Achievable* - dosiahnuteľnosť daného cieľa
- *Realistic* - určenie dosiahnuteľných výsledkov za pomoci dostupných zdrojov
- *Time-related* - špecifikovanie, kedy môžu byť výsledky dosiahnuteľné

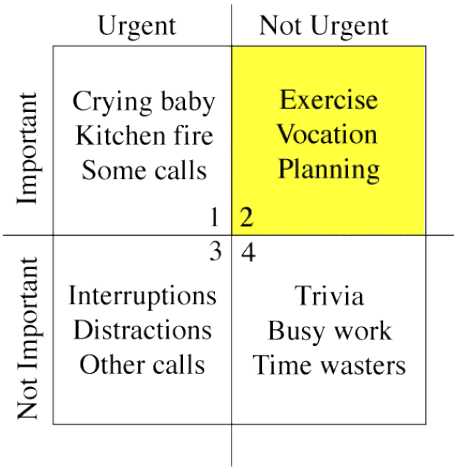
Pri dosahovaní zložitejších cieľov je užitočné ich delenie na čiastkové úlohy. Pre tento účel je výhodné využiť tzv. spätné určovanie cieľov, pri ktorom začíname od výsledku, ktorý chcem dosiahnuť, následne sa vraciame späť a popritom určujeme poradie krokov, ktoré je potrebné vykonať preto, aby sme dosiahli požadovaný výsledok. Oproti iným prístupom je tento spôsob menej intuitívny, avšak umožňuje lepšie určenie toho, či je cieľ dosiahnuteľný v nami vymedzenom časovom rozsahu. Taktiež umožňuje lepšie určenie závislostí a kľúčových bodov, ktoré je potrebné vyriešiť predtým než prejdeme na ďalší krok problému.[2]

Po určený jednotlivých cieľov, môžeme začať s určovaním ich priorít a následným organizovaním pomocou niektorej z metód manažovania času.

2.2 Určovanie priorít

Určovanie priorít nie je nevyhnutné pre prácu s ďalšími metódami manažovania času, avšak má veľký vliv na konečnú efektivitu práce. Proces určovania si priorít je často využívaný napríklad pri vytváraní To-Do listov či delení cieľov na čiastkové úlohy. Najdôležitejšími faktormi, ktoré je treba brať pri určovaní priorít do úvahy, sú urgentnosť a dôležitosť danej úlohy. Z pozorovaní bolo zistené, že najdôležitejšie úlohy často nie sú tými najurgentnejšími, napriek tomu ľudia zvyknú stavať urgentné úlohy do dominantnejšej pozície v ich životoch.

Covey, Marrill a Marrill v knihe First Things First (1994) kategorizujú naše aktivity do štyroch kvadrantov Eisenhowerovej rozhodovacej matice (z ang. Eisenhower Decision Matrix) a to ako urgentné, neurgentné, dôležité a nedôležité. Úlohy, ktoré sú urgentné aj dôležité musia byť vykonané čo najskôr. Úlohám, ktoré nie sú dôležité by sme podľa Coveya malo venovať menej času, bez ohľadu na to či sú urgentné alebo nie, a tento čas by sa mal venovať úlohám, ktoré sú naozaj dôležité. Sústredením sa na dôležité úlohy totižto získame väčšiu kontrolu nad naším časom, a taktiež predídeme tomu, aby sa stali zároveň aj urgentnými. Tento spôsob určovania priorít možno použiť napríklad pri vytváraní To-Do listov, a to tak, že jednotlivým úlohám priradíme prioritu, či už ich rozdelením do určitých skupín alebo za pomoci ich farebného označenia. Takto upravený To-Do list nám umožňuje odložiť úlohy, ktoré síce môžu byť zaujímavé a sľubujú dosiahnutie určitých cieľov, avšak nezapadajú medzi naše základné priority.[5] [3]



Obr. 2.1: Eisenhowerova rozhodovacia matica [16]

Ďalším spôsobom ako určiť, ktoré činnosti sú tie najpodstatnejšie je tzv. Paretova analýza. Tento princíp je založený na tvrdení, že až 80% činností je možné vykonať za 20% času. Zvyšných 20% činností však zaberie 80% času, a práve tieto by mali získať najvyššiu prioritu. Pomáha teda zoskupovať činnosti do určitých prioritných kategórii. Nevýhodou Paretovej analýzy je však to, že ju nie je možné aplikovať na menšie, ale za to podstatné činnosti. Tieto by boli v prípade jej použitia označené nízkou prioritou. Paretova analýza je často používaná s ďalšími analytickými nástrojmi.[16]

	Customer Service (A)	Employee Training (B)	Increasing Sales (C)	Decreasing Lost Revenue (D)
Customer Service (A)		A, 2	C, 1	D,1
Employee Training (B)			C, 2	D, 3
Increasing Sales (C)				C, 2
Decreasing Lost Revenue (D)				

Obr. 2.2: Analýza párovým porovnávaním [2]

Pre určenie priorít daných činností je taktiež možné použiť tzv. analýzu párovým porovnávaním. Táto metóda je založená na princípe porovnávania každej činnosti na našom liste so všetkými zvyšnými. Účinná je hlavne v prípade, kedy nemáme určené rozhodovacie kritériá alebo keď porovnávame navzájom veľmi odlišné činnosti. V prípade, že chceme vy-

užiť analýzu párovým porovnávaním, musíme najskôr označiť jednotlivé úlohy písmenom a následne vytvoriť tabuľku, v ktorej búdu tieto činnosti označovať riadky a stĺpce tabuľky rovnako ako na obrázku 2.2. Následne porovnávame činností medzi sebou a označenie viac prioritnej činnosti zapisujeme do tabuľky spolu s číslom predstavujúcim veľkosť rozdielu priorít medzi týmito činnosťami. Toto číslo označujeme ako hodnotu dôležitosti. Každé dve činnosti porovnávame iba raz, a taktiež neporovnávame danú činnosť so samou sebou. Z tohto dôvodu vypĺňame iba políčka nad (prípadne pod) diagonálou a nie celú tabuľku. Nakoniec spočítame všetky hodnoty dôležitosti pre každú z činností a výsledky porovnáme tak, aby činnosť s najvyšším výsledným číslom získala najvyššiu prioritu.[2]

2.3 To-Do listy

Metóda vytvárania To-Do listov je založená na princípe vytvárania listov obsahujúcich zoznam činností, ktoré je potrebné vykonať. To-Do listy môžu pozostávať zo stránky papiera alebo sa môže jednať napríklad o textový súbor, záleží na osobnej preferencii užívateľa. Dokážu byť silným nástrojom, ktorý nám pripomína činnosti, ktoré musíme vykonať, a tým urýchľujú proces ich kompletizácie. Ak obsahujú činnosti s určenou prioritou, tak dokážu efektívne určiť, ktoré z nich by bolo potrebné vykonať ako prvé. Sú vhodné najmä pre kratšie časové obdobia, ktoré neobsahujú priveľa činností. Splnenie danej činnosti je následne potrebné na liste označiť ako ukončené. Takáto vizualizácia má taktiež vplyv na motiváciu a sebavedomie užívateľa.[24]

2.4 Kalendáre

Kalendáre môžu mať rovnako ako To-Do listy papierovú alebo elektronickú formu. Často sú zabudované ako súčasť rôznych systémov. Sú využívané najmä v oblasti rozvrhovania a organizácie časového plánu.

Rozvrhovanie je forma plánovania činností, ktoré mám umožnia dosiahnuť naše ciele v čase, ktorý máme reálne k dispozícii. Základom rozvrhovania je definovanie dostupného času, následne je potrebné zaznamenať jednotlivé činnosti do kalendára. Ako prvé je potrebné zaznamenať tie, ktoré sú pevne dané a musia byť vykonané za každých podmienok. Po nich nasledujú činnosti s vysokou prioritou, pre tento účel je výhodné využiť To-Do list, v ktorom máme činnosti už vopred rozdelené podľa priorít. Ďalej je užitočné vymedziť si určitý časový úsek pre nepredvídateľné či mimoriadne udalosti. Odhadnutie tohto časového úseku je závislé najmä na osobnej skúsenosti užívateľa. Využitie zostávajúceho času je na našom uvážení. Je vhodné opäť nazrieť do To-Do listu obsahujúceho priority, zvážiť osobné ciele, zhodnotiť čas na ich dosiahnutie a prípadne ich taktiež zaradiť do kalendára.

Ďalším druhom je kalendár obsahujúci dosiahnuté výsledky. Táto forma kalendárov nemá priamo za účel rozvrhovanie a organizáciu nášho časového plánu, ich efekt je hlavne psychologický, pretože nám dávajú pocit uspokojenia z vykonanej práce. Vytvárať sa dajú napríklad tak, že si každý večer spíšete činnosti, ktoré ste za daný deň vykonali a spätne si overíte či ste splnili plán, prípadne či by ste nemali zvoliť lepšiu stratégiu plánovania.

Projektové kalendáre obsahujú činnosti spojené s určitým projektom. Sú nápomocné hlavne v prípade, kedy je jedna činnosť dokončená a nevieme čím je potrebné pokračovať. Činnosti v tomto kalendári sú zväčša ohraničené konečnými termínmi. Tento druh kalendára je jednoduché zdieľať, pretože činnosti, ktoré obsahuje sú často pevne dané už v začiatkoch daného projektu. Vďaka svojej nemennosti je vhodný aj pre prácu v tíme.[24]

Kapitola 3

Existujúce riešenia

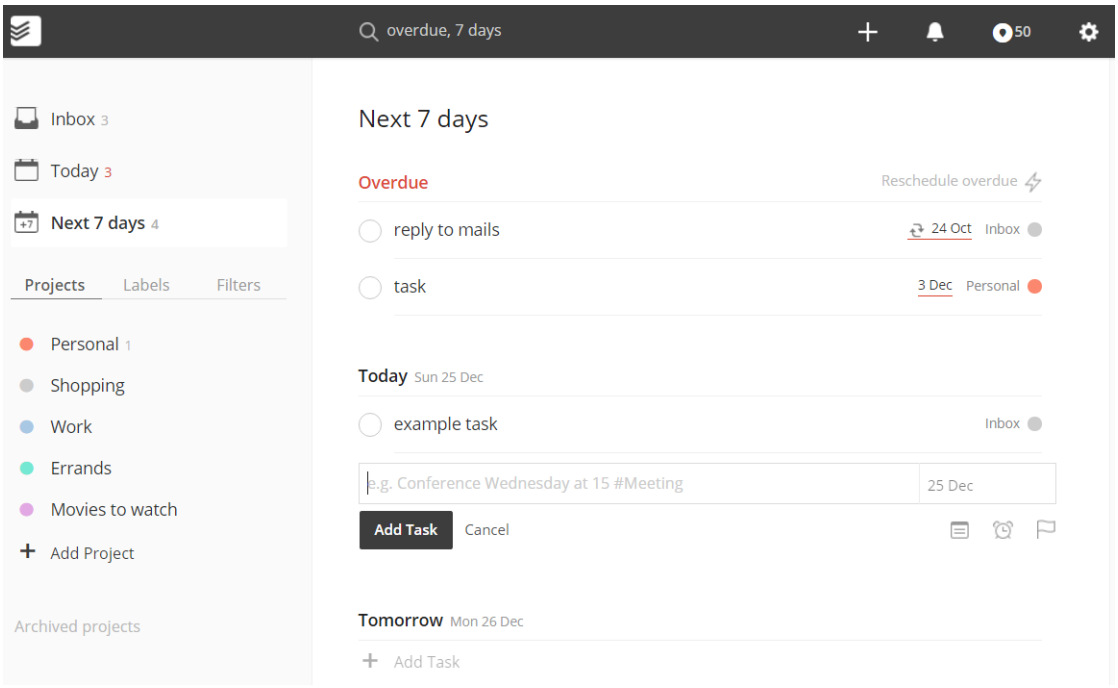
Z manažovania času sa v posledných rokoch stala horúca téma a čoraz viac sa poukazuje na jeho výhody. Tie si začali všímať aj koncový užívatelia, a preto vzniklo a stále vzniká veľa aplikácií, ktoré im v tomto napomáhajú. V tejto kapitole vám priblížime jedny z tých najpoužívanejších aplikácií. Pozrieme sa na ich základnú konštrukciu a na to čo ponúkajú užívateľovi. Taktiež spomenieme, ktoré z metód manažovania času využívajú a na čo sú vhodné. Informácie o jednotlivých aplikáciách boli získané poväčšine priamo z ich webových stránok alebo prácou s danou aplikáciou.

3.1 Todoist

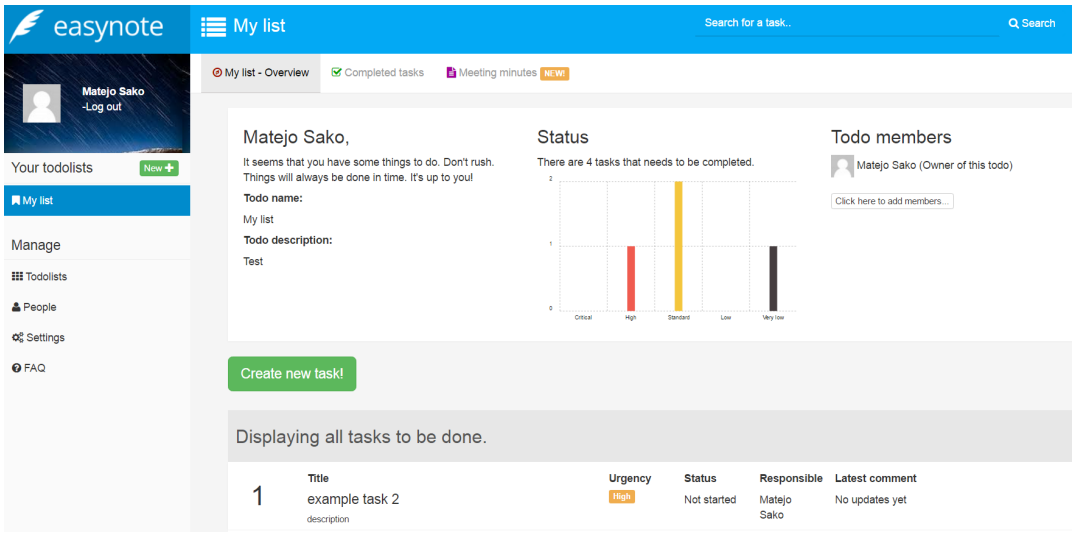
Todoist podporuje vytváranie klasických udalostí, vytváranie kategórií ako projektov, ktoré môžu obsahovať jednotlivé udalosti. Pri vytváraní udalosti je potrebné zadať jej názov, dátum konania a určiť ku ktorému z kalendárov patrí. Systém neumožňuje zadávať presný čas udalosti. Todoist podporuje synchronizáciu s inými zariadeniami, ktoré využívajú Android, Windows Mobile či IOS. Neobsahuje kalendár v klasickej forme, ale namiesto neho má zoznam jednotlivých dní. Tieto následne obsahujú jednotlivé udalosti. Todoist má taktiež platenú verziu za cenu 28.99€ ročne, ktorá navyše obsahuje upozornenia, sledovanie produktivity, pridávanie detailnejších komentárov a príloh k udalostiam či vytváranie užívateľských filtrov na filtrovanie udalostí a mnohé ďalšie.[25]

3.2 Easynote

Táto aplikácia preferuje jednoduchosť. Umožňuje vytváranie To-Do listov, do ktorých je následne možné pridávať jednotlivé udalosti. Týmto udalostiam je možné nadstavovať úroveň urgentnosti, a taktiež dátum a čas ukončenia podľa potreby. Taktiež umožňuje zdieľanie týchto listov s ostatnými užívateľmi pomocou pozvania k danému To-Do listu. Jednotlivé udalosti nemajú presne ohraničený časový rozsah čo neumožňuje sledovanie využitia času, na druhej strane však ponúka štatistiku ukazujúcu rozdelenie udalostí podľa urgentnosti, a to hneď v piatich úrovniach. Tento systém je vhodný najmä pre užívateľov, ktorým stačí práca s To-Do listami a nevyžadujú žiadne ďalšie požiadavky.[21]



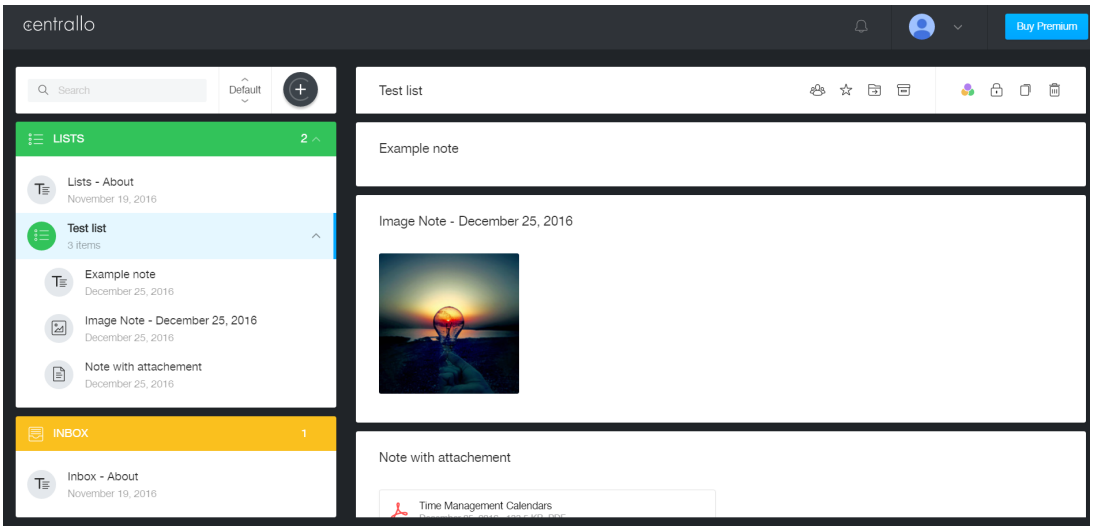
Obr. 3.1: Ukážka aplikácie Todoist



Obr. 3.2: Ukážka aplikácie Easynote

3.3 Centrallo

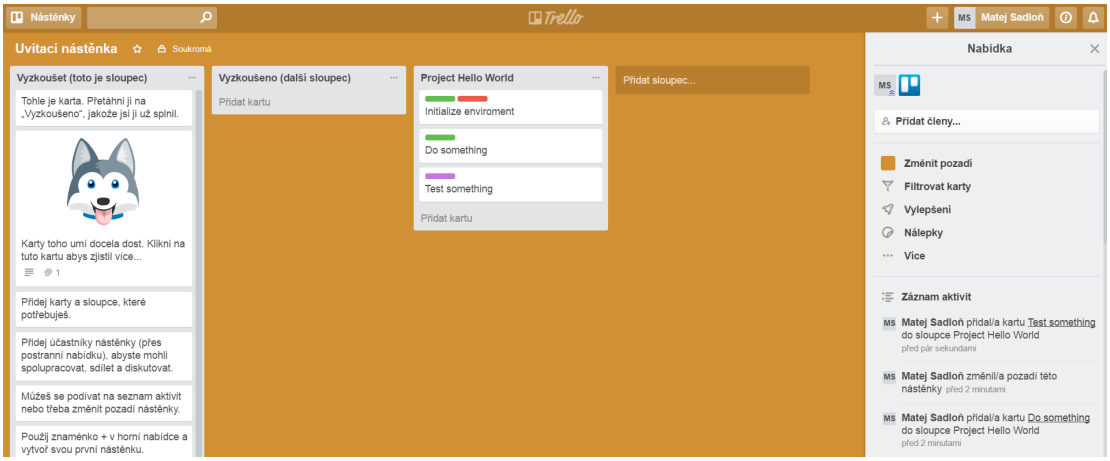
Okrem klasického vytvárania To-Do listov umožňuje Centrallo taktiež určovanie priorít, aj keď iba na jednej úrovni. Tieto listy je následne možné zdieľať pomocou HTTP odkazu. Položky v liste sú prezentované ako poznámky, ktoré je možné editovať a pripájať k nim prílohy. Taktiež obsahuje schránku - inbox, do ktorej je možné ukladať poznámky, fotky, videá či zvukové nahrávky. K dispozícii je taktiež e-mailová schránka. Neobsahuje kalendár, ale iba listy s jednotlivými udalosťami, na úkor čoho nepodporuje sledovanie využitia času či iné štatistiky. Okrem webového klienta existuje aj aplikácia na Android či iOS. Celý systém je synchronizovaný bez ohľadu na aktuálne používaného klienta. Aj z tohto dôvodu je Centrallo ideálne pre užívateľov, ktorí chcú mať prístup k svojim poznámkam, fotkám prípadne e-mailom stále, nezáležiac na tom, či sú práve na počítači alebo svojom inteligentnom telefóne.[19]



Obr. 3.3: Ukážka aplikácie Centrallo

3.4 Trello

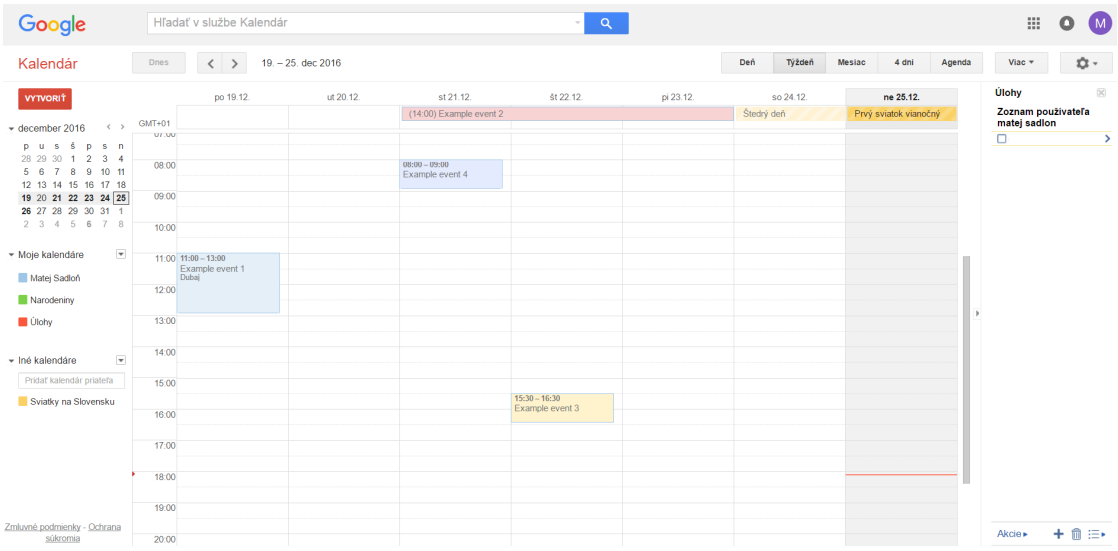
Tento systém funguje na princípe nástenky s úlohami, ktoré je potrebné vykonať. Celá nástenka je niečo ako To-Do list, v ktorom sú jednotlivé úlohy rozdelené do stĺpcov, ktoré môžu predstavovať kategóriu úlohy alebo prípadne fázu, v ktorej sa úloha aktuálne nachádza. K jednotlivým úlohám je možné písať komentáre pridávať menšie prílohy či priradzovať značky, ktoré môžu predstavovať napríklad prioritu danej úlohy. Tieto značky si môže užívateľ ľubovoľne upravovať. Ku celej nástenke a jej úlohám je možné priradzovať ďalších užívateľov, a taktiež z nich umožňuje vytvoriť tým. Túto možnosť je veľmi výhodné využiť v oblasti skupinových projektov a ich delenia na menšie čiastkové úlohy. Všetky zmeny na nástenke sú priradeným užívateľom oznámené formou e-mailových notifikácií a notifikácií priamo v systéme. V Business verzii, ktorá stojí 9.99$ mesačne užívateľ získa navyše prepojenie so systémami ako Github, Google Disk či Dropbox. Ďalej umožňuje pridávanie príloh až do veľkosti 250MB a lepšie manažovanie tímu a jeho nástenieк ako celku.[26]



Obr. 3.4: Ukážka aplikácie Trello

3.5 Google Calendar

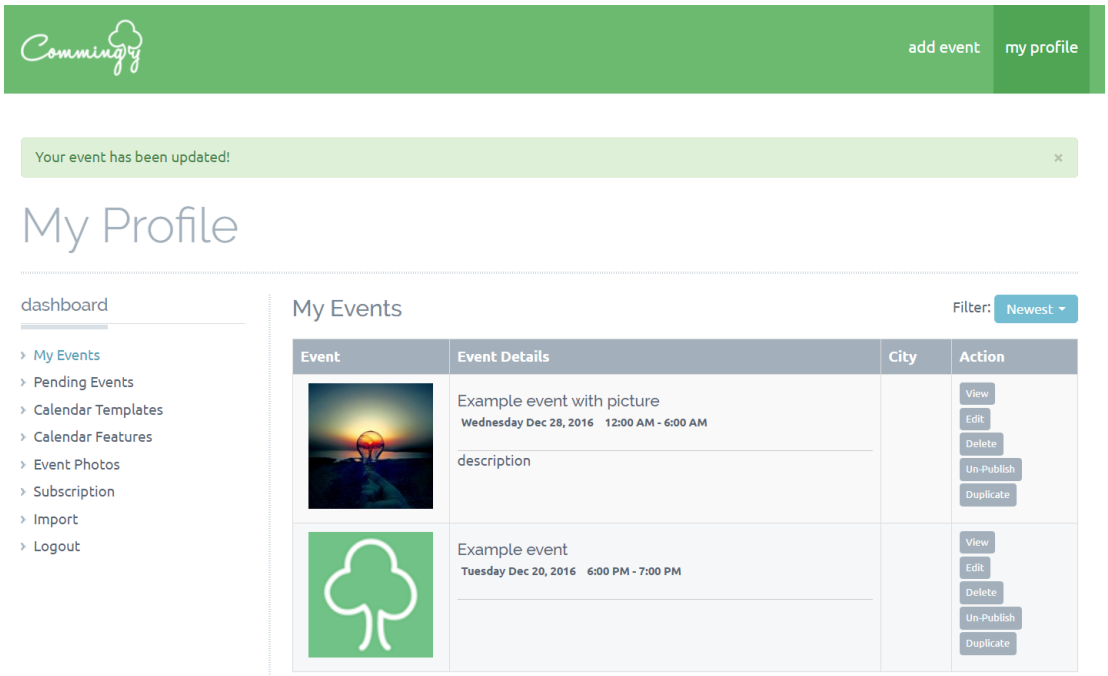
Google Calendar je služba využívaná na manažovanie a rozvrhovanie času pomocou kalendára a bola vytvorená firmou Google.[18] Užívateľom umožňuje vytváranie rôznych kalendárov, do ktorých môžu následne pridávať jednotlivé udalosti, pričom je možné nadstaviť ich trvanie, opakovanie, popis a taktiež to, či a kedy má byť odoslaná e-mailová notifikácia. Tieto kalendáre alebo udalosti je možné zverejniť, a tak ich zdieľať s ostatnými. Tento systém však neumožňuje nadstavovanie priorít pre jednotlivé udalosti, napríklad v prípade ich prekrytia, a taktiež neobsahuje žiadne štatistiky. Prístup ku Google kalendáru získava automaticky každý po vytvorení Google účtu. Je jednoduchý na používanie a je ľahko prístupný z počítača či inteligentného telefónu. Vhodný je najmä pre užívateľov, ktorí potrebujú manažovať a sprehľadniť svoj prehustený program.[12]



Obr. 3.5: Ukážka aplikácie Google Calendar

3.6 Commingly

Podobne ako Google Calendar, tak aj Commingly umožňuje vytváranie kalendárov, do ktorých je následne možné pridávať udalosti. Pri vytváraní je možné danej udalosti nadstaviť titulný obrázok, a taktiež jej prípadné opakovanie. Povinné atribúty udalosti sú jej názov, dátum, čas a trocha nečakane aj adresa konania udalosti. Tento atribút sa môže javiť ako zbytočný, keďže adresa nie je vždy potrebná. Commingly taktiež umožňuje zdieľanie nielen udalostí, ale aj celých kalendárov s ostatnými užívateľmi. Taktiež poskytuje platenú verziu za cenu 9$ mesačne, ktorá obohacuje funkcionalitu o možnosť nadstavenia vzoru či farebnej schémy kalendáru. Za najpodstatnejšiu zmenu oproti neplatenej verzii však považujem možnosť udeľovania práv na editáciu vášho kalendára iným užívateľom. Tento systém taktiež podporuje export udalostí, no neobsahuje žiadne štatistiky či možnosť určovania priorít. Tento systém je vhodný najmä pre užívateľov, ktorí potrebujú zdieľať svoj program s ostatnými, prípadne v platenej verzii pre skupiny ľudí či tímy využívajúce jeden spoločný kalendár.[20]



Obr. 3.6: Ukážka aplikácie Commingly

Kapitola 4

Použité technológie

4.1 PHP

PHP je skratka pre Hypertext Preprocessor. Jedná sa o skriptovací jazyk využívaný najmä pri webovom vývoji, ale v princípe môže byť využitý aj na tvorbu desktopových aplikácií. Jeho prvá verzia PHP/FI bola vytvorená Rasmusom Lerdorfom v roku 1994 a momentálne sa o jeho rozvoj stará tzv. PHP Development Team.[22] PHP kód je zvyčajne spracovávaný PHP interpreterom, ktorý je implementovaný formou modulu na webovom servery. Webový server teda interpretuje a spúšťa PHP kód, ktorým môžu byť rôzne druhy dát a následne z neho generuje webovú stánku. Tento kód môže byť taktiež spúšťaný za pomoci príkazového riadku alebo samostatnej aplikácie s grafickým rozhraním. Posledný spomenutý spôsob je však využívaný iba ojedinele.[27]

4.2 Objektovo orientované programovanie a PHP

Objektovo orientované programovanie, ďalej iba OOP, je v dnešnej dobe často využívaným výrazom v programátorskom svete. Ide o programovacie paradigmy založené na koncepte objektov. Tieto objekty predstavujú abstrakciu reality a vo väčšine objektovo orientovaných jazykoch sú vytvárané na základe predpisu – triedy, ku ktorej patria a rovnako je tomu aj v jazyku PHP. Prvá podpora pre OOP sa objavila už vo verzii PHP 3, kde bolo umožnené vytváranie tried, objektov a ich metód. Plná podpora pre OOP bola následne implementovaná až vo verzii PHP 5, kedy bolo pridané našepkávanie argumentov, kľúčové slová `private` a `protected` pre premenné, menné priestory, výnimky a mnohé ďalšie.[29]

4.3 PHP framework

PHP framework je objektovo orientovaný systém implementovaný v skriptovacom jazyku PHP. Zahŕňa štandardné riešenia problémov bežne vyskytujúcich sa pri tvorbe aplikácie, čím urýchľuje tento proces. Tieto riešenia sa riadia určitými vzormi, ktoré definuje samotný framework, a tým určuje aj výslednú architektúru aplikácie.[1] Pre každý z riešených problémov existuje zväčša samostatný komponent, ktorý poskytuje jeho komplexné riešenie. Framework nám následne spája tieto komponenty spolu s ďalšími knižnicami či pomocnými programami, ktoré môžu mať formu balíkov a zabezpečuje nám komunikáciu medzi nimi, čím vytvára jeden celok, ktorým je výsledná aplikácia. Rôzne frameworky nám poskytujú rozdielne riešenia rovnakých problémov, ktorými môže byť napríklad vytváranie

a spracovávanie formulárov alebo komunikácia s API rozhraním či databázou. Pre správu jednotlivých balíkov využívajú frameworky zvyčajne externé nástroje, ako sú NPM, bower alebo composer, ktorý je detailnejšie popísaný v kapitole 4.5. Momentálne existuje široká škála PHP frameworkov, pričom vznikajú stále nové. Niekoľko z tých najzaujímavejších si spomenieme v nasledujúcich podkapitolách.

4.3.1 Nette

Nette framework je objektovo orientovaný framework vytvorený českým autorom Davidom Grudlom. Momentálne je vyvíjaný spoločnosťou Nette Foundation, a keďže sa jedná o český projekt tak je dokumentácia k nemu písaná najmä v češtine. Je zameraný na tvorbu webových aplikácií v PHP 5 a PHP 7. Kladie veľký dôraz na elimináciu bezpečnostných rizík, podporuje HTML 5, AJAX a SE optimalizáciu. Využíva MVC architektonický vzor a z veľkej časti je zložený z komponentov ktoré predstavujú samostatne fungujúce celky. Ako šablónovací systém využíva Latte, ktorý je vyvíjaný spolu s ním ako jeho súčasť.[23] V rámci Českej republiky je organizovaných veľa školení, ktoré umožňujú hlbšie nazretie do problematiky práce s týmto frameworkom a z časti slúžia ako náhrada neexistujúcich častí dokumentácie. Tento framework je dostupný vo verzii 2.4 pod licenciou GNU GPL a licenciou Nette.[14]

4.3.2 Zendframework

Zend framework je jeden z najrozšírenejších PHP frameworkov, a to z dobrého dôvodu, jeho zakladatelia sú jednými z kľúčových prispievateľov jazyka PHP, takže majú potrebnú znalosť v danej problematike. Zend framework bol zverejnený v roku 2006 a momentálne je vyvíjaný najmä spoločnosťou Zend Technologies Ltd, a RogueWave Company, no k jeho vývoju prispieva veľa ďalších firiem, medzi ktoré sa radia aj veľké mená ako IBM, Google či Microsoft.[17] Tento framework je objektovo orientovaný a využíva menné priestory. Taktiež pokrýva, a tým zjednodušuje takmer všetky funkcionality jazyka PHP formou samostatných balíkov. Pri tých novších je však často potrebné testovanie zo strany užívateľov ktoré zaberie určitý čas, a preto nie je chovanie niektorých z funkcionalít zo začiatku zaručené pre všetky prípady užitia. Vďaka veľkej užívateľskej základni je tento framework veľmi stabilný, čo využívajú pri svojich projektoch hlavne väčšie firmy. Samotnou firmou Zend sú taktiež ponúkané rôzne certifikácie a tréningy k tomuto frameworku priamo na ich webových stránkach. Aktuálne je dostupný vo verzii 3.0.0 pod licenciou New BSD.[17]

4.3.3 CodeIgniter

CodeIgniter je open-soursový PHP framework využívaný pre rýchly vývoj dynamických webových aplikácií. Je založený na MVC architektonickom vzore, avšak pre prácu s ním je povinné využívať iba riadiace jednotky, zatiaľ čo dátové modely a pohľady zostávajú nepovinnými. Pred ostatnými frameworkmi vyniká najme vďaka svojej rýchlosti a malým nárokom. Jeho prvá verzia bola zverejnená 28. 2. 2006 spoločnosťou ElliseLab. Aktuálne je dostupný vo verzii 3.1.4 pod MIT licenciou.[9]

4.3.4 Symfony

Symfony je jedným z najpoužívanejších objektovo orientovaných PHP frameworkov zameraných na znovu použiteľnosť. Jeho hlavnou výhodou je silná užívateľská komunita a podrobne

spracovaná dokumentácia, ktorá ponúka všetko potrebné pre prácu s týmto frameworkom. Bol vydaný ako voľne dostupný 18. októbra 2005 pod MIT licenciou a momentálne je vyvíjaný spoločnosťou SensioLabs.[15] Od tejto doby je stále rozširovaný a sú k nemu pridávané nové komponenty a nástroje prinášajúce inovácie a nové postupy. Určený je hlavne pre stredné a väčšie projekty vyžadujúce práve túto širokú škálu funkcionalít. Ďalšou výhodou, ktorú využijú najmä väčšie projekty je jeho výkonnosť a možnosť ladenia na základe špecifických požiadaviek, a taktiež jeho stabilita, za ktorú vďačí práve spoločnosti SensioLabs, ktorá existuje už 18 rokov a má výborné referencie.[18] Symfony taktiež podporuje objektovo-relačné mapovanie, ktoré zabezpečuje PHP knižnica Doctrine, tej je venovaná podkapitola 4.4. Ani v tejto oblasti však nijako neobmedzuje a podporuje taktiež klasické spôsoby práce s databázou, ako šablónovací nástroj využívaTwig.

Pre potreby našej práce bol zvolený pravé frameworkSymfony vo verzii 3.1.6, a to najmä pre jeho dobre spracovanú dokumentáciu a natívnu podporu knižnice Doctrine.

4.4 Doctrine Project

Doctrine Project je zoskupenie niekoľkých PHP knižníc zameraných na prácu s databázou a mapovanie objektov. Kľúčovými súčasťami tohto projektu sú objektovo-relačný mapper ORM a abstraktná databázová vrstva DBAL, ktorá je postavená nad ním.[11] Entity v Doctrine predstavujú odľahčené PHP objekty uchovávajúce dáta. Triedy týchto entít predstavujú tabuľky databázy, pričom jednotlivé premenné danej entity predstavujú stĺpce tabuľky. Entity samotné následne predstavujú riadky tabuľky, ktoré je možné vkladať alebo vyberať z databázy. Toto je umožnené pravé vďaka mapovaniu, ktoré zabezpečuje ORM. DBAL nám zase slúži na zjednotenie použitých technológii, takže pri práci s Doctrine nemusíme riešiť, či sú dáta uložené v MySQL, PostgreSQL či SQLite databáze. Na starosti má taktiež spracovávanie databázových udalostí či obsluhu databázových transakcií. Jej súčasťou je taktiež schema manager a querybuilder.[8] V práci je využitá verzia Doctrine 2.

4.5 Composer

Composer je nástroj na manažovanie závislostí v programovacom jazyku PHP. Bol vytvorený a zverejnený dvojicou Nils Adermann a Jordi Boggiano v marci 2012. Jeho hlavnou úlohou je manažovanie balíkov a knižníc tretých strán na úrovni aplikácie, takže zvyčajne nič neinštaluje globálne, aj keď poskytuje aj túto možnosť. Pod manažovaní je myslená inštalácia, aktualizácia alebo odstránenie týchto balíkov či knižníc. Tie sú štandardne umiestnené v adresári vendor, ale v princípe ich umiestnenie záleží iba na nás.[10] V kontexte tejto práce bol composer využitý na inštaláciu frameworku Symfony 3 a následnú inštaláciu a správu balíkov tretích strán.

Kapitola 5

Návrh

Táto kapitola je venovaná problematike návrhu aplikácie. Je v nej uvedená jej neformálna špecifikácia, špecifikácie rolí v aplikácii a činností, ktoré môžu vykonávať. Taktiež je v nej podrobne preberaný návrh jednotlivých entít v aplikácii vytvorených na základe jej potrieb. Následne sa zaoberá návrhom vzhľadu a rozloženia aplikácie. Záver tejto kapitoly je venovaný architektúre aplikácie.

5.1 Neformálna špecifikácia

Cieľom práce je vytvorenie webovej aplikácie zjednodušujúcej manažovanie osobného času. Pre tento účel by mala implementovať nástroje bežne využívané v problematike manažovania času, a to najmä metódu delenia úloh na čiastkové úlohy. Hlavnou úlohou aplikácie bude zaznamenávať a zobrazovať jednotlivé úlohy. Tieto môžu byť reprezentované ako činnosti s časovým ohraničením, teda udalosti. Pre ich záznam, správu a zobrazovanie je potrebná implementácia kalendára udalostí, ktorý bude tvoriť hlavnú časť aplikácie. Tieto udalosti môžu patriť k určitej kategórie, vďaka čomu vzniká možnosť ich filtrovania a sledovania štatistík na základe týchto kategórií. Tieto kategórie je potrebné spravovať, takže aplikácia musí umožňovať ich tvorbu, úpravu a mazanie.

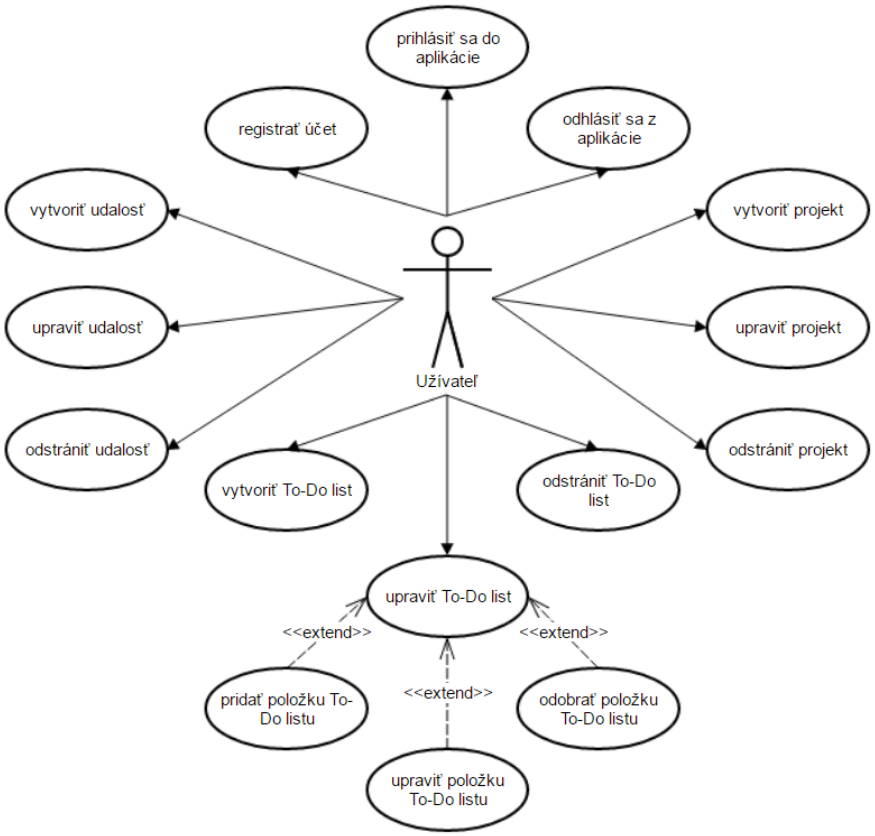
Pre účel delenia úloh na čiastkové úlohy je potrebná implementácia tzv. projektov, kde projekt predstavuje celkový problém, ktorý je potrebné riešiť. Každý projekt obsahuje špecifikáciu problému a jednotlivé čiastkové úlohy nutné pre jeho úspešné dokončenie. Tieto čiastkové úlohy budú v aplikácii reprezentované udalosťami patriacimi k projektu. Rovnako ako pri kategóriách je potrebná správa týchto projektov.

Ďalším druhom úloh, s ktorými sa môže užívateľ stretnúť sú jednoduché časovo neobmedzené úlohy. Pre účel ich záznamu a zobrazovania by mala aplikácia implementovať tzv. To-Do listy, kde každý To-Do list predstavuje zoznam položiek reprezentujúcich úlohy s určitou črtou, ktorá ich spája. Pre tento účel musí aplikácia taktiež implementovať správu týchto To-Do listov a ich položiek.

5.2 Špecifikácia rolí v aplikácii

Nasledujúca podkapitola je venovaná problematike užívateľských rolí v aplikácii. Keďže sa jedná o voľne dostupnú aplikáciu pre bežných užívateľov, tak musí zahŕňať práve rolu užívateľa. Táto rola je zároveň jedinou a umožňuje spravovať a využívať všetky funkcionality

aplikácie. Prehľad všetkých prípadov použitia pre túto rolu zobrazuje nasledujúci diagram na obrázku 5.1.



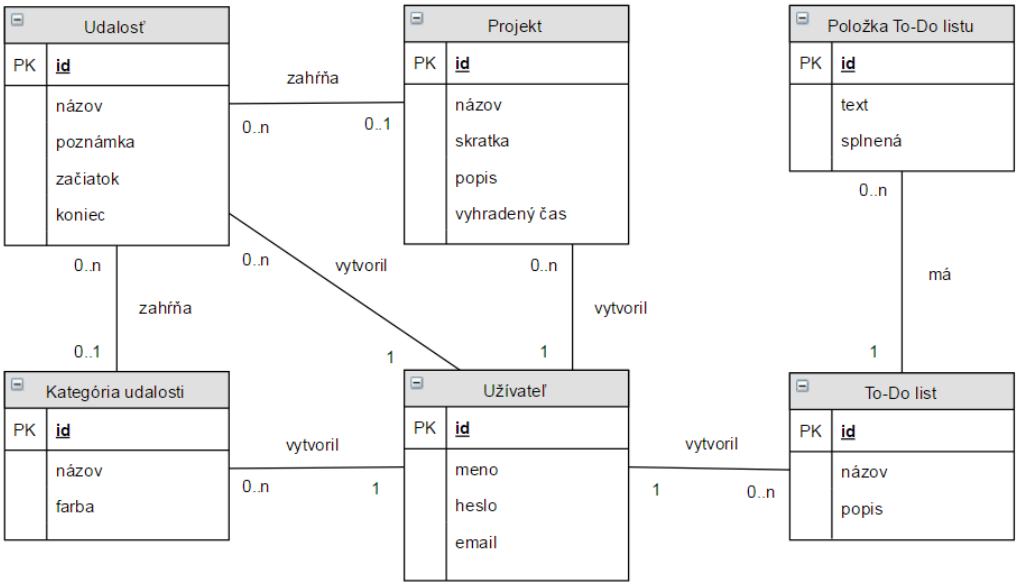
Obr. 5.1: Diagram prípadov použitia

5.3 Návrh entít

Vďaka využitiu objektovo-relačného mapovania sa stal návrh databázy ako takej irelevantným a namiesto neho je potrebný detailnejší návrh entít, ich atribútov a vzťahov medzi nimi, na základe ktorých bude databáza následne vygenerovaná pomocou knižníc Doctrine. Pravé tejto problematike je venovaná táto podkapitola. Na základe analýzy neformálnej špecifikácie bol vytvorený ER diagram 5.2 zobrazujúci jednotlivé entity a vzťahy medzi nimi.

5.3.1 Entita užívateľ

Entita užívateľ má za úlohu predstavovať užívateľský účet v aplikácii. Zachytáva základné údaje o užívateľovi potrebné pre jeho prihlásenie a prácu s aplikáciou. Taktiež sa jedná o hlavnú entitu, na ktorú sú naviazané ostatné entity v aplikácii, pretože týmto spôsobom je určené ich vlastníctvo a zobrazovanie pre daného užívateľa.



Obr. 5.2: Diagram entít a vzťahov medzi nimi

5.3.2 Entita udalosť

Táto entita predstavuje položku kalendára. Ide o časovo ohraničenú udalosť, kde položka začiatok predstavuje čas jej začiatku a položka koniec zase čas jej ukončenia. Udalosť môže taktiež obsahovať poznámku, ktorú predstavuje položka poznámka. Entita udalosť môže mať na seba naviazané ďalšie entity, ako kategória udalosti alebo projekt.

5.3.3 Entita kategória udalosti

Už z názvu je zjavné, že táto entita reprezentuje kategóriu udalosti. Má iba dve položky, a to názov a farbu kategórie. Názov má slúžiť ako jej identifikátor, napríklad pri filtrovaní a položka farba určuje farbu akou má byť daná kategória zobrazovaná v aplikácii.

5.3.4 Entita projekt

Táto entita má za úlohu reprezentovať komplexný problém, ktorý je potrebné riešiť, pričom uchováva niekoľko položiek. Položka názov má slúžiť rovnako, ako pri kategórii udalosti na identifikáciu projektu. Položka skratka zasa reprezentuje akronym projektu, ktorý je využívaný pri označení príslušnosti udalosti k danému projektu. Popis následne predstavuje špecifikáciu problému a poznámky podstatné k riešeniu príslušného projektu. Poslednou položkou je vyhradený čas. Táto položka predstavuje odhadovaný čas potrebný pre dokončenie daného projektu a môže byť využitá napríklad pre sledovanie progresu. Na entitu projektu sú taktiež naviazane ďalšie entity typu udalosť, ktoré predstavujú menšie dielčie kroky potrebné k dokončeniu projektu.

5.3.5 Entita To-Do list

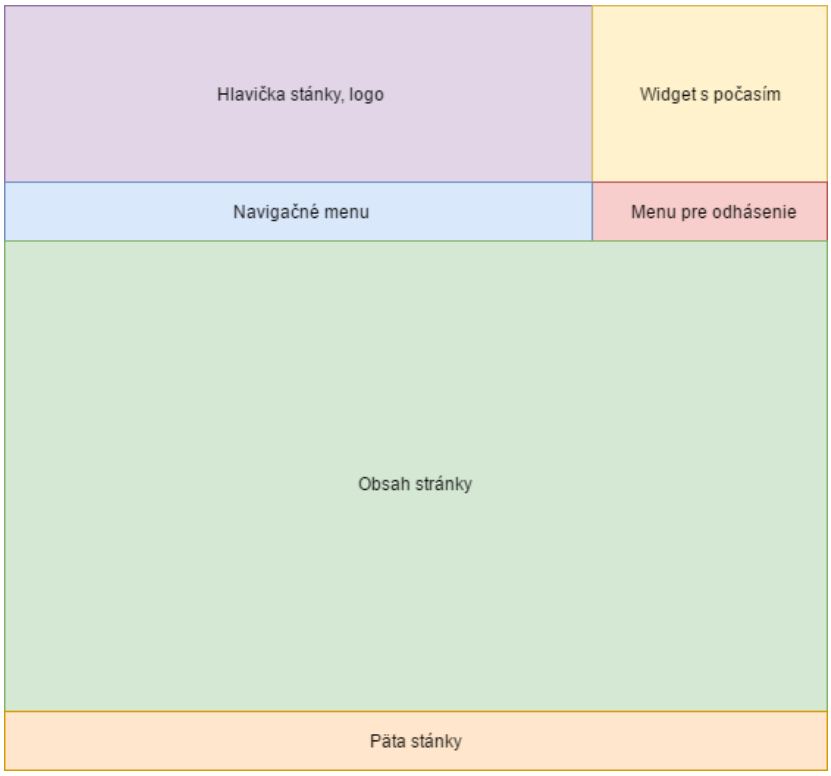
Entita To-Do list reprezentuje To-Do list užívateľa. Táto entita má dve položky, ktorými sú názov a popis, jej hlavnú časť však tvorí kolekcia entít položka To-Do listu. Ich význam je podrobnejšie objasnený v nasledujúcej podkapitole.

5.3.6 Entita položka To-Do listu

Táto entita predstavuje jednoduchú úlohu, ktorá je súčasťou väčšieho celku úloh ktorý predstavuje To-Do list. Uchováva v sebe informáciu o popise danej úlohy. Na tento účel slúži položka text. Taktiež obsahuje informáciu o tom, či bola daná úloha splnená v položke s rovnomenným názvom.

5.4 Návrh grafického rozhrania aplikácie

Nasledujúca podkapitola pojednáva o problematike grafického návrhu aplikácie. Tento návrh zahŕňa rozloženie jednotlivých blokov stránky, a taktiež návrh jednotlivých podstránok aplikácie. Toto rozloženie by malo zabezpečovať prehľadnosť aplikácie, a taktiež intuitívnosť práce s aplikáciou. Na základe týchto potrieb bol vytvorený návrh rozloženia hlavných blokov aplikácie zobrazený na obrázku 5.3, ktorý taktiež ilustruje ich význam. Tieto bloky sú nemenné pre celú aplikáciu, s výnimkou bloku obsah stránky, ktorý je rozdielny pre jednotlivé podstránky aplikácie. O tých si povieme viacej v nasledujúcich odsekoch.



Obr. 5.3: Návrh grafického rozloženia blokov aplikácie

Podstránka Calendar je zameraná na zobrazenie kalendára udalostí. Ten bude tvoriť celý obsah tejto podstránky, pretože musí zobrazovať udalosti pre celý mesiac s dostatočnou prehľadnosťou.

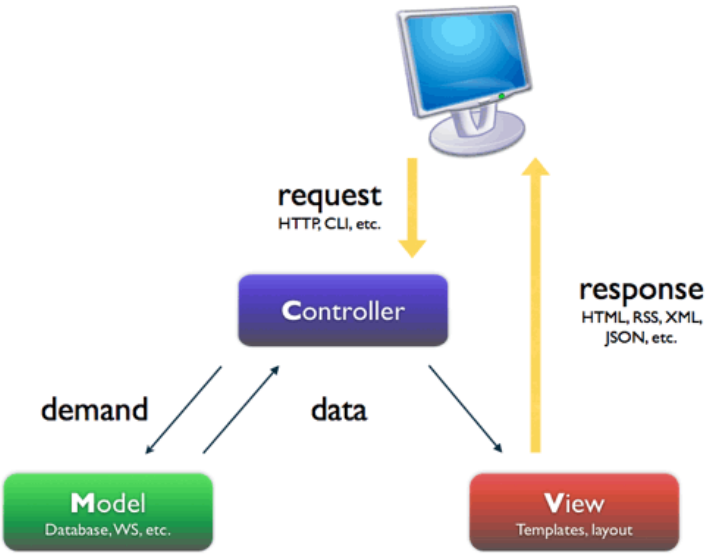
Podstránka Projects reprezentuje funkcionalitu projektov. Pre tento účel je potrebné zobrazovať menu, ktoré umožňuje prepínanie medzi jednotlivými projektmi. Ďalej musí zobrazovať detail zvoleného projektu, ktorý obsahuje názov projektu, základné údaje ako akronym projektu, čas rezervovaný pre projekt, základnú špecifikáciu projektu, a taktiež zoznam udalostí pridelených k projektu.

Podstránka To-Do lists má za úlohu zobrazovať jednotlivé To-Do listy a ich detail. Pre účel prepínania sa medzi listami je rovnako, ako pri projektoch potrebné menu. V detaile To-Do listu je potrebné zobrazovať jeho názov, popis a zoznam jednotlivých položiek listu spolu s ich popisom a aktuálnym stavom.

Podstránka Time-log je zameraná na zobrazovanie štatistík pre udalosti. Mala by taktiež zobrazovať menu pre manažovanie kategórií udalostí spolu s ich detailom.

5.5 Architektúra aplikácie

Architektúra aplikácie je zvyčajne úzko prepojená s frameworkom, na ktorom je založená a rovnako je to aj v tomto prípade. Pre tvorbu aplikácie sme zvolili framework Symfony, ktorý je založený na model–view–controller, skrátene MVC, architektonickom vzor. Tento návrhový vzor rozdeľuje aplikáciu na 3 vzájomne prepojené vrstvy. Ukážku MVC architektonického vzoru využívaného frameworkom Symfony je možné vidieť na obrázku 5.4.



Obr. 5.4: MVC architektonický vzor využívaný frameworkom Symfony [6]

Prvou vrstvou je dátový model (model), ktorý predstavuje jednotlivé objekty, ktoré sú abstrakciou reality. Tento model následne definuje formát dát, logiku a pravidlá pre prácu s nimi.

Tzv. pohľad (view) reprezentuje vrchnú vrstvu aplikácie, čiže tú ktorá je pre užívateľa viditeľná. Jeho hlavnou úlohou je zobrazovanie dát. Tieto dáta sú užívateľovi zasielané for-

mou HTTP odpovedi. Táto odpoveď má zvyčajne formu HTML stránky, ale môže sa jednať aj o objekty vo formáte JSON, ktoré sú následne spracované JavaScriptom na klientskej strane a až následne je vytvorená HTML stránka. Tento prístup je šetrnejší na prenos dát a menej zaťažuje server, ale má taktiež nevýhodu, ktorou je vyššia záťaž klientskej strany. Pre účel zobrazenia dát sa zväčša využíva už spomenutý značkovací jazyk HTML doplnený niektorým zo šablónovacích systémov, ktorým je v našom prípade Twig. Ten nám dovoľuje volať štandardne definované či nami vytvorené makrá priamo v HTML kóde a vďaka tomu umožňuje zobrazovanie rovnakých dát v rôznej forme.

Poslednú vrstvu tvorí riadiaca jednotka (controller). Tá je prostredníkom medzi dátovým modelom a pohľadom. Jej hlavnou úlohou je spracovávanie HTTP žiadostí, čiže prijímanie vstupov od užívateľa a ich následné transformovanie do príkazov, ktorým rozumie či už dátový model, alebo pohľad, záležiac na aktuálnych potrebách.[13]

Kapitola 6

Implementácia

Kapitola implementácia je venovaná procesom spojením s tvorbou aplikácie. Sú v nej zhrnuté poznatky, problémy a postupy spojené s týmto procesom. Taktiež si v nej priblížime niektoré z použitých návrhových vzorov využívaných pri objektovom programovaní. Aplikácia LifeO'clock, ktorá je výsledným produktom tejto práce implementuje najbežnejšie nástroje potrebné k efektívnemu manažovaniu času. V budúcnosti bude umožňovať, hlavne vďaka svojmu objektovému návrhu, pridávať široké spektrum funkcionalít a prvkov, o ktorých si povieme viac v závere našej práce.

6.1 Implementácia entít

Na základe ER diagramu, uvedenom v kapitole návrh, boli implementované jednotlivé entity a vzťahy medzi mini. Každú z týchto entít reprezentuje v aplikácii trieda, ktorá prestavuje dátový model danej entity. Táto trieda určuje aké druhy dát daná entita uchováva a taktiež určuje logiku práce s danou entitou. Pre vytvorenie relačných vzťahov medzi entitami boli použité Doctrine anotácie. Tieto anotácie sú do tried vkladané formou komentárov a sú taktiež využité pre kontrolu jednotlivých položiek entít. Určujú pravidlá, ktoré je potrebné dodržať, ako napríklad to, či je položka povinná alebo v prípade, že ide o reťazec znakov môže určovať napríklad jeho maximálnu dĺžku. Príklad anotácií je možné vidieť na obrázku 6.1.

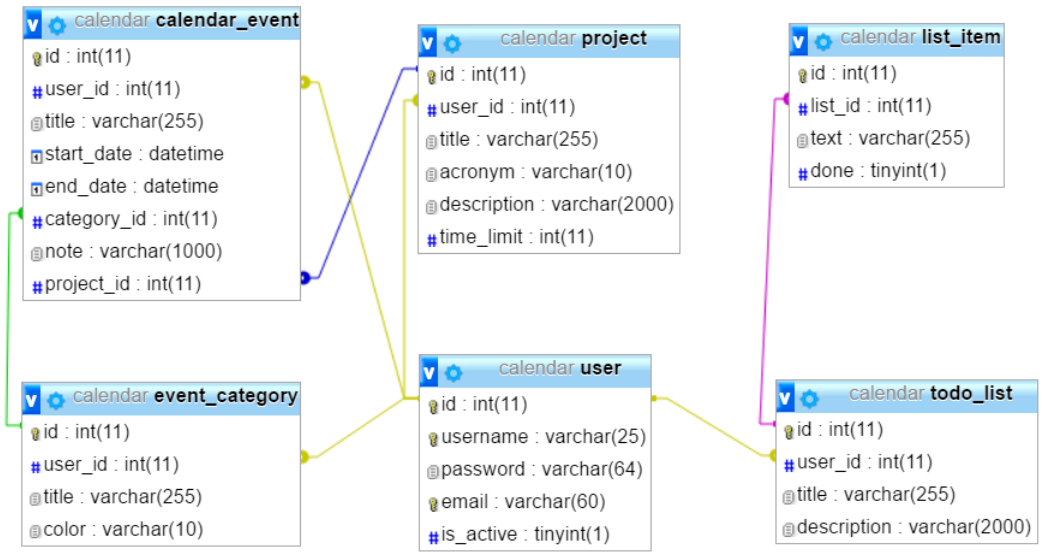
Detailnejšiemu popisu jednotlivých tried sú venované nasledujúce podkapitoly. Z týchto tried bola taktiež vygenerovaná MySQL databáza, ktorej schému ilustruje obrázok 6.2. Táto databáza využíva systém uloženia dát InnoDB.

Pre prácu s objektami týchto tried na databázovej úrovni boli vytvorené repozitáre, ktoré využívajú spoločné rozhranie, a tým je `RepositoryInterface`. Tieto repozitáre taktiež dedia z abstraktnej triedy `AbstractRepository`, ktorá implementuje metódy spoločné pre všetky repozitáre. Zvyšné metódy, ktoré sú špecifické pre každý z repozitárov implementuje ako abstraktné.

```php
/**
 * @ORM\Table(name="calendar_event")
 * @ORM\Entity
 */
class CalendarEvent implements \JsonSerializable
{
    /**
     * @ORM\Column(type="integer")
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="AUTO")
     */
    private $id;

    /**
     * @ORM\Column(type="string", length=255)
     * @Assert\NotBlank(
     *      message="Title must be filled."
     * )
     * @Assert\Length(max=100,
     *      maxMessage="Title cannot be longer than {{ limit }} characters."
     * )
     */
    private $title;
```

Obr. 6.1: Príklad použitia Doctrine anotácií



Obr. 6.2: Schéma vygenerovanej databázy

6.1.1 Trieda User

Trieda `User` v aplikácii reprezentuje užívateľský účet. Zaznamenáva o ňom niekoľko údajov v podobe nasledujúcich parametrov:

- `$id` - predstavuje identifikátor daného užívateľa. Tento identifikátor je využívaný najmä pri väzbách medzi objektmi. V rámci množiny užívateľov ide o unikátnu celočíselnú hodnotu.
- `$username` - predstavuje užívateľské meno, ktoré musí byť pre každého užívateľa unikátne, s maximálnou dĺžkou 25 znakov.
- `$password` – reprezentuje užívateľské heslo potrebné pre prihlásenie sa do aplikácie. Ide o textový reťazec s maximálnou dĺžkou 64 znakov, pre zvýšenie bezpečnosti aplikácie je pri ukladaní do databázy transformované hashovaciou funkciou bcrypt.
- `$email` - predstavuje emailovú adresu užívateľa. Jeho hodnota musí byť v rámci množiny užívateľov unikátna, s maximálnou dĺžkou 60 znakov.
- `$isActive` – určuje, či je daný užívateľský účet aktívny. Táto hodnota je kontrolovaná pri prihlasovaní, avšak v aplikácii nie je momentálne využitá. Jedná sa dátový typ `boolean` ktorý je pri vytvorení účtu automaticky nadstavená na hodnotu `true`.
- `$calendarEvents` - tento parameter reprezentuje prepojenie medzi entitami užívateľ a udalosť. V logike aplikácie uchováva kolekciu udalostí patriacich k danému užívateľskému účtu.
- `$eventCategories` - reprezentuje prepojenie medzi entitami užívateľ a kategória udalosti. V aplikácii slúži na zaznamenanie kolekcie kategórií patriacich k užívateľskému účtu.
- `$todoLists` - predstavuje prepojenie medzi entitami užívateľ a To-Do list. Uchováva v sebe kolekciu To-Do listov patriacich k danému užívateľskému účtu.
- `$projects` - tento parameter reprezentuje prepojenie medzi entitami užívateľ a projekt, a to tým spôsobom, že v sebe uchováva kolekciu projektov patriacich k užívateľskému účtu.

Trieda `User` taktiež implementuje niekoľko metód. Väčšinu z nich tvoria `get` a `set` metódy potrebné pre prácu s vyššie spomenutými parametrami. Taktiež implementuje metódy na pridávanie položiek do jednotlivých kolekcií, metódu `isActive`, ktorá overuje aktivitu účtu a ďalšie metódy vyžadované pri práci s Security komponentom frameworku Symfony.

6.1.2 Trieda CalendarEvent

Táto trieda má za úlohu popisovať jednu to základných entít aplikácie, ktorou je udalosť. Pre potreby aplikácie implementuje niekoľko nasledujúcich parametrov:

- `$id` - v rámci množiny udalostí sa jedná o unikátny celočíselný identifikátor danej udalosti.
- `$title` - reprezentuje povinný názov udalosti vo forme reťazca znakov s maximálnou dĺžkou 100 znakov.

- `$note` - tento parameter predstavuje poznámku k udalosti. Jedná sa o nepovinný textový reťazec s maximálnou dĺžkou 1000 znakov.
- `$startDate` - predstavuje čas začiatku udalosti. Ide o dátový typ `datetime`.
- `$endDate` - reprezentuje čas ukončenia danej udalosti, taktiež ide o dátový typ `datetime`.
- `$user` - tento parameter predstavuje prepojenie medzi entitami udalosť a užívateľ, a to takým spôsobom, že v sebe uchováva identifikátor užívateľa, ktorý vlastní danú udalosť.
- `$category` - slúži na vytvorenie prepojenia medzi entitami udalosť a kategória udalosti. Obsahuje identifikátor kategórie, pod ktorú daná udalosť spadá.
- `$project` - reprezentuje prepojenie medzi entitami udalosť a projekt tak, že v sebe uchováva identifikátor projektu, pod ktorý patrí.

Táto trieda taktiež implementuje všetky potrebné metódy pre prácu s týmito parametrami a taktiež implementuje metódu `jsonSerialize`, ktorá transformuje dáta objektu do tvaru,ktorý je potrebný v prípade, že ich chceme využiť aj v JavaScripte. V aplikácii je tento tvar využívaný napríklad pri dialógových oknách.

6.1.3 Trieda EventCategory

Trieda EventCategory predstavuje kategóriu udalosti. O kategórii zaznamenáva niekoľko informácií v parametroch:

- `$id` - identifikuje kategóriu udalosti v množine všetkých kategórií. Ide o unikátnu celočíselnú hodnotu.
- `$title` - predstavuje názov kategórie. Ide o textový reťazec s maximálnou dĺžkou 100 znakov.
- `$color` - tento parameter v sebe uchováva hexa kód farby, ktorá je využívaná pri zobrazovaní danej kategórie.
- `$user` - reprezentuje vzťah medzi entitami kategória udalosti a užívateľ. Pre tento účel v sebe uchováva identifikátor užívateľa, ktorý vlastní danú kategóriu.
- `$calendarEvents` - tento parameter predstavuje vzťah medzi entitami kategória udalosti a udalosť. Uchováva v sebe kolekciu udalostí patriacich pod danú kategóriu.

Trieda `EventCategory` rovnako ako iné triedy implementuje metódy pre prácu s týmito parametrami,tiež implementuje metódu `jsonSerialize` pre potreby práce v JavaScripte.

6.1.4 Trieda Project

Táto trieda popisuje entitu projekt. Pre tento účel implementuje niekoľko parametrov uchovávajúcich informácie o projekte:

- `$id` - predstavuje celočíselný identifikátor unikátny v rámci množiny projektov.

- `$title` - ide o názov projektu, ktorý tvorí textový reťazec s maximálnou dĺžkou 100 znakov.
- `$acronym` - tento parameter predstavuje skratku projektu. Táto skratka je využívaná ako prefix pre udalosti patriace pod daný projekt. Ide o textový reťazec s maximálnou dĺžkou 10 znakov.
- `$description` - uchováva v sebe popis projektu. Tvorí ho nepovinný textový reťazec s maximálnou dĺžkou 2000 znakov.
- `$timeLimit` - reprezentuje čas vyhradený pre daný projekt v hodinách. Ide o celočíselnú hodnotu v rozsahu 1 až 1000.
- `$user` - tento parameter vytvára prepojenie medzi entitami projekt a užívateľ. Pre tento účel v sebe uchováva identifikátor užívateľa, ktorému projekt patrí.
- `$calendarEvents` - predstavuje relačný vzťah medzi entitami projekt a udalosť. Obsahuje kolekciu udalostí patriacich pod daný projekt.

Táto trieda takisto implementuje metódy pre prácu s týmito parametrami a kolekciou udalostí. Opäť implementuje metódu `jsonSerialize` pre prácu s projektom v JavaScripte.

6.1.5 Trieda ToDoList

Trieda `ToDoList` popisuje, ako je už z názvu jasné, entitu To-Do listu. O tento entite uchováva niekoľko informácií vo forme parametrov:

- `$id` - jedná sa o celočíselný identifikátor ktorý je unikátny v rámci množiny To-Do listov.
- `$title` - predstavuje názov daného To-Do listu. Ten je tvorený reťazcom znakov s maximálnou dĺžkou 100.
- `$description` - uchováva v sebe popis patriaci k To-Do listu. Tento popis je tvorený reťazcom znakov s maximálnou dĺžkou 2000 znakov, a tento parameter je nepovinný.
- `$items` - tento parameter slúži na vytvorenie prepojenia medzi entitami To-Do list a položkami To-Do listu. Jeho účelom je uchovávať kolekciu položiek To-Do listu patriacich k danému listu.
- `$user` - účelom tohto parametri je vytvárať prepojenie medzi entitami To-Do list a užívateľ. Pre tento účel v sebe uchováva identifikátor užívateľa vlastniaceho daný To-Do list.

Trieda To-Do list taktiež implementuje všetky metódy potrebné pre prácu s týmito parametrami a kolekciou položiek To-Do listu.

6.1.6 Trieda ListItem

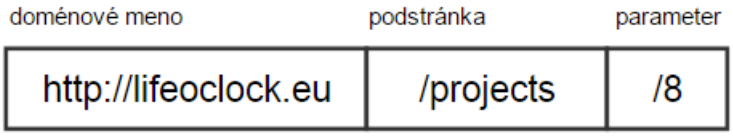
Trieda `ListItem` má za úlohu opisovať entitu položkami To-Do listu. Pre tento účel implementuje niekoľko parametrov, ktoré o nej uchovávajú potrebné informácie:

- `$id` - predstavuje celočíselný identifikátor, ktorý je unikátny v množine položiek To-Do listov.
- `$text` - tento parameter slúži na uchovanie textového obsahu danej položky. Ten je tvorený reťazcom znakov s maximálnou dĺžkou 200 znakov.
- `$done` - tento parameter označuje to, či bola daná položka To-Do listu splnená. Ide o dátový typ `boolean`.
- `$list` - slúži na vytvorenie prepojenia medzi entitami položka To-Do listu a To-Do list. Pre tento účel v sebe uchováva identifikátor To-Do listu, pod ktorý daná položka patrí.

Tak ako aj ostatné triedy aj trieda `ListItem` implementuje všetky potrebné metódy pre prácu s týmito parametrami.

6.2 Smerovanie v aplikácii

Nasledujúca podkapitola je venovaná smerovaniu v aplikácii. To je využívané na dva hlavné účely. Prvým je prepájanie URL adries s jednotlivými podstránkami zobrazujúcimi sa užívateľovi. Tým druhým účelom rozumieme prepájanie URL adries s akciami aplikácie, tento spôsob je využívaný napríklad pri spracovávaní formulárov. Dôležitou vlastnosťou pri vytváraní URL adries je ich prehľadnosť a logické usporiadanie tak, aby bola navigácia medzi stránkami jednotná a držala sa určitého vzoru, ktorý môže užívateľ v prípade potreby využiť. Dosiahnutie tohto efektu je užitočné najmä vo webových aplikáciách s veľkým množstvom podstránok. Aplikácia, ktorá je výsledkom tento práce využíva oba vyššie spomenuté účely smerovania. Tieto využívajú klasický spôsob vytvárania jednotlivých URL, kedy je k hlavnej doméne aplikácie pridaný sufix predstavujúci názov danej podstránky nasledovaný GET parametrami potrebnými pre danú podstránku. Ukážu stavby URL môžete vidieť na obrázku 6.3. Uvedený príklad predstavuje URL adresu podstránky zobrazujúcej detail projektu, ktorého identifikátor má hodnotu 8.



Obr. 6.3: Príklad stavby URL adresy

Pre účel zobrazovania sa obsahu užívateľovi je vytvorených niekoľko podstránok. Podstránky `/login` a `/register` sú voľne dostupné, a teda nepotrebujú, aby bol užívateľ prihlásený. Pre prístup ku všetkým ostatným podstránkam je potrebné prihlásenie sa k užívateľskému účtu. K tejto skupine patrí domovská podstránka, ktorá zobrazuje kalendár udalostí. Táto podstránka nemá žiaden sufix a je zobrazovaná hneď po spustený aplikácie, v prípade ak je užívateľ prihlásený inak, je presmerovaný na podstránku `/login`. Ďalej sem patrí už spomínaná podstránka so sufixom `/projects`, ktorá slúži na zobrazovanie a prácu s funkcionalitou projektov. Podobne existuje podstránky pre zobrazovanie a prácu s To-do listami. Táto má sufix `/lists`. Posledná podstránka z tento skupiny slúži na manažovanie

kategórií udalostí a zobrazovanie štatistík a je dostupná pri použití sufixu `/time_log`. Ku každej podstránke z tento skupiny je taktiež priradená šablóna, ktorá slúži na zobrazenie dát.

Druhou skupinou sú už spomínané podstránky využívané na spúšťanie akcií aplikácie. Tieto nemajú priradenú žiadnu zo šablón a predstavujú len akýsi medzi krok medzi zobrazením dvoch podstránok. Tento postup je využívaný napríklad pri spracovávaní formulárov a je detailnejšie popísaný v podkapitole 6.3.

6.3 Spracovávanie užívateľských vstupov pomocou formulárov

Jedným z hlavných účelov aplikácie je prijímanie a spracovávanie užívateľských vstupov. Najbežnejším spôsobom, ktorý je využívaný vo webových aplikáciách sú formuláre a rovnako je tomu aj tejto aplikácii. Pre tvorbu a prácu s formulármi aplikáciu využíva komponent Form, ktorý je súčasťou frameworku Symfony. Tento komponent nám ponúka komplexné a zároveň jednoduché riešenia, ktoré dopĺňa dobre spracovaná dokumentácia.

Keďže väčšinu užívateľských vstupov tvorí najmä vytváranie či úprava niektorej z entít využívaných v aplikácii, tak aj formuláre sú väčšinou viazané práve na túto entitu. Výnimkou je formulár spracovávajúci filtrovanie udalostí, ktorý nie je priamo naviazaný na žiadnu z entít, ale slúži na filtrovanie na základe závislostí medzi entitami. Pre každý z týchto formulárov aplikácia implementuje jednu triedu zo sufixom `Type`. Toto označenie je dané konvenciou frameworkuSymfony, každá z týchto tried implementuje metódu `buildForm` využívanú pre stavbu kostry formulára z jednotlivých prvkov ponúkaných už spomenutým komponentom Form. Na túto kostru je následne naviazaná daná entita. Príklad triedyformulára pre tvorbu a úpravu To-Do listov obsahujúcej metódu `buildForm` môžete vidieť na obrázku 6.4.

Tento formulár taktiež využíva tzv. embed - vnorené formuláre, čo znamená, že aspoň jednou z jeho položiek je ďalší formulár alebo ako je tomu v tomto prípade kolekcia formulárov, ktorá predstavuje jednotlivé entity položka To-Do listu. Vďaka tomuto prístupu je možné vykresľovať a spracovávať To-Do list spolu s jeho položkami ako jeden celok. Pre účely vykresľovania je objekt formulára zaslaný riadiacou jednotkou do šablóny s využitím metódy `createView` a následne je vykreslený v šablóne pomocou makier. Tieto makrá nám taktiež dovoľujú formulár dodatočne upravovať. To sa nám môže hodiť napríklad v prípade, že chceme formulár vykresliť viacnásobne, no potrebujeme rozlišovať medzi jeho jednotlivými inštanciami. Príklad vykreslenia formulára v šablóne je možné vidieť na obrázku 6.5
.

Po vyplnení a odoslaní formuláru nasleduje presmerovanie na podstránku, ktorá má na starosti jeho spracovanie. Toto presmerovanie určuje parameter formulára s názvom `action`, ktorý je nadstavený pri jeho vytváraní v riadiacej jednotke. Samotné spracovanie má na starosť iná riadiaca jednotka, ktorá prijme odpoveď, z nej získa dáta, overí ich validitu a následne vytvorí a naplní nový objekt danej entity alebo upraví ten pôvodný. Tento objekt následne uloží do databázy. V prípade chyby, ale aj úspešného spracovania je riadiacou jednotkou vygenerovaná tzv. flashmessage, čiže správa obsahujúca informácie o vykonanej akcii. Nasleduje opätovné presmerovanie na niektorú z hlavných podstránok, kde je táto správa zobrazená. Hlavnými podstránkami sú myslené tie, ktoré majú priradenú šablónu, a sú teda viditeľné pre užívateľa. Ukážku kódu, ktorý vykonáva proces spracovania je možné vidieť na obrázku 6.6.[7]

```php
class TodoListType extends AbstractType
{
    /**
     * @param FormBuilderInterface $builder
     * @param array $options
     */
    public function buildForm(FormBuilderInterface $builder, array $options)
    {
        $builder
            ->add('id', HiddenType::class, ['mapped'=>false])
            ->add('title', TextType::class, array(
                    'label' => 'List title:',
                    'attr' => array('class' => 'form-control')
                )
            )
            ->add('description', TextareaType::class, array(
                    'label' => 'Description:',
                    'required' => false,
                    'attr' => array('class' => 'form-control', 'rows' => '4')
                )
            )
            // collection of embed forms for list items
            ->add('items', CollectionType::class, array(
                    'entry_type'   => ListItemType::class,
                    'allow_add' => true,
                    'allow_delete' => true,
                )
            )
            ->add('add', SubmitType::class, array(
                    'label' => 'Submit',
                    'attr' => array('class' => 'btn btn-default btn-success width80px')
                )
            );
    }
```

Obr. 6.4: Trieda ToDoListType implementujúca metódu buildForm

```twig
<div class="row">
    {{ form_start(event_filter_form) }}
    <div class="col-md-2 col-sm-12">
        {#Category filter#}
        {{ form_label(event_filter_form.filter_category) }}
        {{ form_errors(event_filter_form.filter_category) }}
        {{ form_widget(event_filter_form.filter_category, { 'id': 'category_filter_selector'} )}}
    </div>
    <div class="col-md-2 col-sm-12">
        {#Project filter#}
        {{ form_label(event_filter_form.filter_project) }}
        {{ form_errors(event_filter_form.filter_project) }}
        {{ form_widget(event_filter_form.filter_project, { 'id': 'project_filter_selector'} )}}
    </div>
    {{ form_end(event_filter_form) }}
</div>
```

Obr. 6.5: Príklad vykreslenia formulára pre filtrovanie udalostí

```php
/**
 * @Route("/add_category", name="add_category", options = { "expose" = true })
 * @param Request $request
 * @return \Symfony\Component\HttpFoundation\RedirectResponse
 */
public function addCategoryAction(Request $request)
{
    $eventCategoryRepository = $this->get('app.event_category.repository');
    $validator = $this->get('validator');
    $user = $this->get('security.token_storage')->getToken()->getUser();

    $category = new EventCategory();
    $category->setUser($user);
    $addCategoryForm = $this->createForm(EventCategoryType::class, $category);

    $addCategoryForm->handleRequest($request);
    if ($addCategoryForm->isSubmitted()) {
        if($addCategoryForm->isValid()) {
            try {
                $eventCategoryRepository->add($category);
                $eventCategoryRepository->save();
            } catch (\Exception $e) {
                $this->addFlash(
                    'error',
                    'Unable to create category!'
                );
            }
            return $this->redirectToRoute('time_log');
        }
        else {
            $errors = $validator->validate($category);

            foreach ($errors as $error){
                $this->addFlash(
                    'error',
                    'Unable to add category: ' . $error->getMessage()
                );
            }
        }
    }
    return $this->redirectToRoute('time_log');
}
```

Obr. 6.6: Príklad spracovania formulára pre pridanie kategórie udalostí

6.4 Šablóny

Šablóny sú úzko prepojené s návrhom rozloženia aplikácie. V princípe predstavujú bloky stránky uvedené v kapitole 5.4. Tvorí ich najmä HTML kód, ktorý je doplnení JavaScriptom pre zvýšenie dynamickosti aplikácie. Pre úpravu vzhľadu jednotlivých prvkov je použitá CSS knižnica bootstrap spolu ďalšími doplňujúcimi úpravami v oddelených súboroch CSS. Pre účel využívania makier a práce s dátami zaslanými do šablóny je využitý šablónovací nastroj Twig, ktorý je úzko prepojený so samotným frameworkom Symfony.

Význam najpodstatnejších šablón je vysvetlený v nasledujúcom zozname:

- `base.html.twig` - predstavuje základnú kostru aplikácie a obsahuje všetky ostatné bloky.
- `login.html.twig` - zobrazuje samostatnú podstránku pre prihlásenie sa k užívateľskému účtu.
- `register.html.twig` - slúži na zobrazenie samostatnej podstránky pre potreby registrácie užívateľského účtu.
- `header.html.twig` - táto šablóna obaľuje viacero vnorených blokov a predstavuje hornú časť stránky, ktorá zahŕňa logo, widget s počasím, navigačné menu a menu pre odhlásenie sa užívateľa.
- `navbar.html.twig` - má za účel predstavovať navigačné menu stránky spolu s menu preodhlásenie užívateľa. Je súčasťou šablóny `header.html.twig`.
- `weatherWidget.thml.twig` - táto šablóna predstavuje blok stránky obsahujúci widget s počasím, a taktiež je súčasťou väčšieho celku, ktorý tvarý šablóna `header.html.twig`.
- `footer.html.twig` - reprezentuje blok päta stránky obsahujúci doplňujúce informácie o aplikácii.
- `index.html.twig` - ide o jednu zo šablón predstavujúcich blok- obsah stránky. V tomto konkrétnom prípade ide o domovskú stránku zobrazujúcu kalendár udalostí.
- `lists.html.twig` - taktiež ide o šablónu reprezentujúcu blok- obsah stránky. Tá má za úlohu zobrazovať detail To-do listu spolu s jeho položkami, a taktiež menu pre prepínanie medzi jednotlivými To-Do listami.
- `projects.html.twig` - rovnako ako predchádzajúce šablóny patrí aj táto do skupiny predstavujúcej blok- obsah stránky. Účelom tejto šablóny je zobrazovanie detailu projektu spolu so štatistikou ukazujúcou kategorizáciu jednotlivých udalostí patriacich pod projekt. Taktiež zobrazuje zoznam týchto udalostí a menu určené na prepínanie medzi projektmi.
- `timeLog.thml.twig` - ide o poslednú šablónu predstavujúcu blok obsah stránky. Jej účelom je zobrazovanie štatistiky časového rozdelenia udalostí do kategórií, pričom udalosti je možné vyberať na základe časového ohraničenia. Taktiež zobrazuje menu pre manažovanie týchto kategórií.

6.5 Kalendár udalostí

Srdcom výslednej aplikácie je kalendár udalostí. V jadre sa jedná o produkt tretej strany, a to tzv. FullCalendar, upravený pre potreby frameworku Symfony. Tento kalendár tvoria najmä JavaScriptové a CSS knižnice, ďalej trieda `FullcalendarEventEntity` opisujúca objekty udalostí pre potreby kalendáru a trieda `CalendarEventListener`, ktorá zachytáva žiadosť kalendára na vykreslenie udalostí. Pre tento účel vyťahuje užívateľom vytvorené objekty udalostí z databázy a následne ich transformuje na objekty udalostí využívané v samotnom kalendári. FullCalendar je napísaný hlavne v už spomínanom JavaScripte, vďaka tomu je veľmi dynamický a umožňuje zachytávať a reagovať na široké spektrum užívateľských akcií a vstupov. Toto správanie je využívané napríklad pri vytváraní udalostí, kedy stačí kliknúť na políčko dňa, pre ktorý chceme vytvoriť udalosť a následne sa automaticky zobrazí dialógové okno pre vytvorenie novej udalosti, spolu s dátumom začiatku udalosti, ktorý je prednastavený na základe zvoleného dňa. Taktiež je využívané pri zobrazovaní a editácii udalostí, kedy stačí kliknúť na danú udalosť v kalendári a následne je zobrazené dialógové oknu s detailom udalosti a možnosťou jej editácie či zmazania.

Kalendár umožňuje využitie troch druhov pohľadov pre zobrazovanie udalostí. Prvým z nich je pohľad zobrazujúci celý mesiac. Pre lepší detail je možné zvoliť pohľad pre daný týždeň prípadne až deň, ktorý tvorí tretiu možnosť. Bol taktiež upravený vzhľad kalendára pre potreby aplikácie. Farba, ktorou je udalosť zobrazovaná v kalendári, je závislá na kategórii udalosti, a ide teda o farbu, ktorá je nadstavená pre danú kategóriu. Na základe tejto farby je taktiež určená farba textu, a to tak, aby bol vždy viditeľný. Pre tento účel sú využité výpočty kontrastu v YIQ farebnom priestore. Ku kalendáru bolo taktiež pridané filtrovanie udalostí v reálnom čase. Pre tento účel boli vytvorené dva filtre, a to pre projekt a kategóriu, pod ktorú daná udalosť spadá. Tieto filtre je možné spolu kombinovať a tvorené sú Symfony formulárom, ktorý je spracovávaní JavaScriptom.

6.6 Widget s počasím

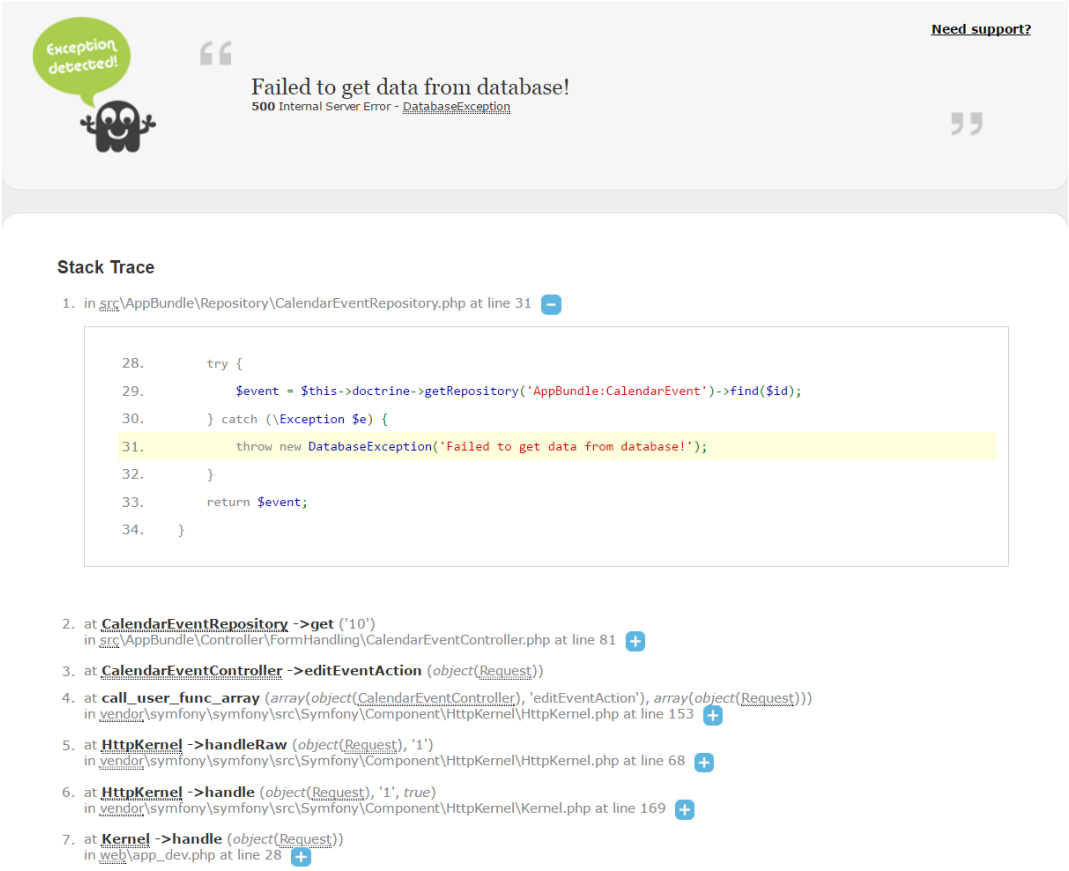
Jedným z najviditeľnejších prvkov aplikácie je widget s počasím. Ten v sebe skrýva implementáciu viacerých procesov, o ktorých si povieme viac v tejto podkapitole. Widget ukazuje počasie pre aktuálnu polohu užívateľa. Táto poloha je získavaná z jeho IP adresy, za pomoci GeoLite2, ktorá predstavuje databázu IP adries spolu s ich mapovaním na geografickú polohu. Aby sme boli presní, v našom prípade získavame z tejto databázy skratku štátu a lokalitu, kde sa užívateľ aktuálne nachádza. Po získaní polohy je vytvorený dotaz na aplikačné rozhranie, ďalej iba api, služby OpenWeaterMap, z ktorej je následne získaná odpoveď s informáciami o počasí pre danú lokalitu. Z tých sú vyselektované informácie obsahujúce aktuálnu teplotu, ktorá je následne prepočítavaná na Kelvinov na stupne Celzia, taktiež sa získava informácia o rýchlosti vetra, tlaku vzduchu, ale aj informácia o vlhkosti vzduchu a tieto sú následne pomocou základného radiča `DefaultController` zaslané globálne do všetkých šablón. Je to najmä pre to, aby ich bolo možné zobrazovať na každej podstránke. Pre zasielanie týchto dotazov je však potrebný účet, ktorý má v základnej verzii obmedzenie pre počet odoslaných dotazov. Z tohto dôvodu je odpoveď ukladaná do pamäte na dobu 30min. Ako kľuč pre opätovné získanie uloženej odpovede je použitá práve poloha pre ktorú je odpoveď určená. Vďaka tomu je znížený počet dotazov pri načítavaní stránok aplikácie, a tým je urýchlený aj jej chod.

Kapitola 7

Testovanie

Nasledujúca kapitola je venovaná testovaniu aplikácie. Bude v nej preberaný Debug komponent frameworku Symfony, testovanie widgetu s počasím a záver kapitoly bude venovaný užívateľskému testovaniu a jeho výsledkom.

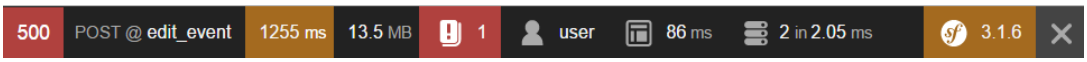
7.1 Debug komponent



Obr. 7.1: Ukážka výskytu chyby

Pre účel ladenia a overenia funkčnosti kódu bol využitý ladiaci nástroj, ktorý tvorí Debug komponent. Ten je jednou zo základných súčasti samotného frameworku Symfony. Jeho hlavným účelom je zachytávanie chýb a zobrazovanie detailnejších informácií o nich. Tieto chyby sú prevažne syntaktického charakteru, ale taktiež zachytáva neošetrené výnimky a mnohé iné druhy chýb. Poskytuje informácie o danej chybe, ako napríklad miesto vzniku a cestu programu až do miesta zistenia chyby. V určitých prípadoch obsahuje našepkávanie, napríklad v prípade ak chýba menný priestor pre niektorú z využívaných tried. Ukážku jeho zobrazenia v prípade výskytu chyby je možné vidieť na obrázku 7.1.

Jednou z hlavných súčastí Debug komponentu je taktiež tzv. debug lišta, ktorá je v prípade zapnutého debug módu zobrazovaná v spodnej časti aplikácie. Táto lišta obsahuje mnoho informácií o behu aplikácie, medzi ktoré patrí HTTP odpoveď zo serveru spolu s aktuálne použitým smerovaním pre danú stránku. Ďalej zobrazuje čas, ktorý bol potrebný pre načítanie aktuálne stránky, využitie pamäte, informácie týkajúce sa AJAX volaní, informácie o počte a chybách vykresľovaných formulárov, informácie o aktuálne prihlásenom užívateľovi, informácie vzťahujúce sa k procesu vykreslenia šablóny stránky a v neposlednom rade informácie o využití databázy a verzii frameworku. Obrázok 7.2 zobrazuje ukážku tejto lišty.



Obr. 7.2: Ukážka debug lišty

7.2 Testovanie widgetu s počasím

Súčasťou testovania aplikácie bolo taktiež testovanie widgetu s počasím. To prebehlo formou zmeny IP adresy, za pomoci zmeny VPN siete. Pre tento účel bolo otestovaných niekoľko lokácií v rozdielnych štátoch. Výsledky zobrazované widgetom je možné vidieť na obrázku 7.3.

Obr. 7.3: Výsledky testovania widgetu s počasím za pomoci VPN

Testovanie widgetu bolo taktiež súčasťou užívateľského testovania, pričom bolo zistené, že v určitých prípadoch nezobrazuje presnú polohu užívateľa. Táto chyba je pravdepodobne spôsobená tým, že nie je získaná presná IP adresa užívateľa, ale iba IP adresa jeho internetového poskytovateľa, pre ktorého umiestnenie sú následne získavané a zobrazované informácie o počasí. Pre odstránenie tejto chyby a lepšie určovanie užívateľskej polohy by mohla aplikácia v budúcnosti využívať napríklad geolokáciu, získavanú priamo z webového prehliadača, tá však nie je v starších verziách prehliadačov podporovaná.

7.3 Užívateľské testovanie

Hlavnou súčasťou testovania aplikácie bolo užívateľské testovanie. Jeho účelom bolo dokázať praktickú využiteľnosť aplikácie a zároveň odhaliť nedostatky a miesta ktorým bude pri budúcom rozvoji aplikácie potrebné venovať pozornosť. Toto testovanie prebehlo nasledujúcou formou. Štrnástim užívateľom bol predložený formulár obsahujúci 12 povinných a 1 doplňujúcu otázku ktoré boli zamerané na vzhľad, výkonnosť, využiteľnosť a úroveň práce s aplikáciou. Na väčšinu z týchto otázok bolo potrebné odpovedať formou hodnotenia známkou od 1 do 5, kde 1 predstavuje najvyššie pozitívne hodnotenie. Zvyšné otázky bolo potrebné zodpovedať výberom jednej z ponúkaných možností. Pre ich pravdivé zodpovedanie bolo potrebné, aby si užívatelia skúsili prácu s aplikáciu a jej jednotlivými funkcionalitami.

Z výsledkov tohto testovania bolo zistených niekoľko zaujímavých skutočností. Vzhľad a rozloženie aplikácie bolo hodnotené priemernou známkou 1.71 čo naznačuje väčšinovú spokojnosť avšak stále ponúka možnosť zlepšenia. Intuitívnosť aplikácie bola hodnotená rovnako známkou 1.71. Rýchlosť aplikácie bola hodnotená veľmi pozitívne a to priemernou známkou 1.28. Tento výsledok je daný najmä výkonnosťou serverov domény na ktorej je aplikácia dostupná. Z funkcionalít bol najlepšie hodnotený kalendár udalostí. Práca s ním bola hodnotená priemernou známkou 1.28 a za užitočný ho považuje až 85,7% opýtaných. Prácu s funkcionalitou To-Do listov ohodnotili užívatelia priemernou známkou 1.64 a za užitočnú ju označilo 71,4% opýtaných. Ako najmenej využiteľná sa javí funkcionalita projektov. Práca s ňou bola hodnotená priemernou známkou 2 pričom za užitočnú ju považovalo iba 50% opýtaných. Na základe tohto hodnotenia sa naskytujú 2 možnosti pre prípad budúceho rozvoja aplikácie a to buď úplné odobratie tejto funkcionality z dôvodu jej slabého využitia alebo jej prepracovanie do takej formy aby bola využívaná väčším percentom užívateľov. Hodnotený bol taktiež widget s počasím. Ten dostal priemernú známku 1.43. Posledné otázky boli venované celkovému hodnoteniu aplikácie. Aplikácia ako celok získala priemernú známku 1.71 čo približne zodpovedá priemeru jednotlivých hodnotení. K otázke či by užívatelia aplikáciu reálne využívali najčastejšie vybrali ako odpoveď možnosť z času na čas a to v 71.4% prípadoch. Pravidelne by aplikáciu využívalo 21.4% opýtaných a zvyšných 7.1% by ju využívalo iba výnimočne. Na základe doplňujúcej otázky s pripomienkami užívateľov bola do aplikácie pridaná správa o úspešnom vytvorení užívateľského účtu a možnosť pridania projektovej udalosti priamo v detaile projektu.

Testovací formulár spolu s jeho výsledkami je súčasťou príloh k práci.

Kapitola 8

Záver

Cieľom tejto práce bolo vytvorenie aplikácie Life O'clock, ktorá má za účel poskytovať nástroje pre efektívne manažovanie osobného času. Pre tento účel aplikácia implementuje funkcionalitu kalendára udalostí, projektov a To-Do listov. Ďalej zobrazuje štatistiky využitia času podľa kategórií udalostí a taktiež implementuje widget s počasím ktorý zobrazuje informácie o počasí pre aktuálnu polohu užívateľa.

Úvod práce je venovaný náhľadu do problematiky manažovania času, a nástrojom a postupom využívaným na tieto účely. Následne bola venovaná pozornosť existujúcim riešeniam a tomu čo ponúkajú.

Ďalšiu kapitolu predstavujú použité technológie. V tejto kapitole sú priblížené nástroje bežne využívané pri tvorbe webových aplikácií, ako aj objektovo orientované programovanie v PHP s využitím frameworku. Taktiež sa v nej spomína objektovo-relačného mapovanie objektov na databázové tabuľky pomocou PHP knižnice Doctrine.

Nasledujúcu kapitolu predstavuje Návrh, ktorý obsahuje neformálnu špecifikáciu, návrh rolí a návrh entít využívaných v aplikácií. Záver tejto kapitoly je venovaný grafickému návrhu aplikácie a jej architektúre.

Z kapitoly Návrh následne vychádza kapitola implementácia, ktorá je venovaná vytvoreniu samotnej aplikácie. Úvod tejto kapitoly je venovaný implementácii jednotlivých entít v aplikácií formou tried. Ďalej pojednáva o smerovaní v aplikácii, spracovávaní užívateľských vstupov, šablónach s čím je spojené zobrazovanie dát a v závere sa venuje implementácii najzaujímavejších funkcionalít aplikácie.

Posledná kapitola je venovaná testovaniu. To je zamerané na testovanie kódu pomocou Debug komponentu frameworku Symfony, spomenuté je tiež testovanie widgetu s počasím a taktiež užívateľské testovanie a jeho výsledky.

Rozsah, do ktorého bola aplikácia vyvinutá v rámci tejto práce nie je konečný. Ponúka sa nám hneď niekoľko rozšírení, ktoré sú však často náročné najmä na časové zdroje. Jedným z možných rozšírení sú notifikácie pre udalosti, ktoré môžu mať formu emailu alebo tzv. push notifikácie priamo v prehliadači. Ďalším zaujímavým rozšírením je rozšírenie užívateľského účtu o ďalšie informácie, ako je napríklad obrázok či nadstavenia k účtu spojené práve so spomenutými notifikáciami. Po vzore existujúcich aplikácií s rovnakou tematikou sa nám ponúka takisto rozšírenie umožňujúce zdieľanie, či už kalendára udalostí, projektu alebo To-Do listu s ostatnými užívateľmi, s čím by mohlo byť takisto spojené vyhľadávanie a prehliadanie účtov iných užívateľov.

Výsledná implementácia aplikácie bola nahraná na webovú doménu http://lifeoclock.eu/ kde je voľne dostupná. Pre prácu s ňou je však potrebná prvotná registrácia užívateľského

účtu. Ako ukážka vzhľadu aplikácie a práce s ňou bolo vytvorené krátke video ktoré je súčasťou príloh k práci.

Literatúra

[1] Johnson, R. E.: Frameworks = (Components + Patterns). *Communications of the ACM*, October 1997, vol. 40, no. 10, p. 41. ISSN 0002-0782.
URL
http://juacompe.mrchoke.com/natty/thesis/FrameworkComparison/frameworks=components+patterns.pdf

[2] Pheat, S. M.: *Successful Time Management*. MTD Training, 2010.

[3] Stephen Covey, A. R. M.; Merrill, R. R.: *First Things First*. Simon and Schuster, 1994.

[4] Stoilov, T.: *Time Management*. InTech, 2012, ISBN 978-953-51-0335-6.

[5] Sue W. Chapman, M. R.: *Time Management: 10 Strategies for Better Time Management*. University of Georgia, [Online; navštívené 27.12.2016].
URL http://www.fcs.uga.edu/docs/time_management.pdf

[6] Symfony Documentation: *Day 4: The Controller and the View*. [Online; navštívené 24.04.2017].
URL http://symfony.com/legacy/doc/jobeet/1_2/en/04?orm=Propel

[7] Symfony Documentation: *Forms*. [Online; navštívené 21.04.2017].
URL http://symfony.com/doc/current/forms.html

[8] Wage, J. H.: *Doctrine DBAL: PHP Database Abstraction Layer* . [Online; navštívené 14.03.2017].
URL http://jwage.com/post/31080076112/doctrine-dbal-php-database-abstraction-layer

[9] Wikipédia: Slobodná encyklopédia: *CodeIgniter*. [Online; navštívené 03.03.2017].
URL https://cs.wikipedia.org/wiki/CodeIgniter

[10] Wikipédia: Slobodná encyklopédia: *Composer (software)*. [Online; navštívené 06.03.2017].
URL https://en.wikipedia.org/wiki/Composer_(software)

[11] Wikipédia: Slobodná encyklopédia: *Doctrine (PHP)*. [Online; navštívené 14.03.2017].
URL https://en.wikipedia.org/wiki/Doctrine_(PHP)

[12] Wikipédia: Slobodná encyklopédia: *Google Calendar*. [Online; navštívené 18.04.2017].
URL https://en.wikipedia.org/wiki/Google_Calendar

[13] Wikipédia: Slobodná encyklopédia: *Model–view–controller*. [Online; navštívené 25.02.2017].
URL https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller

[14] Wikipédia: Slobodná encyklopédia: *Nette Framework*. [Online; navštívené 18.02.2017].
URL https://cs.wikipedia.org/wiki/Nette_Framework

[15] Wikipédia: Slobodná encyklopédia: *Symfony*. [Online; navštívené 18.02.2017].
URL https://en.wikipedia.org/wiki/Symfony

[16] Wikipédia: Slobodná encyklopédia: *Time Management*. [Online; navštívené 29.12.2016].
URL https://en.wikipedia.org/wiki/Time_management

[17] Wikipédia: Slobodná encyklopédia: *Zend Framework*. [Online; navštívené 19.02.2017].
URL https://en.wikipedia.org/wiki/Zend_Framework

[18] WWW stránky: *: 6 good reasons to use Symfony*. [Online; navštívené 18.02.2017].
URL http://symfony.com/six-good-reasons

[19] WWW stránky: *Centrallo*. [Online; navštívené 07.04.2017].
URL https://centrallo.com

[20] WWW stránky: *Commingly*. [Online; navštívené 08.04.2017].
URL https://commingly.com/

[21] WWW stránky: *Easynote*. [Online; navštívené 07.04.2017].
URL https://easynote.io

[22] WWW stránky: *: History of PHP* . [Online; navštívené 17.02.2017].
URL http://php.net/manual/en/history.php.php

[23] WWW stránky: *Seznámení s Nette Frameworkem* . [Online; navštívené 19.02.2017].
URL https://doc.nette.org/cs/2.4/getting-started

[24] WWW stránky: *Time Management and Calendars*. North Dakota State University, [Online; navštívené 07.01.2017].
URL https://www.ag.ndsu.edu/smallbusiness/documents/time-management-and-calendars/view

[25] WWW stránky: *Todoist*. [Online; navštívené 07.04.2017].
URL https://todoist.com

[26] WWW stránky: *Trello*. [Online; navštívené 07.04.2017].
URL https://trello.com

[27] WWW stránky: *: What can PHP do?* . [Online; navštívené 17.02.2017].
URL https://php.net/manual/en/intro-whatcando.php

[28] WWW stránky: *What Is Time Management?* Mind Tools, [Online; navštívené 27.12.2016].
URL https://www.mindtools.com/pages/article/newHTE_00.htm

[29] Zandstra, M.: *PHP Objects, Patterns and Practice*. Apress, 2010.

Príloha A

Obsah CD

A.1 lifeoclock.zip

Tento archív obsahuje všetky zdrojové súbory aplikácie. Balíčky tretých strán ktoré majú byť umiestnené v adresári vendor je však potrebné stiahnuť samostatne za pomoci nástroja Composer. Tým zaručíme že aplikácia využíva ich aktuálnu verziu.

A.2 video.zip

Tento archív obsahuje krátku video ukážku vzhľadu a práce s výslednou aplikáciou.

A.3 latex.zip

Ide o archív obsahujúci zdrojový tvar písomnej správy zo všetkými súbormi potrebnými pre jej opätovné vytvorenie a preklad do formátu PDF.

Príloha B

Manuál

B.1 Inštalácia aplikácie

Po rozbalení archívu so zdrojovými kódmi aplikácie je potrebná inštalácia baličkou tretích strán. Toho docielime spustením príkazu:

```
composer install
```

Následne je potrebné vytvorenie databázy. To sa skladá z niekoľkých krokov. Ako prvé je potrebné správne nadstavenie parametrov určujúcich pripojenie k databáze. Tie sa nachádzajú v súbore `/app/config/parameters.yml`. Po ich nadstavení môžeme vygenerovať prázdnu databázu či už ručne pomocou nástroja ako je phpMyAdmin (názov databázy sa však musí zhodovať s parametrom `database_name` zadaným v predchádzajúcom kroku) alebo príkazom:

```
php bin/console doctrine:database:create
```

Následne je možné vygenerovať databázové tabuľky aplikácie za pomoci príkazu:

```
php bin/console doctrine:schema:update -force
```

Po tomto kroku je aplikácia pripravená na použitie.

Príloha C

Užívateľské testovanie - formulár

C.1 Otázky k užívateľskému testovaniu

Práca s aplikáciou - Kalendár udalostí *

pridanie/úprava/zmazanie udalosti v kalendári

	1	2	3	4	5	
výborné	○	○	○	○	○	veľmi zlé

Využiteľnosť funkcionality - Kalendár udalostí *

- ○ užitočná
- ○ čiastočne použitelná
- ○ zbytočná

Práca s aplikáciou - To-Do listy *

pridanie/úprava/zmazanie To-Do listu a jeho položiek

	1	2	3	4	5	
výborné	○	○	○	○	○	veľmi zlé

Využiteľnosť funkcionality - To-Do listy *

- ○ užitočná
- ○ čiastočne použitelná
- ○ zbytočná

Práca s aplikáciou - Projekty *

pridanie/úprava/zmazanie Projektu a udalostí k nemu priradených

	1	2	3	4	5	
výborné	○	○	○	○	○	veľmi zlé

Využiteľnosť funkcionality - Projekty *

- ○ užitočná
- ○ čiastočne použitelná
- ○ zbytočná

Widget s počasím *

umiestnenie, vzhľad, rozloženie, informačná hodnota

	1	2	3	4	5	
výborné	○	○	○	○	○	veľmi zlé

Celkový dojem z aplikácie *

	1	2	3	4	5	
výborné	○	○	○	○	○	veľmi zlé

Využívali by ste túto aplikáciu ? *

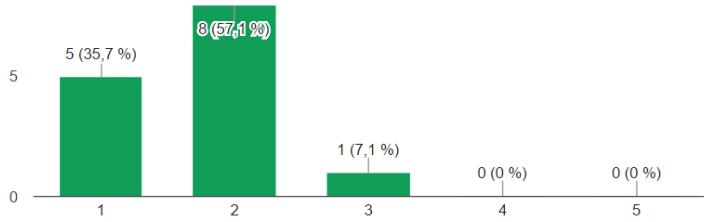
- ○ pravidelne
- ○ z času na čas
- ○ iba výnimočne
- ○ vôbec

Dodatočné pripomienky - čo by ste zmenili/odstránili/pridali...

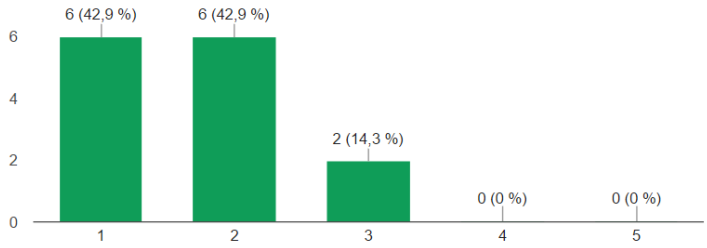
Text dlhej odpovede

C.2 Odpovede na otázky k užívateľskému testovaniu

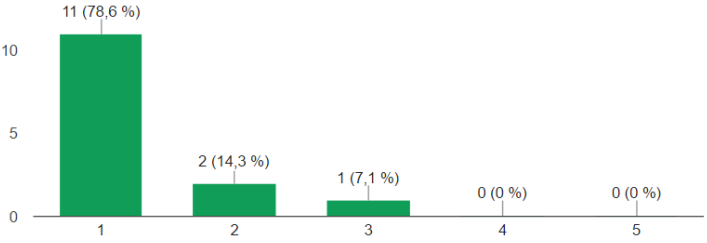
Grafický vzľad/rozloženie aplikácie (14 odpovedí)



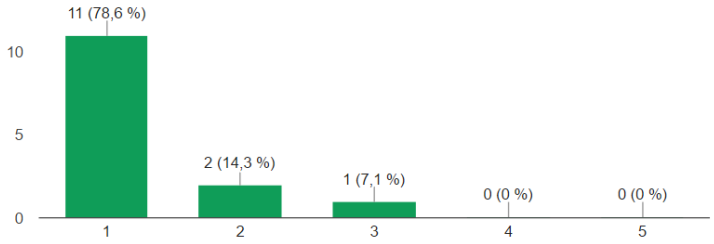
Intuitívnosť/prehľadnosť aplikácie (14 odpovedí)



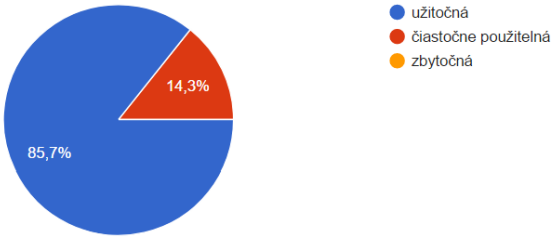
Rýchlosť/výkonosť aplikácie (14 odpovedí)



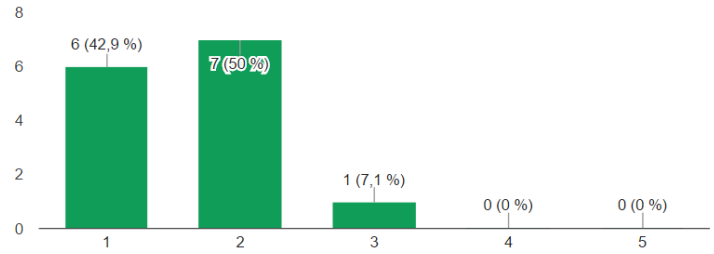
Práca s aplikáciou - Kalendár udalostí (14 odpovedí)



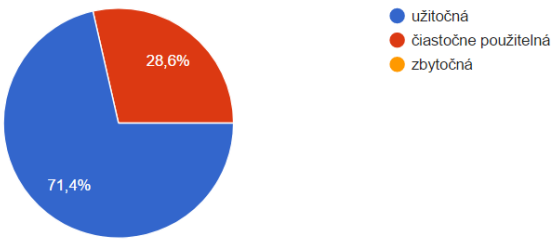
Využiteľnosť funkcionality - Kalendár udalostí (14 odpovedí)



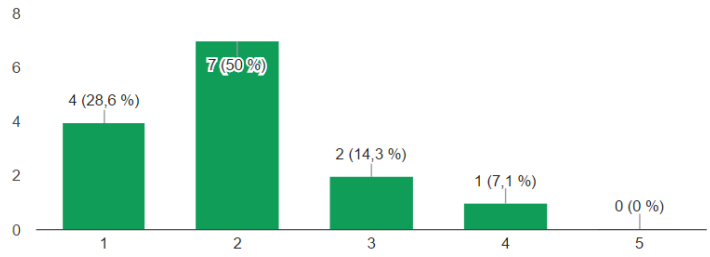
Práca s aplikáciou - To-Do listy (14 odpovedí)



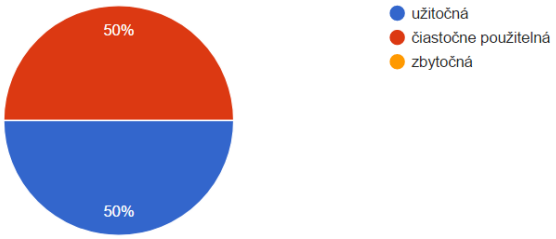
Využiteľnosť funkcionality - To-Do listy (14 odpovedí)



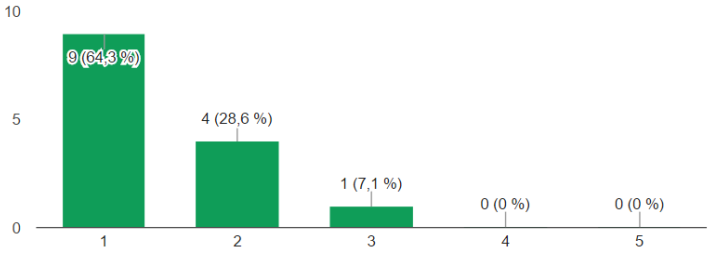
Práca s aplikáciou - Projekty (14 odpovedí)



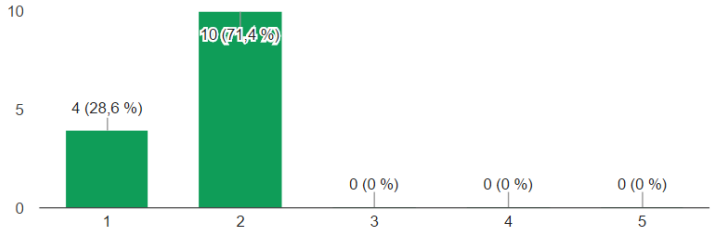
Využiteľnosť funkcionality - Projekty (14 odpovedí)



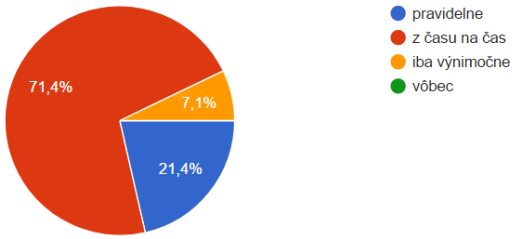
Widget s počasím (14 odpovedí)



Celkový dojem z aplikácie (14 odpovedí)



Využívali by ste túto aplikáciu ? (14 odpovedí)



Dodatočné pripomienky - čo by ste zmenili/odstránili/pridali... (2 odpovede)

Pri registracii by to chcelo nejaku odpoved "succesfully registered" alebo nieco take...."manage categories" by som dal asi vedla "projects" ako sucast menu alebo nejako tak ....lebo som to nemohol najst ...inak celkom fajn

V casti Projects neviem ako mam zadat Poject events. To do list ak je splneny na 100%, mohlo by sa to zmenit z cerveneho krizika na nieco ine.