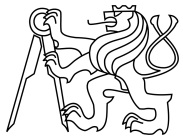
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Systém pro správu výkresových štítků
Student:	Tomáš Bučko
Vedoucí:	Ing. Petr Špaček, Ph.D.
Studijní program:	Informatika
Studijní obor:	Informační systémy a management
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2018/19

Pokyny pro vypracování

Cílem práce je navrhnout a implementovat webový systém pro správu výkresových štítků (tzv. rozpisek výkresu), který bude podporovat všechny případy užití existujícího řešení nasazeného v podniku spolupracujícím se zadavatelem a po dohodě s vedoucím práce.
1. Student se v počátcích práce seznámí s existujícím řešením pro správu výkresových štítků.
2. V rámci tvorby vlastní projektové dokumentace, se student pokusí eliminovat návrhové nedostatky, kterými současné řešení trpí.
3. Následně, pomocí metod softwarového inženýrství a refactoringu, navrhne a implementuje systém nový, který bude mít architekturu typu SPA a bude dodržovat pravidla dobrého návrhu tak, aby vylepšil udržitelnost a rozšiřitelnost systému jako takového. Nároky na použité technologie jsou: php & symfony, javascript & angularjs.
4. Návrh nového systému student v projektové dokumentaci zhodnotí jak z hlediska technického (procesního) přínosu, tak z hlediska nákladů na vývoj a provoz nového systému.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 12. října 2017

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalárska práca

Systém pre správu výkresových štítkov

Tomáš Bučko

Katedra softwarového inženýrství
Vedúci práce: Ing. Petr Špaček, Ph.D.

8. mája 2018

Poďakovanie

Ďakujem vedúcemu mojej práce, pánovi Ing. Petrovi Špačkovi PhD, za ochotu, čas a odborné vedenie, ktoré mi na konzultáciach venoval. Moje poďakovanie ďalej patrí oponentovi mojej práce, pánovi Ing. Davidovi Bernhauerovi, za konzultáciu v oblasti zabezpečenia webových aplikácií.

Prehlásenie

Prehlasujem, že som predloženú prácu vypracoval(a) samostatne a že som uviedol(uviedla) všetky informačné zdroje v súlade s Metodickým pokynom o etickej príprave vysokoškolských záverečných prác.

Beriem na vedomie, že sa na moju prácu vzťahujú práva a povinnosti vyplývajúce zo zákona č. 121/2000 Sb., autorského zákona, v znení neskorších predpisov, a skutočnosť, že České vysoké učení technické v Praze má právo na uzavrenie licenčnej zmluvy o použití tejto práce ako školského diela podľa § 60 odst. 1 autorského zákona.

V Prahe 8. mája 2018 .....

Odkaz na túto prácu

Bučko, Tomáš. *Systém pre správu výkresových štítkov*. Bakalárska práca. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2018.

Abstrakt

Predmetom tejto práce je analýza existujúceho webového systému nasadeného v podniku, eliminácia prípadných nedostatkov, a následný návrh a implementácia nového, vylepšeného systému. Výsledkom práce je nový systém, ktorý dodržuje štandardy softvérového inžinierstva a podporuje všetky prípady použitia pôvodného systému. Nedostatky pôvodného systému boli eliminované. Práca je rozdelená do troch častí – analýza, návrh a implementácia. V závere práce sa nachádza zhodnotenie prínosov a nedostatkov nového systému.

Klíčová slova Symfony, AngularJS, webový systém, webová aplikácia, PHP parser, MVC, REST API, JWT

Abstract

The subject of this thesis is the analysis of the existing web system deployed in a company, elimination of its potential weaknesses, followed by the design and implementation of a new, enhanced system. The outcome of the thesis is a system which adheres to software engineering standards and supports all of the use cases of the previous system. The weaknesses of the previous system has been eliminated. The thesis is divided into three parts – analysis, design and implementation. In the end of the thesis, there's a conclusion of benefits and shortcomings of the implemented system.

Keywords Symfony, AngularJS, web system, web application, PHP parser, MVC, REST API, JWT

Obsah

Úvod **1**

1 Cieľ práce **3**

2 Analýza **5**
2.1 Požiadavky na systém 5
2.2 Postup práce 5
2.3 Prvotná analýza 6
2.4 Nedostatky systému 6
2.5 PHP Parser 8
2.6 Analýza funkcionality 9
2.7 Dátový model existujúcej aplikácie 11

3 Návrh **13**
3.1 Použité technológie 13
3.2 Zabezpečenie 13
3.3 Návrh dátového modelu novej aplikácie 15
3.4 Používateľské rozhranie 17
3.5 Architektúra 20

4 Implementácia **23**
4.1 Použité nástroje 23
4.2 Back end 23
4.3 Zabezpečenie 25
4.4 Front end 27

Záver **31**
Vlastnosti nového systému 31
Nedostatky nového systému 32
Prínosy nového systému 33

Literatúra **35**

A Zoznam použitých skratiek **39**

B Návod na použitie **41**

C Obsah priloženého CD **43**

Zoznam obrázkov

2.1 Zjednodušený model existujúcej aplikácie 6
2.2 Ukážka zdrojového kódu starého systému 7
2.3 Diagram dátového modelu existujúcej aplikácie 12

3.1 Diagram dátového modelu novej aplikácie 16
3.2 Prehľad akcií 17
3.3 Detail akcie 18
3.4 Detail rozpisky 19
3.5 Prehľad záznamov 20

4.1 Tabuľka entity Node 24
4.2 Schéma procesu autentizácie a autorizácie 26
4.3 Stavový diagram pohľadov aplikácie 27
4.4 Strom vykreslený pomocou Angular Tree Control 28

Zoznam tabuliek

4.1 Prehľad operácií s entitami 25

Úvod

Informačné systémy sa v dôsledku informatizácie spoločnosti stali štandardným nástrojom pre podporu činnosti podnikov, úradov, a iných organizácií. Pod pojmom informačný systém, si mnoho ľudí predstaví komplexný softvér, ktorý používateľom umožňuje manipulovať s dátami uloženými v databáze, bežiacej na samostatnom serveri. Takýto systém však nemusí mať nutne formu dedikovaného počítačoveho programu. U podnikov, kde elektronický informačný systém nie je potrebný na vykonávanie podnikateľskej činnosti, je častokrát lacnejšie a praktickejšie viesť evidenciu v textovom súbore, či pomocou tabuľkového procesoru. Za informačným systémom, ktorý má byť pre podnik prínosom, sa totiž okrem samotnej implementácie a nasadenia skrýva aj analýza agendy podniku, znalosť požiadaviek zadávateľa, a štúdia realizovateľnosti. Môžeme teda konštatovať, že informačný systém je systém zberu, uchovania, analýzy a prezentácie dát, ktorý poskytuje informácie rôznym používateľom.

Kapitola 1

Cieľ práce

Cieľom práce je navrhnúť a implementovať webový systém pre správu výkresových štítkov (rozpisiek výkresov) založený na existujúcom systéme nasadenom v podniku. Nový systém bude podporovať všetky prípady použitia existujúceho systému, navyše v ňom budú eliminované nedosatatky, ktorými existujúci systém disponuje. Nový systém bude mať MVC architektúru. Back end systému bude implementovaný ako Symfony aplikácia, komunikujúca s front endom pomocou API. Front end bude implementovaný v AngularJS, bude mať formu SPA.

Kapitola 2

Analýza

2.1 Požiadavky na systém

Zo zadania práce sú zrejmé nasledujúce nefunkčné požiadavky na systém:

- Systém má mať architektúru typu SPA.
- Návrh systému má umožnovať jednoduchú udržovateľnosť a rozšíriteľnosť systému.
- Systém má byť implementovaný pomocou technológií Symfony a AngularJS.

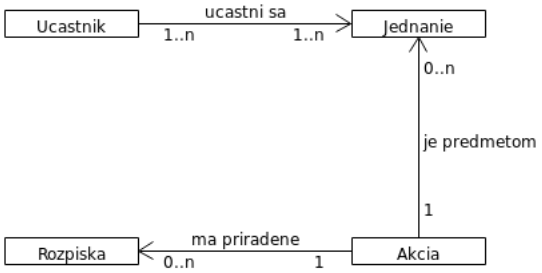
Jediným funkčným požiadavkom na systém je zo zadania podpora všetkých prípadov použitia existujúceho systému, ktorým je venovaná samostatná kapitola. Žiadne iné funkčné ani nefunkčné požiadavky nie sú známe, a preto pri návrhu a implementácií nového systému budú zohľadnené len požiadavky uvedené v zadaní práce.

2.2 Postup práce

Prvým krokom je zoznámenie sa s existujúcim systémom, počas ktorého bude identifikovaná funkcionalita a prípady použitia, ktoré systém umožňuje. Nasleduje analýza zdrojového kódu aplikácie, v ktorej budú identifikované nedostatky, ktorými systém disponuje. Ďalším krokom je návrh nového systému tak, aby v ňom boli eliminované nedostatky súčasného systému. Následne bude nový systém implementovaný, čím bude naplnený cieľ práce. Po implementácií nasleduje zhodnotenie zhodnotenie nového systému ako z hľadiska technického prínosu, tak z hľadiska nákladov na vývoj a prevádzku.

2.3 Prvotná analýza

Po prvotnom zoznámení sa s dátovým modelom a zdrojovým kódom existujúcej aplikácie bol navrhnutý nasledujúci jednoduchý model 2.1, v ktorom sú zachytené ústredné entity systému. Diagram zároveň načrtáva základnú funkcionalitu systému, ktorá sa vždy týka niektorej z entít tohto modelu.



Obr. 2.1: Zjednodušený model existujúcej aplikácie

Objekt akcia je ústrednou entitou systému. Jedná sa o udalosť súvisiacu s architektúrou, resp. stavbou. Príkladom môže byť úsek diaľnice, či cyklotrasa. Akcia má priradené rozpisky (výkresové štítky) usporiadané do stromovej štruktúry. Ďalšou entitou je jednanie akcie, ktoré má priradených účastníkov. (zainteresované úrady alebo organizácie) Systém umožňuje export rozpisiek, generovanie zoznamov entit a dokumentov s prednastaveným obsahom. V systéme neboli identifikované žiadne používateľské roly.

2.4 Nedostatky systému

Systém má podobu PHP aplikácie s používateľským rozhraním v HTML, ktorá komunikuje s MSSQL databázou. Zdrojový kód aplikácie má na dnešné pomery nevhodnú štruktúru. Hlavný súbor obsahuje rozsiahly switch čakajúci na akciu, ktorá sa nastaví v závislosti na interakcií používateľa. Po nastavení akcie sa následne zo switchu zavolajú príslušné funkcie, ktoré sú definované v pomocných súboroch. Každá z funkcií najprv pomocou SQL syntaxe vyberie dáta z databázy a programová logika v PHP nad nimi vykoná príslušné operácie. V prípade, že sa jedná o operácie typu update alebo delete, uskutočnia sa ďalšie SQL operácie, ktoré odošlú upravené dáta naspäť do databázy. Následne funkcia vygeneruje HTML tabuľku s výslednými dátami. Toto riešenie má niekoľko očividných problémov. Štruktúra funkcií využívajúca rôzne technológie má za následok, že funkcie sú samoúčelné, čo znemožňuje ich znovupoužitie. Mixovanie SQL, PHP a HTML syntaxe (viď. obr. 2.2) má ďalej za následok neprehľadnosť zdrojového kódu. To sa prejavuje obtiažnou spätnou

revíziou a analýzou kódu, najmä pokiaľ analýzu vykonáva nezainteresovaná osoba. Pri zmenách v dátovom modeli je nutné upravovať zdrojový kód všetkých funkcií, ktorých sa zmena týka. V prípade, že systém vyhadzuje chybu, alebo nesprávny výstup, sa obtiažne určuje, ktorá časť zdrojového kódu chybu zapríčiňuje. Bolo identifikovaných aj niekoľko nedokonalostí z funkčného hľadiska. Systém umožňuje filtrovať zobrazené akcie podľa troch filtrov, ktoré fungujú nezávisle na sebe, nie je možné filtrovať akcie podľa viacerých filtrov zároveň. Ďalšou vlastnosťou systému, ktorú je možné vylepšiť, je začlenenie rozpisiek do štruktúry, ktorá je obmedzená na dve úrovne.

```
function over_rozpisku()
{
  if (isset($_GET['rok'])&&$_GET['rok']!='') $test_rok=substr($_GET['rok'],2);
    else $test_rok='';
  if (isset($_GET['nazev_overeni'])&&$_GET['nazev_overeni']!='')
    $sql= "SELECT id, left(cislozakazky,6) as cislozakazky, nazev
    FROM TabZakazka WHERE CisloZakazky Like '".$test_rok."%-01-000'
    AND nazev Like '%".$_GET['nazev_overeni']."%' Order by id;";
  else $sql= "SELECT id,left(cislozakazky,6) as cislozakazky, nazev
    FROM TabZakazka WHERE CisloZakazky
    Like '".$test_rok."%".$_GET['id_overeni']."%-01-000' Order by id;";
    $con=viewercon_helios();
  $rs= mssql_query ($sql, $con);
  if (!$rs) {
        die('Probl�m s proveden�m dotazu datab�ze Helios! - Ov��en� rozpisek');
    }
  if (mssql_rows_affected($con, $rs)!=0){
    ?>
      <script type="text/javascript">
  <!--
        function dotaz(formular){
            var kopiruj = confirm("M� se prov�st kop�rov�n� struktury
                                   rozpisek z jin� existuj�c� akce?");
            if (kopiruj) {
            document.forms[formular]['kopirovat'].value="ANO";
             document.forms[formular].submit();
            } else {
            document.forms[formular].submit();
            }
         }
  //-->
  </SCRIPT>
    <TABLE cellspacing="0" cellpadding="0" valign="middle" border="0" width="100%">
    <tr>
      <th width="50%" align="left"> Byly nalezeny tyto z�znamy:</th>
    </tr>
    <tr><td>
      <TABLE cellspacing="1" cellpadding="1" valign="middle" border="0" width="100%">
        <tr>
            <th width="15px">Po�.�.</th><th width="20px">Rozpiska</th>
            <th width="20px">Mo�nosti</th>
            <th>��slo akce</th><th>N�zev</th>
        </tr>
    <?PHP

    $poradi=0;
    while ($row = mssql_fetch_array($rs, MSSQL_ASSOC))
      {
      ?>
      <tr class="vypis" onMouseOver="this.style.backgroundColor='#fdf9e3'"
                        onMouseOut="this.style.backgroundColor='#CDEBEB'">
      <td align="center"><?PHP echo $poradi+=1;?>.</td>

//...
```

Obr. 2.2: Ukážka zdrojového kódu starého systému

2.5 PHP Parser

Súčasťou analýzy existujúceho systému bolo využitie PHP parseru. Myšlienkou bolo z existujúceho zdrojového kódu vygenerovať AST, prejsť vygenerovaný strom, pomocou návrhového vzoru visitor modifikovať želané listy stromu, a následne zo stromu spätne vygenerovať zdrojový kód. Prieskum ukázal, že existuje pomerne málo PHP parserov. Jedná sa o nasledujúce parsery:

- PHP parser od nikic [1]
- PHP parser od TysonAndre [2]
- PHP parser od glayzzle [3]

Všetky uvedené požiadavky splňoval len PHP parser od používateľa nikic. Medzi uvedenými parsermi je tiež najlepšie zdokumentovaný a má najväčšiu používateľskú základňu. Na základe navrhnutého modelu v sekcií 2.3 boli pomocou parseru z jednotlivých súborov oddelené a prerozdelené funkcie do nových súborov reprezentujúcich entity modelu. (akcia, rozpiska, jednanie, účastník) Tieto súbory reprezentujú kontroléry nového systému, obsiahnuté funkcie definujú minimálnu funkcionalitu, ktorou majú príslušné kontroléry nového systému disponovať. Po parsovaní tieto funkcie lepšie zapadajú do kontextu systému. Zároveň bola z jednotlivých funkcií oddelená HTML syntax do samostatných súborov. Tieto súbory určujú funkcionalitu formulárov používateľského rozhrania v novom systéme, po odstránení HTML syntaxe sa tiež zlepšila čítateľnosť jednotlivých funkcií.

2.6 Analýza funkcionality

2.6.1 Všeobecná funkcionalita

F1 Prihlásiť sa

F2 Odhlásiť sa

2.6.2 Akcia

F3 Zobraziť posledných 10 akcií zmenených v systéme

F4 Zobraziť posledných 10 akcií zmenených prihláseným používateľom

F5 Filtrovať akcie podľa časti čísla akcie

F6 Filtrovať akcie podľa časti názvu akcie

F7 Filtrovať akcie zhora podľa roku

F8 Zobraziť detail akcie

F9 Upraviť akciu

F10 Zobraziť stromovú štruktúru začlenenia rozpisiek akcie

F11 Rozbaliť stromovú štruktúru začlenenia rozpisiek akcie

F12 Zbaliť stromovú štruktúru začlenenia rozpisiek akcie

F13 Nahradiť časť názvu všetkých objektov štruktúry začlenenia

F14 Kopírovať rozpisky objektu štruktúry začlenenia do iného objektu

F15 Exportovať všetky rozpisky akcie

F16 Nahradiť dátum všetkých rozpisiek akcie

F17 Dokopírovať rozpisky z inej akcie

2.6.3 Rozpiska

F18 Vytvoriť novú rozpisku

F19 Zobraziť detail rozpisky

F20 Upraviť rozpisku

F21 Zmeniť začlenenie rozpisky do objektu

F22 Zmazať rozpisku

F23 Exportovať rozpisku

2.6.4 Objekt štruktúry začlenenia

F24 Zobraziť prehľad objektov

F25 Vytvoriť nový objekt

F26 Upraviť objekt

F27 Zmazať objekt

2.6.5 Dodatok

F28 Zobraziť prehľad dodatkov

F29 Zobraziť detail dodatku

F30 Upraviť dodatok

F31 Priradiť dodatok všetkým rozpiskám akcie

2.6.6 Export

F32 Vygenerovať hlavnú obálku

F33 Vygenerovať zoznam objektov štruktúry začlenenia

F34 Vygenerovať zoznam rozpisiek

F35 Vygenerovať ISO dokument týkajúci sa akcie

2.6.7 Jednanie

F36 Zobraziť prehľad jednaní

F37 Vytvoriť nové jednanie

F38 Upraviť jednanie

F39 Zmazať jednanie

F40 Priradiť účastníka jednaniu

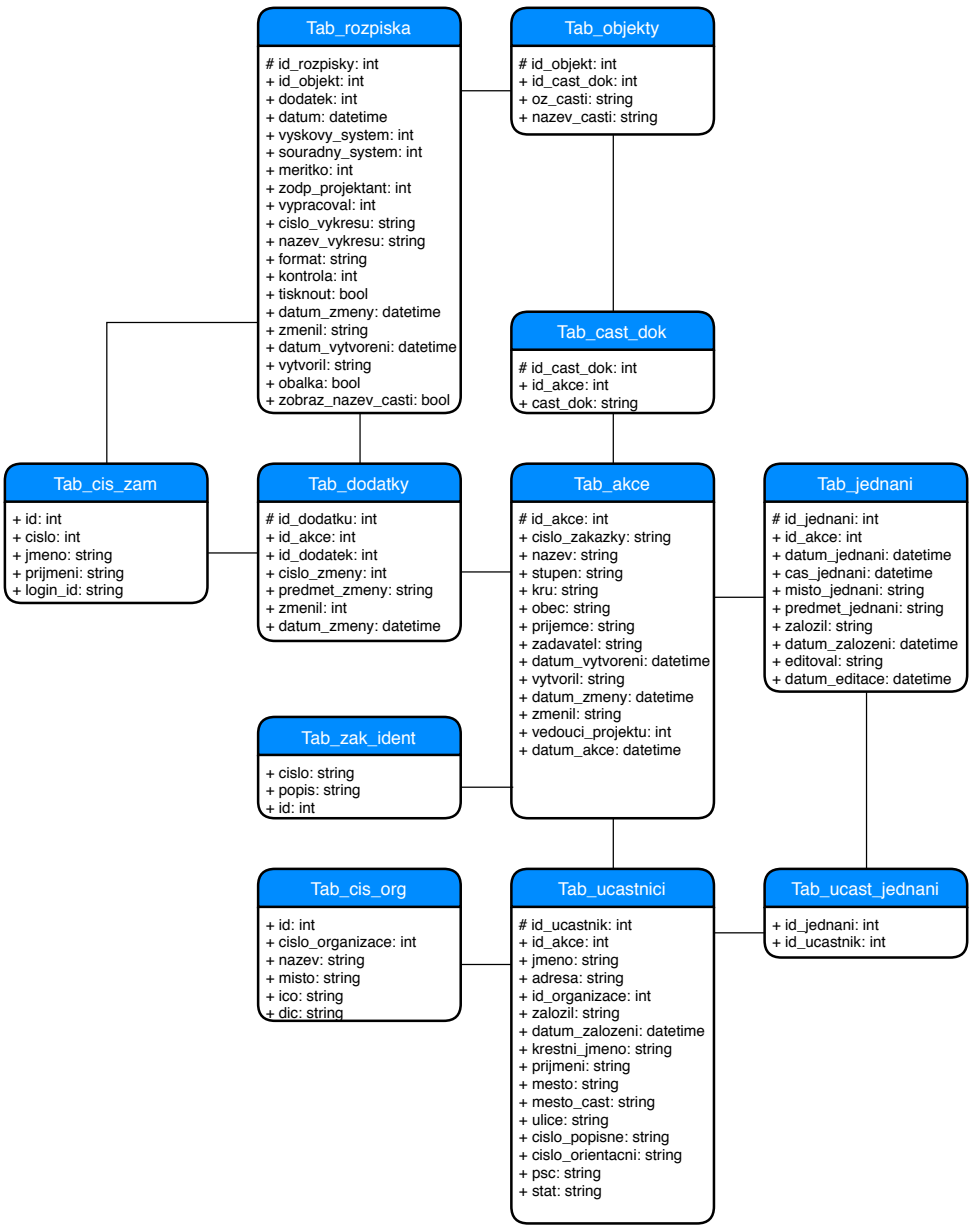
F41 Vygenerovať dokument týkajúci sa jednania

2.7 Dátový model existujúcej aplikácie

Na úvod je nutné poznamenať, že navrhnutý diagram dátového modelu existujúcej aplikácie 2.3 je len odhadom skutočného dátového modelu. Diagram bol navrhnutý na základe analýzy tabuliek databázy a zdrojového kódu aplikácie. V uvedenom diagrame nie sú zachytené všetky entity. Vypustené boli predovšetkým číselníky, ktoré boli z diagramu vynechané z dôvodu prehľadnosti. Ďalej boli z diagramu vypustené entity, ktoré aplikácia vôbec nevyužíva, a preto nemajú pre potreby analýzy žiadny prínos.

2.7.1 Entity

Centrálnou entitou modelu je akcia. (Tab_akce) Akcia má priradené rozpisky (Tab_rozpiska) začlenené v stromovej štruktúre s dvoma úrovňami. (Tab_cast_dok, Tab_objekty) Akcia má ďalej priradené dodatky, (Tab_dodatky) ktoré reprezentujú zmeny v akcií. Každá rozpiska môže mať priradený niektorý z dodatkov príslušnej akcie. Dodatky a rozpisky čerpajú niektoré položky z číselníka zamestnancov. (Tab_cis_zam) Akcia má tiež priradené jednania a účastníkov (Tab_jednani, Tab_ucastnici). Tabuľka účastníkov jednania čerpá údaje z číselníka organizácií. (Tab_cis_org) Vzťah účastníkov a jednaní je dekomponovaný do entity Tab_ucast_jednani.



Obr. 2.3: Diagram dátového modelu existujúcej aplikácie

Kapitola 3

Návrh

3.1 Použité technológie

Zo zadania sú dané technológie, ktoré majú byť použité na implementáciu nového systému. Jedná sa o frameworky Symfony [24] a AngularJS. [23] Silnou stránkou oboch frameworkov je reputácia a rozšírenie. Príkladom webových služieb vybudovaných na Symfony je Spotify, Trivago, či Dailymotion. [8] Medzi služby využívajúce AngularJS patrí PayPal, Netflix, či ProtonMail. [9] Oba frameworky sú aktívne vyvíjané a dobre zdokumentované vrátane množstva ukážok a tutoriálov na internete. Nevýhodou frameworkov vo všeobecnosti je pomerne strmá krivka učenia, čo najmä u neskúsených programátorov implikuje závislosť na dokumentácií a tutoriáloch na internete. Ďalšou nevýhodou frameworkov sú časté zmeny v rozhraní pri vydaní novej verzie. [10] Extrémnym prípadom tohto javu sú zmeny medzi AngularJS a Angular 2. Jedná sa o prakticky odlišné frameworky. Zatiaľ čo AngularJS je založený na JavaScript, Angular 2 je založený na TypeScript. Medzi ďalšie odlišnosti novej verzie Angularu patrí modularita, podpora reaktívneho programovania a odlišná štruktúra aplikácií vytvorených v novej verzií. [11] Keďže Symfony a AngularJS nie sú vzájomne kompatibilné, bude na prenos dát medzi back a front endom použité API. Dáta budú prenášané vo formáte JSON.

3.2 Zabezpečenie

Medzi najrozšírenejšie metódy zabezpečenia API webových aplikácií patrí v dnešnej dobe autentizácia pomocou:

- HTTP Basic
- API kľúč
- JWT
- OAuth

3.2.1 HTTP Basic

[5] Jedná sa o najjednoduchšiu metódu zabezpečenia. Autentizácia prebieha odoslaním používateľského mena a hesla oddeleného dvojbodkou ako reťazec zašifrovaný pomocou Base64. Reťazec je odoslaný ako súčasť HTTP hlavičky, ktorá má nasledujúcu podobu:

```
Authorization: Basic dXNlcm5hbWU6cGFzc3dvcmQ=
```

Výhodou HTTP Basic je podpora všetkých bežne používaných prehliadačov a jednoduchosť implementácie. Na druhej strane je HTTP Basic najmenej bezpečná metóda, pretože posiela prihlasovacie údaje ako cleartext, čo sa však dá mitigovať použitím HTTPS protokolu. Ďalšou vlastnosťou, ktorú treba zvážiť, je fakt, že prehliadač si prihlásenie pamätá až do vypnutia.

3.2.2 API kľúč

[6] API kľúč je unikátny reťazec, ktorý je súčasťou hlavičky alebo URL požiadavku odosielaného klientom. Jeho účelom je identifikácia používateľa, ktorý požiadavku odosiela. Je možné použitie privátneho a verejného kľúča, kde privátny kľúč je uložený na serveri a funguje ako heslo, pričom verejný kľúč sa prikladá do odosielaných požiadaviek. Podobne ako u HTTP Basic, je na odosielanie API kľúčov nutné použiť šifrovaný protokol.

3.2.3 JWT

[7] JSON Web Token je JSON reťazec definovaný štandardom, ktorý obsahuje zašifrované dáta, často sa používa na autentizáciu. Autentizácia pomocou JWT funguje tak, že po odoslaní používateľského mena a hesla a úspešnej autentizácií server vygeneruje a vráti klientovi JWT, ktorý je nutné uložiť do local storage, alebo cookie súboru. Klient následne do hlavičky každého odoslaného požiadavku prikladá uložený JWT. Obsah hlavičky zvyčajne vyzerá nasledovne:

```
Authorization: Bearer eyJhbGciOiJIUzI1NiJ9.
eyJ1c2VybmFtZSI6InUiLCJwYXNzd29yZCI6InB3In0.
4MSQVmMDz5ZD2TmKMopvGqI2pQE11_KYFuzXU_fgIEE
```

Server po obdržaní požiadavky na zabezpečený zdroj skontroluje platnosť priloženého JWT, po úspešnej kontrole klientovi sprístupní požadované dáta. JWT má obmedzenú platnosť, po vypršaní platnosti je nutné sa znovu prihlásiť.

3.2.4 OAuth 2.0

[6] V autentizácií pomocou OAuth 2.0 vystupujú tri strany:

- autentizačný server

- klient
- API server (obsahuje dáta, ktoré chce klient sprístupniť)

Klient najprv odošle na autentizačný server kľúč. Po úspešnej autentizácií server vráti klientovi odpoveď (HTTP 302), ktorá obsahuje access token a presmeruje klienta na API server. Klient pri odoslaní požiadavku na API server do hlavičky požiadavku prikladá access token. API server pri každom prijatom požiadavku skontroluje platnosť priloženého access tokenu, po úspešnej kontrole sprístupní klientovi požadované dáta. Access token má obmedzenú platnosť, po vypršaní platnosti je nutné sa znovu prihlásiť.

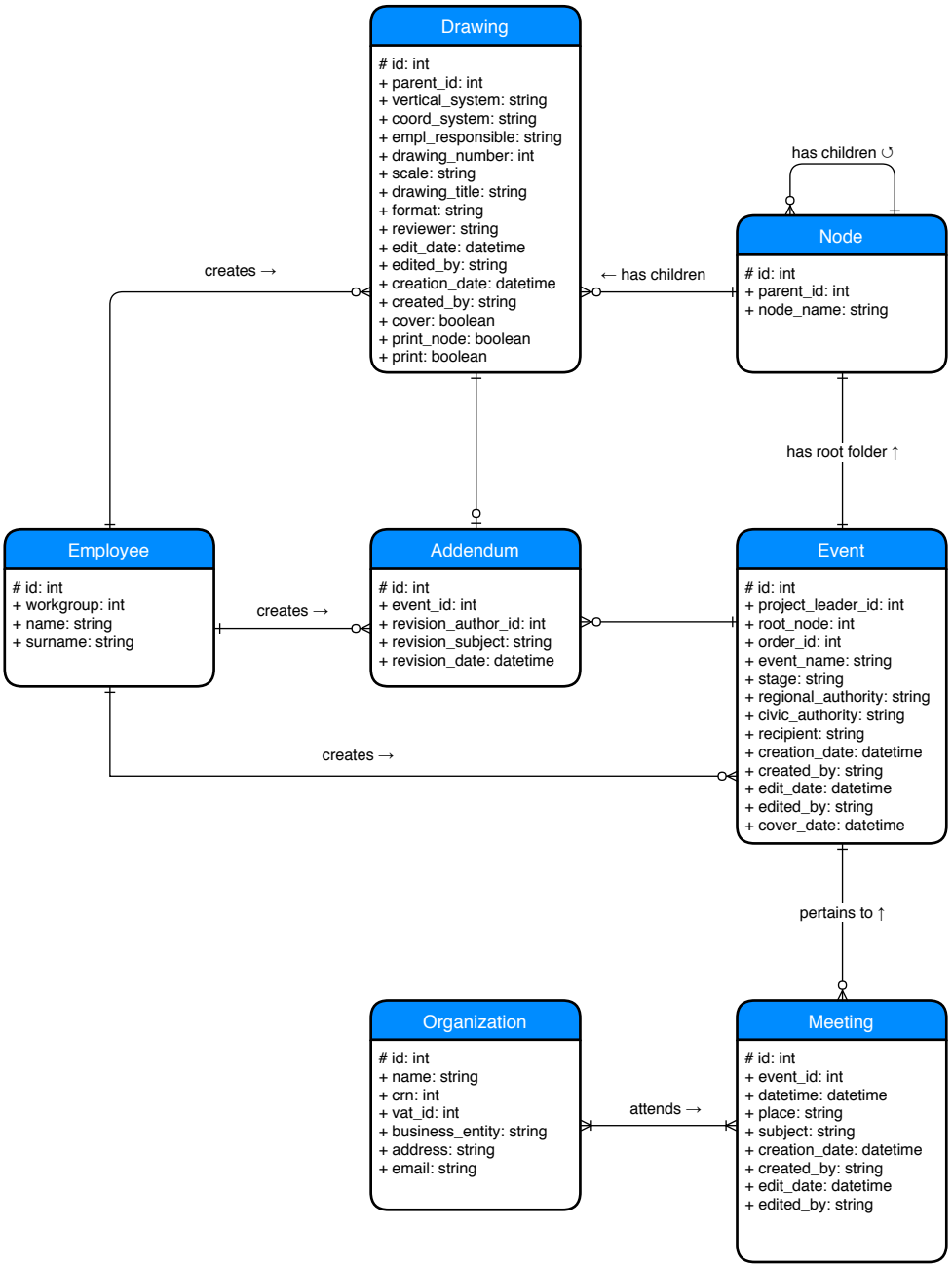
3.2.5 Voľba metódy autentizácie

Po zvážení dostupných možností autentizácie padla voľba na JWT. Metóda nemá žiadné očividné nedostatky. V porovnaní s OAuth 2.0 je jednoduchšia na implementáciu a prevádzku. V porovnaní s HTTP Basic je praktickejšia a bezpečnejšia. Pôvodným účelom API kľúčov nie je autentizácia, ale identifikácia používateľa. JWT navyše dodržuje štandard prenosu dát vo formáte JSON, ktorý bude v systéme implementovaný.

3.3 Návrh dátového modelu novej aplikácie

Ústrednou entitou navrhnutého modelu bude aj v novom systéme akcia (Event). Akcia môže mať priradené dodatky (Addendum) a jednania. (Meeting) Zmeny v novom diagrame sa týkajú predovšetkým začlenenia rozpisiek. Navrhnutá bola entita uzol (Node) s položkou parent_id, ktorá bude držať informáciu o rodičovskom uzle. Takto navrhnutá entita umožňuje vytvoriť stromovú štruktúru s neobmedzenou hlbkou. Každá akcia má priradený práve jeden koreňový uzol, ktorý je (pra)rodičom všetkých ďalších uzlov a rozpisiek akcie. V entite rozpiska (Drawing) obdobne pribudla položka parent_id. Rozpiska má ďalej voliteľný jeden z dodatkov nadradenej akcie. Medzi jednaním a účastníkom (Organization) zmizla dekompozícia účasti jednania, použitý framework si dekompozíciu vygeneruje automaticky. Nový datový model bude podobne ako ten starý obsahovať aj potrebné číselníky, ktoré slúžia ako množina preddefinovaných hodnôt, ktoré sa dajú nastaviť v iných entitách. Jedná sa o výškový systém, súradný systém a mierku, využívané entitou rozpiska. Entita akcia využíva číselník stupeň. Poslednou entitou modelu je zamestnanec, (Employee) ktorý reprezentuje prihláseného používateľa. Zamestnanec je taktiež súčasťou každej akcie, rozpisky a dodatku ako autor príslušnej entity. Dátový model bude v novom systéme implementovaný za pomoci Doctrine. Jedná sa o súbor PHP knižníc, ktoré plnia úlohu ORM. [12] Doctrine teda zaisťuje mapovanie PHP objektov reprezentujúcich entity na relačnú databázu. Triedy sú mapované na tabuľky databázy, členské premenné tried sú mapované na stĺpce

príslušnej tabuľky. Doctrine je kompatibilný s väčšinou populárnych databázových serverov, ako sú MySQL, Oracle, či MSSQL. V novej aplikácií bude použitý MySQL server.



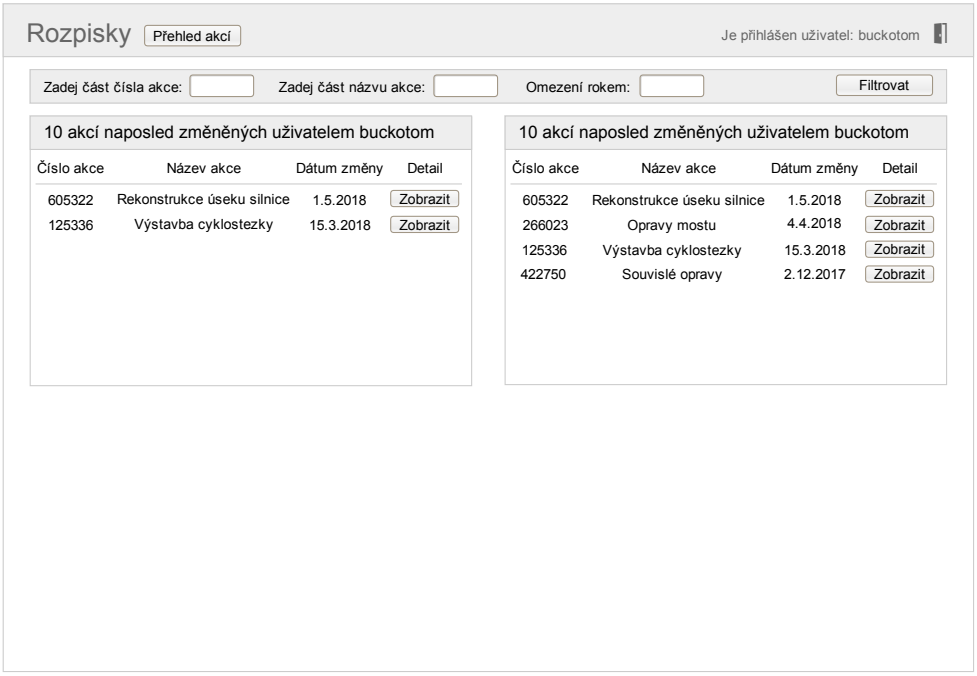
Obr. 3.1: Diagram dátového modelu novej aplikácie

3.4 Používateľské rozhranie

Používateľské rozhranie nového systému bude inšpirované rozhraním existujúceho systému. Všetky pohľady systému budú obsahovať lištu, ktorá bude slúžiť na navigáciu v systéme. Dáta všetkých pohľadov budú zobrazené ako tabuľky.

3.4.1 Prehľad akcií

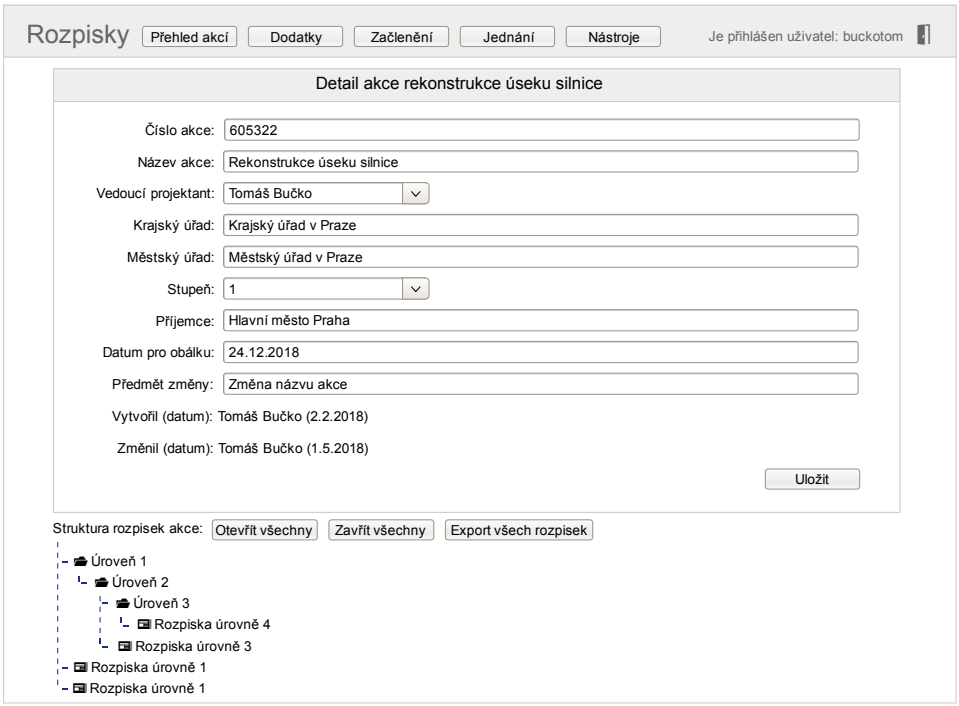
Po prihlásení sa používateľovi zobrazí prehľad akcií. Na stránke sa bude nachádzať prehľad posledných 10 akcií zmenených prihláseným používateľom a posledných 10 akcií zmenených v celom systéme. Zobrazené akcie bude možné filtrovať. Z prehľadu akcií bude možné vyvolať detail vybranej akcie.



Obr. 3.2: Prehľad akcií

3.4.2 Detail akcie

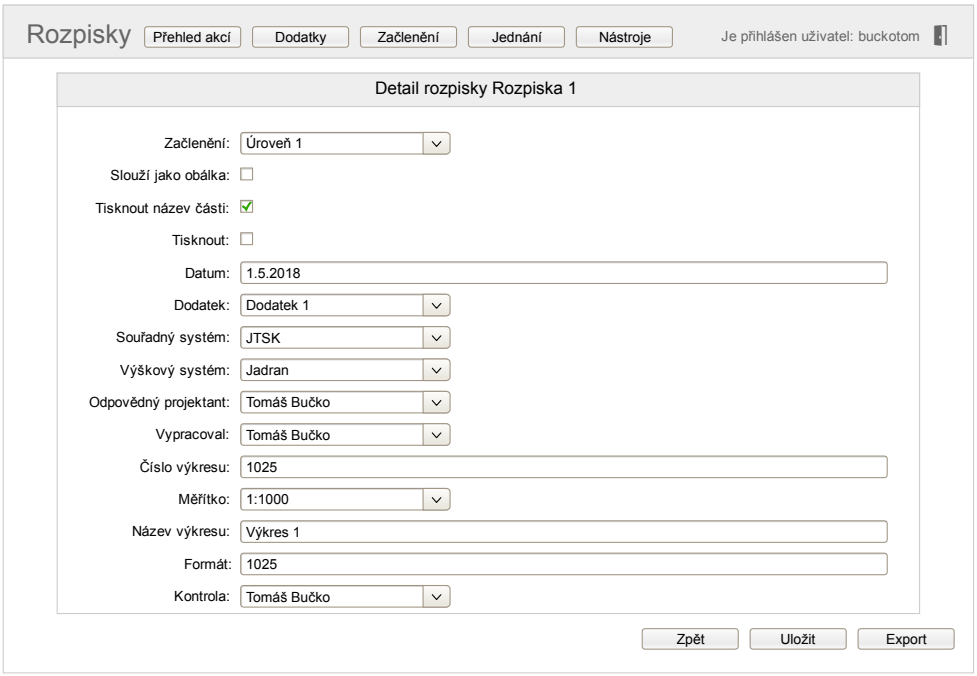
Stránka s detailom akcie bude obsahovať tabuľku s údajmi vybranej akcie, v pohľade bude možné upraviť jej údaje. Pred každým uložením zmien bude nutné vyplniť povinnú kolonku predmet zmeny. Po potvrdení zmien systém zároveň akcií vytvorí nový dodatok, ktorý bude obsahovať vyplnený predmet zmeny. Pod tabuľkou s údajmi sa bude nachádzať stromová štruktúra rozpisiek vybranej akcie s možnosťou rozbalenia a zbalenia všetkých objektov obsahujúcich rozpisky. Po kliknutí na rozpisku v strome sa zobrazí pohľad s jej detailom. Pohľad ďalej bude tiež obsahovať tlačítko pre export všetkých rozpisiek akcie. V kontexte akcie zvolenej v prehľade akcií sa zároveň v lište zobrazia nové položky. Jedná sa o prehľad zmien, začlenenia a jednaní zvolenej akcie. Položka nástroje bude obsahovať hromadné kopírovanie rozpisiek do inej akcie, export dokumentov a doplnkovú funkcionalitu aplikácie, ktorú sa nehodí začleniť inde.



Obr. 3.3: Detail akcie

3.4.3 Detail rozpisky

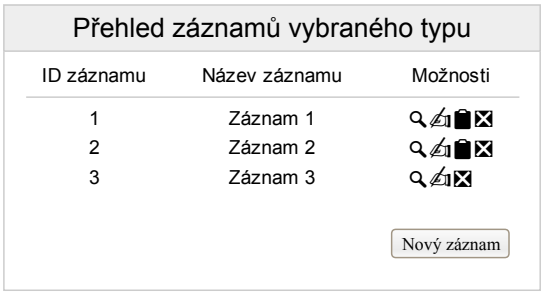
Po kliknutí na rozpisku v stromovej štruktúre sa vyvolá detail vybranej rozpisky. Pohľad bude slúžiť na úpravu údajov vybranej rozpisky. Okrem iného tu bude možné zmeniť začlenenie rozpisky v stromovej štruktúre a exportovať príslušnú rozpisku.



Obr. 3.4: Detail rozpisky

3.4.4 Prehľad záznamov

Položky lišty prehľad zmien, prehľad začlenenia a prehľad jednaní budú mať jednotný vzhľad. V každom prehľade bude zobrazená tabuľka s prehľadom záznamov vybraného typu, odkiaľ bude v závislosti na implementácií možné so záznamom vykonávať jednu alebo viac CRUD operácií.



Obr. 3.5: Prehľad záznamov

3.5 Architektúra

Nový systém bude členený do modulov zodpovedajúcich architektúre MVC. Modul pohľadov je popísaný v sekcií 3.4. Dátovému modelu je venovaná sekcia 3.3. Táto sekcia je venovaná modulu kontrolérov. Na základe analýzy boli navrhnuté nasledujúce hlavné kontroléry back endu:

- Kontrolér akcie (pokrýva F3, F4, F5, F6, F7, F8, F14, F32, F35)
- Kontrolér zamestnancov (používateľov)
- Kontrolér rozpisiek (pokrýva F14, F15, F16, F17, F18, F19, F20, F21, F22, F23, F31, F34)
- Kontrolér začlenenia (pokrýva F10, F11, F12, F13, F24, F25, F26, F27, F33)
- Kontrolér jednaní (pokrýva F36, F37, F38, F39, F40, F41)
- Kontrolér účastníkov (pokrýva F40)
- Kontrolér zmien (pokrýva F28, F29, F30)
- Kontrolér autentizácie (pokrýva F1, F2)

Na základe navrhnutých pohľadov v sekcií 3.4 budú v systéme prítomné minimálne tieto kontroléry front endu:

- Kontrolér prehľadu akcií (pohľad 3.4.1)
- Kontrolér detailu akcie (pohľad 3.4.2)
- Kontrolér detailu rozpisky (pohľad 3.4.3)
- Kontrolér prehľadu zmien (pohľad 3.4.4)
- Kontrolér prehľadu rozpisiek (pohľad 3.4.4)
- Kontrolér prehľadu objektov štruktúry začlenenia (pohľad 3.4.4)
- Kontrolér prehľadu jednaní (pohľad 3.4.4)
- Kontrolér autentizácie

Kapitola 4

Implementácia

4.1 Použité nástroje

Vývoj webových aplikácií sa v dnešnej dobe nezaobíde bez IDE. Na odporučenie vedúceho práce bol na implementáciu použitý nástroj Visual Studio Code. [22] Silnými stránkami tohto IDE sú multiplatformovosť, integrovaná správa verzií (štandardne Git), integrovaný príkazový riadok, veľké množstvo dostupných zásuvných modulov vrátane integrovaného vyhľadávania a celková používateľská prívetivosť rozhrania a ovládania IDE. Textová časť práce bola vytvorená použitím zásuvného modulu LaTeXWorkshop. [20] Pri tvorbe dátového modelu bol na vizualizáciu databázy použitý nástroj MySQL Workbench. [21] Na správu verzií a zdieľanie zdrojového kódu s vedúcim práce bol použitý spomenutý Git.

4.2 Back end

4.2.1 Entity a kontroléry

Entity nového systému, ich atribúty a vzťahy medzi entitami zodpovedajú diagramu na obrázku 3.1. Entity systému, ktoré nie sú v diagrame zahrnuté, sú číselníky VerticalSystem, CoordinateSystem, Scale a Stage. Všetky štyri číselníky podporujú operáciu Read. Pre potreby autentizácie je v systéme implementovaná entita User s atribútmi username, email a plainPassword, od ktorej dedí entita Employee. V systéme bola ďalej implementovaná dedičnosť u entity Drawing, ktorá dedí od entity Node. Dedičnosť bola implementovaná použitím class table inheritance. Jedná sa o mapovaciu stratégiu, kde všetky triedy jednej hierarchie sú mapované na vlastnú tabuľku, aj všetky rodičovské tabuľky. Tabuľka potomka je s tabuľkou rodiča spojená pomocou cudzieho kľúča. Tabuľka rodiča obsahuje tzv. discriminator column, ktorý obsahuje informáciu o tom, akého typu sú jednotlivé riadky tabuľky. [26] Princíp class table inheritance bol implementovaný na dosiahnutie polymorfného správania entít.

V systéme to umožňuje na objektovej úrovni entite Node do objektu children priradiť entity typu Node, aj Drawing. Stratégia class table inheritance je v systéme implementovaná nasledovne:

```php
<?php
namespace App\Entity;

/**
  * @ORM\Entity
  * @ORM\InheritanceType("JOINED")
  * @ORM\DiscriminatorColumn(name="discr", type="string")
  * @ORM\DiscriminatorMap({
  *      "node" = "Node", "drawing" = "Drawing"
  * })
  */
class Node
{
  //...
}

/**
 * @ORM\Entity
 */
class Drawing extends Node
{
  //...
}
```

Listing 4.1: Class table dedičnosť entít Node a Drawing

Tabuľka entity Node s načítanými ukážkovými dátami vyzerá nasledovne:

id	parent_id	name	discr
1	7	Technická zpráva	drawing
2	8	Celková situace stavby	drawing
3	10	Bilance zemin a ornice	drawing
4	NULL	Root folder	node
5	4	Dokumenty	node
6	5	Dopravní značení	node
7	6	Dopravní opatření	node
8	5	Pěší komunikace	node
9	4	Silniční kanalizace	node
10	9	Rekultivace staré silnice	node
11	4	Vegetační úpravy	node
12	NULL	Root folder	node
13	NULL	Root folder	node
NULL	NULL	NULL	NULL

Obr. 4.1: Tabuľka entity Node

Každá entita systému má svoj kontrolér, v ktorom je definovaná logika operácií s entitou. Implementované boli len tie operácie, ktoré novému systému umožňujú dosiahnutie funkcionality pôvodného systému. Operácie sú zhrnuté v nasledujúcej tabuľke:

Entity	Create	Read	Update	Delete
Addendum	*	*	*	
Drawing	*	*	*	*
Employee		*		
Event		*	*	
Meeting	*	*	*	*
Organisation		*		
Node	*	*	*	*

Tabuľka 4.1: Prehľad operácií s entitami

Súčasťou systému je súbor AppFixtures.php obsahujúci ukážkové dáta. Tie je možné načítať pomocou Doctrine Fixtures Bundle. [17] Jedná sa o nástroj umožňujúci načítanie testovacích dát do databázy pre účely vývoja, testovania a demonštrácie aplikácie.

4.2.2 API

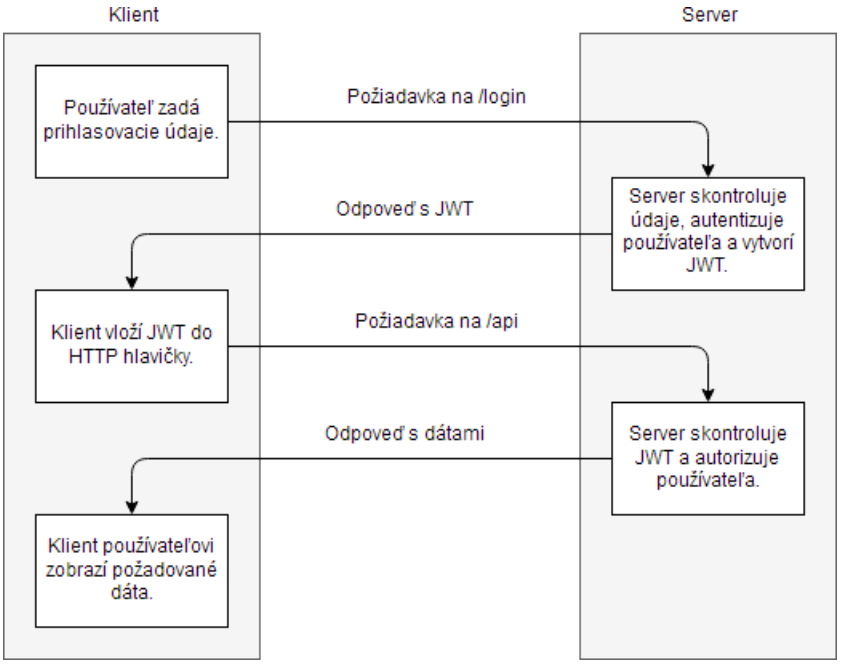
Back end komunikuje s front endom pomocou REST API. REST je architektúra, ktorá umožňuje pristupovať k dátam a uskutočňovať nad nimi CRUD operácie. Každý zdroj dát má vlastný prístupový bod. REST je bezstavový, čím jednak značne zjednodušuje komunikáciu s API a umožňuje paralelné spracovanie obsahu. V neposlednom rade poskytuje určitý štandard, takže nie je problém použiť cudzie API alebo naopak poskytovať vlastné veľkému množstvu ďalších používateľov. [16] Výmena dát medzi back endom a front endom systému prebieha použitím formátu JSON, syntax má nasledujúcu podobu:

```
{ "username":"buckotom", "password":"secret" }
```

4.3 Zabezpečenie

V systéme je zabezpečenie implementované použitím JWT. [13] [14] Proces autentizácie začína vyplnením prihlasovacieho formuláru. Používateľ na front ende vyplní a odošle meno a heslo. Kontrolér back endu po obdržaní požiadavky vyhľadá používateľa v databázi. Pri úspechu kontrolér ďalej porovná obdržané heslo so zahašovaným heslom uloženým v atribúte password entity User. Ak je heslo platné, kontrolér vygeneruje JWT [15] a vráti ho používateľovi v odpovedi. Kontrolér front end skontroluje, či odpoveď obsahuje JWT.

Pri úspechu kontrolér do local storage uloží používateľské meno a JWT, vďaka čomu ostane používateľ prihlásený aj po aktualizácií stránky v prehliadači. Kontrolér následne upraví hlavičku všetkých HTTP požiadavkov tak, aby obsahovala JWT. Back end takto pozná, že používateľ bol úspešne autentizovaný. Po vypršaní platnosti JWT aplikácia používateľa presmeruje na prihlasovací formulár. Autorizácia funguje na základe access control. V systéme má každý prihlásený používateľ rolu ROLE_USER, back end autorizuje prístup k API práve na základe tejto roly. Používatelia bez tejto roly majú prístup iba k prihlasovaciemu formuláru. V systéme je na ukážku implementovaná aj registrácia používateľov, ktorá je v priloženom zdrojovom kóde taktiež prístupná neprihláseným používateľom. Pri ostrom nasadení systému je možné ľahko znemožniť prístup k registračnému formuláru vymazaním príslušného odkazu zo súboru index.html a obmedzením prístupu k príslušnej ceste na API v súbore security.yaml. Proces autentizácie a autorizácie je znázornený v nasledujúcom diagrame:

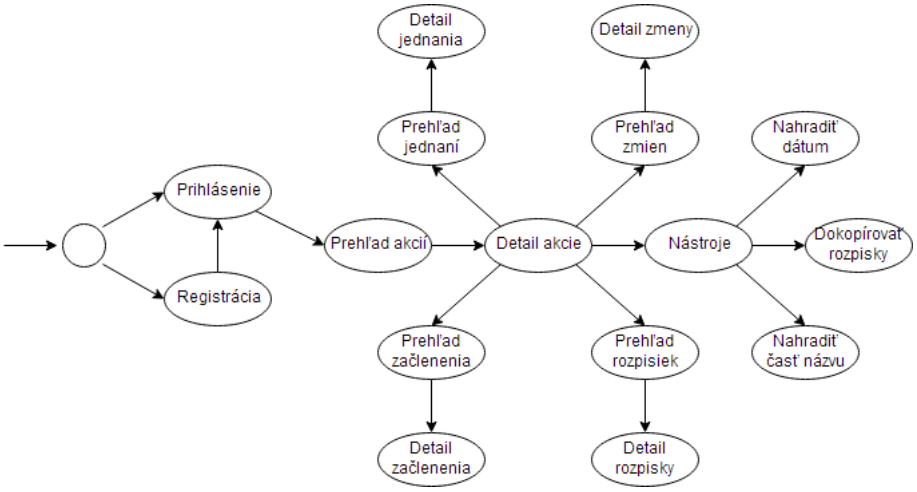


Obr. 4.2: Schéma procesu autentizácie a autorizácie

4.4 Front end

4.4.1 Angular UI-Router

Používateľské rozhranie je na základe zadania implementované ako single-page aplikácia. To bolo dosiahnuté použitím komponentu Angular UI-Router. Jedná sa o komponent, ktorý umožňuje vybudovať aplikáciu ako hierarchický strom stavov, ktoré reprezentujú jednotlivé pohľady (stránky) aplikácie. UI-Router funguje ako konečný stavový automat a umožňuje prechod medzi stavmi na jedinej stránke. [18] Najväčšou výhodou SPA je rýchla odozva po prvotnom načítaní, pretože sa nesťahuje celý HTML kód, ale iba dáta, ktoré sa menia. Okrem dátovej úspory nemusí prehliadač neustále prekresľovať celú stránku, ale len časť, ktorá sa mení. Medzi nevýhody SPA patrí nutnosť povoleného JavaScriptu a dlhší čas prvotného načítania v porovnaní s multiple-page aplikáciami. [19] Jednotlivé pohľady (views) aplikácie a prechody medzi nimi zachytáva nasledujúci diagram:



Obr. 4.3: Stavový diagram pohľadov aplikácie

4.4.2 Bootstrap

Veľká časť používateľského rozhrania aplikácie využíva framework Bootstrap. Jedná sa o kolekciu šablon pre typografiu, formuláre, tlačítka, tabuľky, a ďalšie prvky používateľského rozhrania založenú na HTML a CSS. Bootstrap okrem iného umožňuje vytvorenie responzívneho dizajnu stránky, čo znamená, že prvky stránky sa automaticky prispôsobia rozlíšeniu zariadenia, na ktorom je stránka zobrazená. [25] Implementovaná aplikácia má naozaj vlastnosť responzívneho dizajnu, čo je možné ľahko overiť horizontálnym zmenšovaním okna prehliadača, v ktorom je aplikácia spustená.

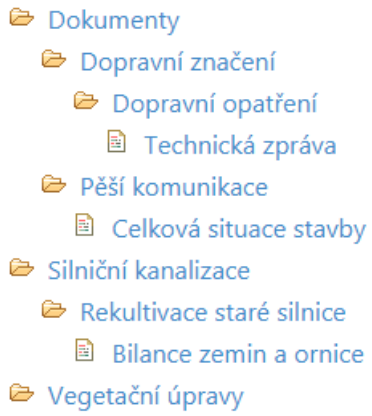
4.4.3 Angular Tree Control

Zobrazenie stromovej štruktúry rozpisiek je v systéme implementované použitím komponentu Angular Tree Control. [27] Najväčšou výhodou tohto komponentu je, že priamo umožňuje pracovať s JSON objektom, ktorý vracia implementovaný kontrolér StructureController. Predpokladajme nasledujúcu JSON štruktúru:

```
{"name": "Root folder", "children": [
  {"name": "Dokumenty", "children": [
    {"name": "Dopravní značení", "children": [
      {"name": "Dopravní opatření", "children": [
        {"name": "Technická zpráva", "discr": "drawing", "children": [
        ]}
      ]}
    ]},
    {"name": "Pěší komunikace", "children": [
      {"name": "Celková situace stavby", "discr": "drawing", "children": [
      ]}
    ]}
  ]},
  {"name": "Silniční kanalizace", "children": [
    {"name": "Rekultivace staré silnice", "children": [
      {"name": "Bilance zemin a ornice", "discr": "drawing", "children": [
      ]}
    ]}
  ]},
  {"name": "Vegetační úpravy", "children": [
  ]}
]}
```

Listing 4.2: Štruktúra rozpisiek akcie

Komponent uvedenú JSON štruktúru vykreslí nasledovne:



Obr. 4.4: Strom vykreslený pomocou Angular Tree Control

Komponent umožňuje nastaviť, ktorý atribút JSON objektu je zobrazený ako text pri ikonke uzlu stromu. Vďaka atribútu objektu „discr“ je možné nastaviť, ktoré uzly tvoria listy stromu a funkciu, ktorá sa po kliknutí na list stromu zavolá. Je tiež možné nastaviť, ktorý atribút predstavuje potomkov uzlu. Prednastavená hodnota je „children“, čo zodpovedá objektu, ktorý vracia kontrolér StructureController. Medzi ďalšie vlastnosti komponentu, ktoré boli v aplikácií využité, patrí nastavenie vzhľadu (na výber je niekoľko tém) a okamžité nastavenie rozbalených uzlov, čo umožňuje implementáciu tlačítka pre okamžité rozbalenie a zbalenie celého stromu.

Záver

Cieľom práce bol návrh a implementácia webového systému pre správu výkresových štítkov založeného na existujúcom systéme nasadenom v podniku. Nový systém mal podporovať všetky prípady použitia pôvodného systému, mali v ňom byť eliminované jeho nedostatky. Cieľ práce bol úspešne naplnený.

Vlastnosti nového systému

Nový systém je implementovaný ako Symfony aplikácia s front endom implementovaným pomocou AngularJS. Back a front end komunikujú prostredníctvom REST API, prenášané dáta majú JSON formát. Aplikácia má MVC architektúru, čo znamená, že dátová vrstva, používateľské rozhranie a programová logika fungujú nezávisle na sebe. Nezávislosť troch komponentov umožňuje paralelný vývoj jednotlivých komponentov bez toho, aby zmeny v jednom komponente ovplyvnili funkčnosť zvyšných dvoch. Dôsledkom modularity systému je aj prehľadnosť a čítateľnosť zdrojového kódu. Ďalšou výhodou architektúry prítomnou v aplikácií je viacero pohľadov na jeden zdroj dát. Príkladom sú rozpisky zobrazené v prehľade rozpisiek ako tabuľka, v detaile akcie sú zobrazené v stromovej štruktúre. Výhodou MVC architektúry a použíteho REST rozhrania je aj možnosť používať back end aplikácie s viacerými front end klientmi, v praxi je možné implementovať a používať napr. mobilný, či desktop klient. V procese analýzy starého systému bol zjednodušený a optimalizovaný dátový model. Rozpisky boli v starom systéme začlenené do stromovej štruktúry s dvoma úrovňami. Nový systém umožňuje vytvoriť stromovú štruktúru s neobmedzenou hlbkou. Najväčšou zmenou oproti starému systému je front end. Ten na základe požiadaviek zadania práce funguje ako single-page aplikácia. Zmenami prešlo používateľské rozhranie aj funkcionalita, nový front end má naviac responzívny dizajn. Tabuľku s prehľadom akcií bolo v starom systéme možné filtrovať buď podľa časti názvu alebo časti čísla akcie, zobrazené akcie bolo možné zhora obmedziť rokom. Nový systém umožňuje filtrovať podľa časti názvu a časti čísla akcie zároveň, navyše pribudlo obmedzenie

rokom zhora aj zdola. Funkcionalita front endu bola rozdelená do pohľadov v závislosti na type entity, ktorej sa funkcionalita týka. Jedná sa o prehľad rozpisiek, prehľad zmien, prehľad začlenenia a prehľad jednaní. Každý prehľad obsahuje tabuľku záznamov entity konkrétneho typu a tlačítka operácií, ktoré sa s príslušným záznamom dajú vykonať. (pridať, zmeniť, odstrániť, zobraziť, exportovať...)

Nedostatky nového systému

Kontroléry pohľadov nového systému obsahujú pomerne veľké množstvo volaní na dátový model back endu, pri rýchlej navigácií v systéme je v niektorých situáciách možno pozorovať oneskorené zobrazovanie dát. Tento jav je zapríčinený asynchrónnou povahou dotazovania v AngularJS a rýchlym načítavaním pohľadov vďaka architektúre SPA. To má za následok, že pohľad front endu je načítaný a zobrazený skôr, než z back endu dorazí odpoveď s požadovanými dátami. Funkcionalita nového systému zodpovedá funkcionalite starého systému s výnimkou exportu dokumentov. V novom systéme je možné exportovať jednu alebo všetky rozpisky do formátu PDF, exportovať štruktúru objektov začlenenia vrátane začlenených rozpisiek a exportovať jednanie ako prostý text. Nový systém neumožňuje export ISO dokumentov. V dodanom starom systéme export týchto dokumentov nefunguje, v procese analýzy navyše nebolo zistené, aký účel tieto dokumenty plnia, ani ako tieto dokumenty zapadajú do agendy firmy. Export obálky a dokumentov jednania v starom systéme prebieha tak, že systém pomocou PHP načíta RTF súbor so šablonou príslušného dokumentu z disku na serveri, nahradí v ňom preddefinované reťazce údajmi z databázy a výsledný súbor ponúkne používateľovi na stiahnutie. V skutočnosti sa teda nejedná o export, ale o parsing a vyplnenie šablony. V novom systéme nebolo možné implementovať načítanie súboru pomocou back endu a následné odoslanie súboru na front end, pretože RTF súbory majú veľkosť rádovo v jednotkách MB, prenos takého súboru cez REST API ako JSON objekt trvá pomerne dlho. Namiesto toho je možné implementovať funkcionalitu vyplnenia šablony – používateľ si vyberie šablonu z lokálneho disku, následná náhrada reťazcov sa uskutoční na úrovni front endu pomocou JavaScriptu. Snaha o implementáciu tejto funkcionality však zlyhala na kódovaní, nahradzované reťazce sa pomocou JavaScriptu nepodarilo zakódovať do znakovej sady Windows-1250 používanej formátom RTF, výsledné dokumenty obsahovali nečítateľný text.

Prínosy nového systému

Systém je implementovaný pomocou súčasných technológií, návrh systému zodpovedá štandardom softvérového inžinierstva. V praxi to umožňuje jednoduchú rozšíriteľnosť systému o ďalšie kontroléry, či pohľady používateľského rozhrania. Systém je implementovaný použitím frameworkov s MIT licenciou, čo znamená, že ani na komerčné použitie nie je nutné platiť poplatky za licenciu. To sa týka aj použitého ORM Doctrine, ktoré je možné použiť s ľubovoľným databázovým systémom, nový systém nie je viazaný na špecifickú databázu. Front end nového systému je navyše možné rozšíriť o komunikáciu s inými API, ktoré poskytujú dáta vo formáte JSON. Nový systém v porovnaní so starým nemá žiadne špecifické požiadavky na prevádzku ani hardvér.

Literatúra

[1] nikic. In: Github [online]. [Citované 1.5.2018.] Dostupné z: https://github.com/nikic/PHP-Parser

[2] TysonAndre. In: Github [online]. [Citované 1.5.2018.] Dostupné z: https://github.com/TysonAndre/php-parser-to-php-ast

[3] glayzzle. In: Github [online]. [Citované 1.5.2018.] Dostupné z: https://github.com/glayzzle/php-parser

[4] Max Silva. In: https://itnext.io/ [online]. [Citované 2.5.2018.] Dostupné z: https://itnext.io/angularjs-exporting-to-pdf-using-pdfmake-js-library-49f3afec97ef

[5] Guy Levin. In: https://dzone.com/ [online]. [Citované 3.5.2018.] Dostupné z: https://dzone.com/refcardz/rest-api-security-1

[6] Tom Johnson. In: https://idratherbewriting.com/ [online]. [Citované 3.5.2018.] Dostupné z: http://idratherbewriting.com/2015/09/04/authorizing-apis/

[7] In: https://jwt.io/ [online]. [Citované 3.5.2018.] Dostupné z: https://jwt.io/introduction/

[8] Eton Digital team. In: https://etondigital.com/ [online]. [Citované 4.5.2018.] Dostupné z: https://www.etondigital.com/popular-symfony-projects/

[9] Sumit Maurya. In: https://mobiloitte.com/ [online]. [Citované 4.5.2018.] Dostupné z: http://www.mobiloitte.com/blog/top-10-apps-websites-developed-angularjs

[10] David Čápka. In: https://itnetwork.cz/ [online]. [Citované 4.5.2018.] Dostupné z: https://www.itnetwork.cz/php/knihovny/php-tutorial-uvod-do-knihoven-a-frameworku

[11] Michał Dziwoki. In: https://gorrion.io/ [online]. [Citované 4.5.2018.] Dostupné z: https://gorrion.io/blog/angularjs-vs-angular/

[12] In: https://doctrine-project.org/ [online]. [Citované 4.5.2018.] Dostupné z: https://www.doctrine-project.org/about/

[13] Matko Đipalo. In: https://matkodjipalo.com/ [online]. [Citované 4.5.2018.] Dostupné z: http://matkodjipalo.com/index.php/2016/08/18/symfony-rest-api-token-authentication/

[14] Jason Watmore. In: https://jasonwatmore.com/ [online]. [Citované 4.5.2018.] Dostupné z: http://jasonwatmore.com/post/2016/04/05/angularjs-jwt-authentication-example-tutorial

[15] In: https://github.com/ [online]. [Citované 4.5.2018.] Dostupné z: https://github.com/lexik/LexikJWTAuthenticationBundle

[16] Drahomír Hanák. In: https://itnetwork.cz/ [online]. [Citované 4.5.2018.] Dostupné z: https://www.itnetwork.cz/nezarazene/stoparuv-pruvodce-rest-api

[17] In: https://github.com/ [online]. [Citované 4.5.2018.] Dostupné z: https://github.com/doctrine/DoctrineFixturesBundle

[18] In: https://github.com/ [online]. [Citované 4.5.2018.] Dostupné z: https://github.com/angular-ui/ui-router

[19] Rafael Carvalho. In: https://scalablepath.com/ [online]. [Citované 4.5.2018.] Dostupné z: https://www.scalablepath.com/blog/single-page-applications/

[20] James Yu. In: https://marketplace.visualstudio.com/ [online]. [Citované 5.5.2018.] Dostupné z: https://marketplace.visualstudio.com/items?itemName=James-Yu.latex-workshop

[21] In: https://mysql.com/ [online]. [Citované 5.5.2018.] Dostupné z: https://www.mysql.com/products/workbench/

[22] Microsoft Corporation In: https://code.visualstudio.com/ [online]. [Citované 5.5.2018.] Dostupné z: https://code.visualstudio.com/

[23] Google LLC In: https://angularjs.org/ [online]. [Citované 5.5.2018.] Dostupné z: https://angularjs.org/

[24] SensioLabs In: https://symfony.com/ [online]. [Citované 5.5.2018.] Dostupné z: https://symfony.com/

[25] In: https://w3schools.com/ [online]. [Citované 5.5.2018.] Dostupné z: https://www.w3schools.com/bootstrap/bootstrap_get_started.asp

[26] In: https://doctrine-project.org/ [online]. [Citované 6.5.2018.] Dostupné z: https://www.doctrine-project.org/projects/doctrine-orm/en/2.6/reference/inheritance-mapping.html#class-table-inheritance

[27] wix. In: https://github.com/ [online]. [Citované 6.5.2018.] Dostupné z: https://github.com/wix/angular-tree-control

Dodatok **A**

Zoznam použitých skratiek

API Application Programming Interface

CRUD Create, Read, Update, Delete

HTML HyperText Markup Language

HTTP Hypertext Transfer Protocol

HTTPS Hypertext Transfer Protocol Secure

IDE Integrated Development Environment

ISO International Organization for Standardization

JSON JavaScript Object Notation

JWT JSON Web Token

MVC Model-View-Controller

ORM Object-Relational Mapping

PDF Portable Document Format

PHP PHP: Hypertext Preprocessor

REST Representational State Transfer

RTF Rich Text Format

SPA Single-Page Application

SQL Structured Query Language

Dodatok B

Návod na použitie

B.0.1 Poznámky

- V systéme musí byť nainštalovaný a spustený MySQL server.
- V systéme musí byť nainštalovaný nástroj composer.
- V systéme musí byť nainštalovaný nástroj yarn.
- K sprevádzkovaniu systému je nutné pripojenie na internet.
- Všetky príkazy sú uvádzané v kontexte adresára src/impl/

B.0.2 Použitie PHP parseru

- cd parser
- composer install
- php Parse.php

Výstup PHP parseru je v adresári parser/out/

B.0.3 Vygenerovanie verejného a súkromného kľúča

- mkdir var/jwt
- openssl genrsa -out var/jwt/private.pem -aes256 4096
- openssl rsa -pubout -in var/jwt/private.pem -out var/jwt/public.pem

Zadané heslo je nutné zadať do súboru config/packages/lexik_jwt_authentication.yaml do kolonky „pass_phrase“. (pass_phrase: 'vase_heslo')

B.0.4 Inštalácia závislostí

- composer install

B.0.5 Vytvorenie databázy

Pred vytvorením databázy je v súbore .env nutné zmeniť riadok „DATABASE_URL=mysql://db_user:db_password@127.0.0.1:3306/db_name“ na „DATABASE_URL=mysql://root@127.0.0.1:3306/rozpisky“.

- php bin/console doctrine:database:create
  (spustiť iba v prípade, že v systéme neexistuje databáza s názvom „rozpisky“)
- php bin/console doctrine:schema:drop –force
  (príkaz vymaže dáta a schému databázy „rozpisky“)
- php bin/console doctrine:schema:create
- php bin/console doctrine:fixtures:load

B.0.6 Nastavenie frontendu

- cd public
- yarn install

B.0.7 Spustenie serveru

- php bin/console server:run

B.0.8 Spustenie aplikácie

Aplikácia sa spúšťa otvorením súboru public/views/index.html vo webovom prehliadači.

Dodatok C

Obsah priloženého CD

/
- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `impl` ..... zdrojové kódy implementácie
  - `thesis` ..... zdrojová forma práce vo formáte LaTeX
- `thesis`
  - `thesis.pdf` ..... text práce vo formáte PDF
  - `assignment.pdf` ..... zadanie práce vo formáte PDF