Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Správa dokumentov v kurzovom systéme

Bakalárska práca

2022
Rastislav Urbanek

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Správa dokumentov v kurzovom systéme

Bakalárska práca

Študijný program:	Aplikovaná Informatika
Študijný odbor:	Informatika
Školiace pracovisko:	Katedra aplikovanej informatiky
Školiteľ:	doc. RNDr. Martin Homola, PhD.
Konzultant:	Mgr. Ján Kľuka, PhD.

Bratislava, 2022
Rastislav Urbanek

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

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

Názov: Document management in a course management system
Správa a dokumentov v kurzovom systéme

Anotácia: Cieľom práce je implementovať modul do kurzového systému courses.maftyz.sk pre tvorbu a správu dokumentov.

Cieľ:
- Rôzne formáty dokumentov
- Systém práv a zdieľanie dokumentov
- História
- Rôzne formy zobrazovania (napr. prezentácia)
- Reprezentácia metadát (témy a vzťahy medzi dokumentami)

Literatúra:
1. Homola, M., Kl'uka, J., Kubincová, Z., Marmanová, P. and Cifra, M., 2019. Timing the Adaptive Learning Process with Events Ontology. In International Conference on Web-Based Learning (pp. 3-14). Springer, Cham.
2. Antoniou, G. and Van Harmelen, F., 2004. A semantic web primer. MIT press.
3. Tarus, J.K., Niu, Z., Mustafa, G., 2018. Knowledge-based recommendation: a review of ontology-based recommender systems for e-learning. Artificial intelligence review 50(1), 21–48

Vedúci: doc. RNDr. Martin Homola, PhD.
Konzultant: Mgr. Ján Kľuka, PhD.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: prof. Ing. Igor Farkaš, Dr.

Dátum zadania: 13.10.2021

Dátum schválenia: 18.10.2021 doc. RNDr. Damas Gruska, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Obsah

Úvod

Na Fakulte matematiky, fyziky a informatiky sa používa na vybraných predmetoch kurzový systém *Courses* vyvíjaný od roku 2013. Prešiel mnohými zmenami, pričom väčšia iterácia bola *Courses 2*. Na jeho vývoji sa podielali nielen učitelia, ale aj študenti.

Postupom času sa ukázalo, že je nevyhnutné rozšíriť jeho využívanie na ďalšie možnosti. Novovytvárajúci *Courses 3* je ešte vo vývoji. Avšak už teraz vieme, aké zmeny sú potrebné k efektívnemu využívaniu tejto verzie v súčasnej dobe.

Pre lepšie porozumenie problematiky *Courses* sme sa zamerali na východiská, ktoré popisujú základné princípy práce s dokumentmi v teoretickej časti bakalárskej práce.

V jej implementačnej časti sa zaoberáme využívaním efektívnych nástrojov s *Courses 2*, pričom pridávame systém „diff“, ktorý nám pomáha sprehľadniť zmeny v histórii dokumentov. Touto verziou chceme taktiež sprehľadniť navigáciu v histórii zmien jednotlivých dokumentov podľa vzoru Google Docs.

Kapitola 1

Teoretické východiská

V každodennom živote sa stretávame s akýmikoľvek druhmi dokumentov. V našom prípade v rámci vzdelávacieho procesu na fakulte. Aj ľudia z iných profesií sa potýkajú s byrokraciou a rigidnosťou pri práci s dokumentmi a preto sa snažíme o uľahčenie, zjednodušenie a zefektívnenie ich používania. Primárnym účelom dokumentov je poskytovanie informácií, pričom my sa budeme zaoberať integráciou podpory pre dokumenty vo webovom prostredí, čo má svoje špecifiká.

V tejto kapitole popisujeme základné princípy práce s dokumentmi, popis vyžadovaných technológií pri implementácií dokumentového modulu a popis prác, ktoré približujú systém do ktorého sa spomínaný modul bude integrovať.

1.1 Manipulácia so súbormi dokumentového typu a ich zobrazovanie

Možnosti zobrazovania dokumentov sú rôzne. Záleží to od formátu aj od toho, či chceme dať možnosť používateľom dokumenty upravovať.

Napr. pri PDF dokumente je možnosť východzieho zobrazenia pomocou prehliadača. Tento prístup je nie najefektívnejší, keďže sa zvyčajne otvára nový tab v okne prehliadača a robí to prácu s dokumentom neprehľadnú. Ak sa neuloží dokument dočasne, tak sa tým pádom zapĺňa úložný priestor v počítači, na čo môže človek zabudnúť.

Prípadne je možné si takýto PDF dokument stiahnuť a zobraziť v PDF čítačke, no nie je to ideálne.

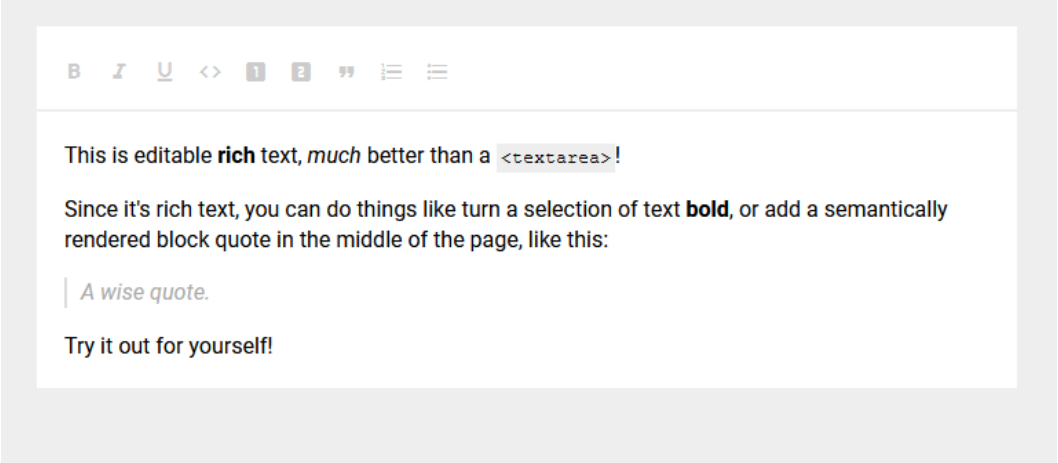
Iný prístup je zobrazovanie takéhoto dokumentu priamo na stránke a teda z nej neodísť. Takto sa vám nestiahne nič do počítača. S tým, že viete dať možnosť si takýto dokument predsa stiahnuť, ak sa to javí používateľovi ako potrebné.

1.1.1 Upravovanie dokumentov

V tejto časti sa pozrieme na rôzne spôsoby úpravy dokumentov.

WYSIWYG editor

Z anglickej skratky *What you see is what you get* (vytlačený dokument verne zodpovedá vizuálnej reprezentácií dokumentu v počítači). Takýto editor má používateľsky prívetivé nástroje k úprave dokumentu. Kliknutím na tlačidlo vieme modifikovať rôzne aspekty dokumentu akými sú napr. tvorba zoznamu alebo nadpisu, odsadenie a iné. Tieto zmeny sa prejavujú v zdrojovom kóde a ak by ho mal používateľ upravovať priamo, bolo by to pomalé, nemusel by vedieť, ktoré príkazy čo robia, o to viac ho môže brzdiť angličtina pokiaľ ju neovláda a mnohé iné úskalia, ktoré sa prekrývajú s nutnou väčšou zručnosťou s informačnými technológiami [12].

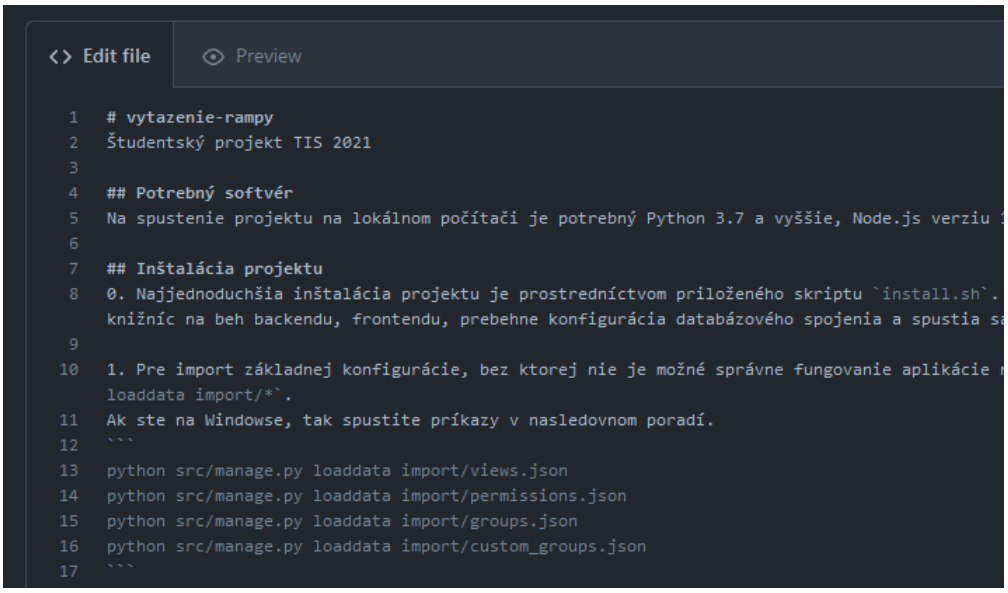


Obr. 1.1: Ukážka editora WYSIWYG

Zdroj: Editor Slate

Editor zdrojového kódu

Obyčajný textový editor predstavuje editor zdrojového kódu. V najštíhlejšej forme je bez akýchkoľvek tlačidiel a jediný nástroj úpravy dokumentu je teda písanie textu. Všetko formátovanie sa deje pomocou špeciálnej syntaxe známej pre daný typ dokumentu (napr. `## Potrebný softvér` pre formát Markdown vyjadruje nadpis 2. úrovne s textom Potrebný softvér). Pripomína editor používaný na programovanie, keďže sa v ňom vyskytuje množstvo špeciálnych znakov nevyužívaných v bežnom texte [11].



Obr. 1.2: Ukážka editora zdrojového kódu

Zdroj: Github source code editor

1.2 Technológie a knižnice

K tvorbe dokumentového modulu využívame rôzne technológie a s nimi sa oboznámime. Taktiež uvedieme knižnice, frameworky umožňujúce naplnenie zadania. Keďže frontend *Courses* je naprogramnovaný v modernom a populárnom frameworku *React*, tak hľadáme optimálne také nástroje, knižnice, ktoré sú v ňom taktiež naprogramované.

1.2.1 REST API

Pre potreby práce si potrebujeme dokumenty ukladať a spätne získavať. Túto funkcionalitu nám sprostredkuje REST API systému *Courses* popísaná v diplomovej práci *Sémantický dátový model pre podporný kurzový systém* [2]. Na začiatok si povieme o rozhraní API, štýle REST a potom popíšeme rozhranie API, ktoré spĺňa požiadavky štýlu REST.

API *(Application Programming Interface)*

Rôzne aplikácie/systémy komunikujú vzájomne medzi sebou prostredníctvom rozhrania API [13]. Ide o softvér poskytujúci sadu metód, prístupových bodov, za účelom sprístupnenia dát softvérovej služby. Výhoda použitia rozhrania je tá, že aplikácia, ktorá ho používa vie byť implementovaná nezávisle od aplikácie na "druhej strane" rozhrania. Iným slovami ak máme aplikáciu, ktorou chceme získavať dáta z Facebooku, tak to docielime tvorením správnych požiadaviek na API Facebooku.

REST *(REpresentational State Transfer)*

Na základe vonkajších požiadaviek bolo vytvorenie štandardov do sveta WWW (angl. World Wide Web) nevyhnutnosťou. V 90. rokoch bol internet „voľnejší“ v zmysle absencie bezpečnostných protokolov a iných štandardov, čo ale nevyhovovalo biznisovému sektoru, ktorý začal chápať potenciál WWW a chcel si túto platformu upraviť pre vlastné biznisové potreby. Bolo teda nutné štandardizovať, nastavovať zrozumiteľné mantinely. Z tohto dôvodu programátor Roy Fielding, ktorý pracoval aj na protokole HTTP, navrhol softvérový architektonický štýl **REST** [1]. Tento štýl prináša obmedzenia pre sieťovú klient-server komunikáciu.

Vníma serverové entity ako objekty, ktoré je možné čítať, vytvárať, meniť a mazať. Tieto operácie sú mapované na metódy protokolu sprostredkujúceho komunikáciu Najpoužívanejší je pre tento účel HTTP protokol.

Hovorí o správaní architektúry ale nehovorí, ako dané správanie implementovať, ani aký komunikačný protokol využívať.

Pre túto prácu sú zo štýlu REST relevantné nasledovné pojmy:

- zdroj *(angl. resource)* – abstraktný pojem mapovateľný na doménovú entitu alebo entity. Vieme ho pomenovať slovom, v našom prípade napr. študent(i), kurz(y) alebo dokument(y),
- identifikátor zdroja *(angl. resource identifier)* – umožňuje zaslaním požiadavky vykonávať operáciu nad zdrojom, ktorý tento identifikátor označuje. Používa sa aj termín koncový bod zdroja *(angl. resource endpoint)*. Identifikátorom je URL,
- reprezentácia zdroja *(angl. resource representation)* – zachytenie stavu zdroja pomocou formátov ako JSON, XML, atď. Takáto reprezentácia je posielaná v telách požiadaviek a odpovedí.

REST API

Rozhranie API, ktoré spľňa požiadavky štýlu REST sa budeme zaoberať v tejto časti. V princípe REST API po obdržaní požiadavky *zdroja* s požadovaným *identifikátorom* posiela kontextovo-nezávislý stav tohoto *zdroja* v želanej *reprezentácií*. Príklad identifikátora zdroja: `http://hostname/api/:collection/:resourceID`. Kde `collection` a `resourceID` sú parametrami.

Ideálny protokol umožňujúci komunikáciu s REST API je HTTP. Operácie nad zdrojmi serveru sa môžu diať nasledovnými HTTP metódami: [7].

- `GET` – získame želanú reprezentáciu zdroja,
- `POST` – vytvoríme nový zdroj podľa priložených dát v požiadavke,

- `PUT` – aktualizujeme zdroj podľa priložených dát v požiadavke,
- `PATCH` – *čiastočne* aktualizujeme zdroj podľa priložených dát v požiadavke,
- `DELETE` – zmažeme daný zdroj.

1.2.2 PDF.js a React-pdf

Pre potreby systému *Courses* chceme možnosť zobrazenia PDF dokumentu priamo na stránke, teda aby ju nemusel používateľ sťahovať resp. otvárať v novom okne, kde sa mu otvorí v čítačke prehliadača. Pre tento účel nám pomôže *PDF.js* - open-source JavaScript knižnica pre tvorbu štandardizovaných čítačiek PDF dokumentov. Je moderná, priamo integrovaná v HTML5 štandarde a udržiavaná veľkou komunitou. V našom prípade použijeme knižnicu *React-PDF*, čo je komponent využívajúci *PDF.js* a napísaný vo frameworku *React* [9, 10].

1.2.3 Axios

Komunikáciu so serverom sprostredkúva Axios, čo je HTTP klient, ktorý dáva možnosť posielať požiadavky na server a prijímať jeho odpovede pomocou HTTP protokolu [8].

Jedna z vymožeností Axiosu je možnosť zo strany prehliadača tvoriť XMLHTTP požiadavky. JavaScriptové objekty známe z AJAX, ktoré umožňujú dynamicky upravovať stránku bez nutnosti plného obnovenia stránky (angl. full page refresh). Taktiež zo strany servera vie posielať štandardné HTTP požiadavky pomocou Node.js.

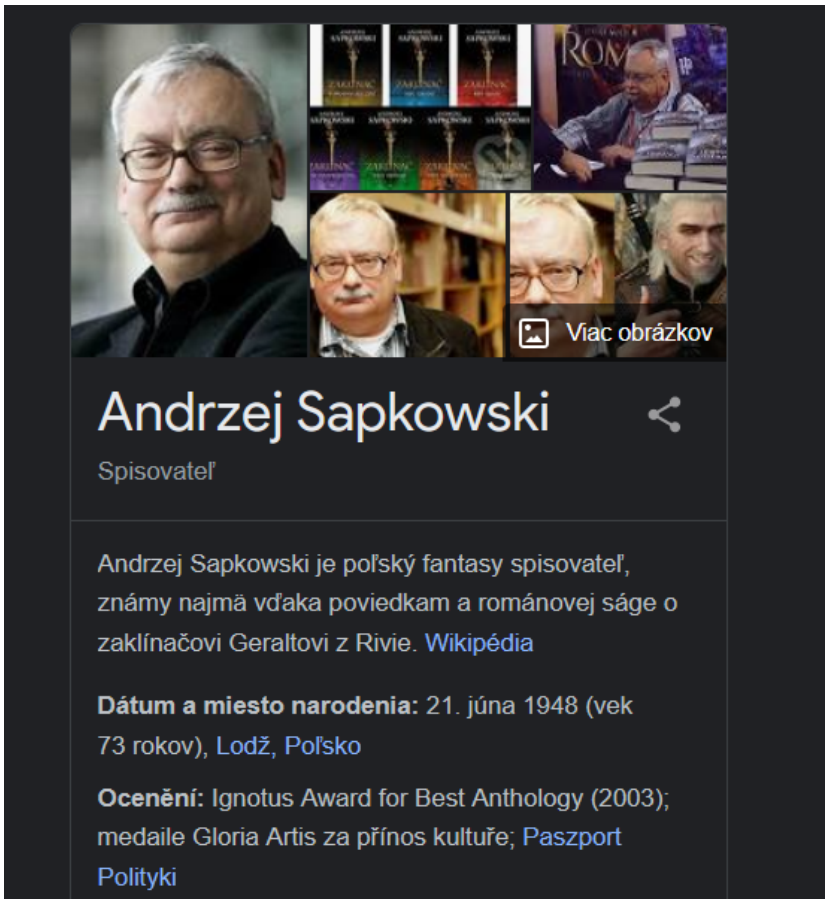
Axios použijeme pri komunikácií so serverom systému *Courses* pomocou HTTP protokolu.

1.3 Sémantické dáta

Výplod iniciatívy *Semantic Web* je možnosť pre stroje chápať a vykonávať logiku nad dátami pomocou sémantiky [6]. Primárnym cieľom *Semantic Web* je zefektívnenie vyhľadávačov, aby generovali výsledky, čo najbližšie k požadovanému dopytu.

Pomocou ontológií a ich metadát sa priniesol do sveta WWW sémantický kontext a výsledok môžeme vidieť napr. vo vymoženosti vyhľadávača Google generovať pri dopytovaní slávnej osobnosti vedomostný panel (ukážka 1.3) so základným informáciami o danej osobnosti a krátkym popisom o nej.

Systém *Courses* používa sémantické dáta taktiež.



Obr. 1.3: Ukážka vedomostného panel Googlu

Zdroj: Google

1.3.1 Ontológia

Formálny popis nejakej domény nazývame *ontológia* (používa sa aj slovo slovník). Ontológia popisuje vzťahy entít v doméne, ich vlastnosti a klasifikuje tieto entity do tried, ktoré spolu tvoria hierarchiu.

Ontológia *Courses* je popísana jazykom OWL 2 (angl. Web Ontology Language).

1.3.2 RDF

Dáta o *Courses* sú uložené vo formáte RDF (angl. Resource Description Framework). Jedná sa o štandardný formát sémanticky prepojených dát. Dátová štruktúra predstavuje orientovaný graf. Umožňuje definovať čokoľvek, kýmkoľvek a aby takéto dáta boli konzistentné zgrupujú sa pod organizované ontológie bez redundantných definícií (napr. Linked Open Vocabularies). Veľkou výhodou je využitie takýchto existujúcich ontológií, ktoré už niektoré koncepty definovali za nás.

Formát RDF dát predstavuje tzv. trojice (angl. triples) nasledovne:

```
<subject> <predicate> <object> .
```

V grafovej databáze typu RDF je `<predicate>` (predikát alebo prísudok) reprezentovaný ako hrana medzi 2 vrcholmi. Ďalej `<subject>` a `<object>` (subjekt a objekt alebo podmet a predmet) reprezentujú vrcholy. Vrcholy môžu nadobúdať jednu z hodnôt:

- IRI (angl. Internationalized Resource Identifier) – jednoznačne identifikuje zdroj. Príklad `http://example/ontology#segedin`,
- literál – dátový typ, v textovom reťazci.

Hodnoty `<subject>` a `<predicate>` sú vždy IRI. Hodnotou `<object>` je IRI alebo literál.

Príklad RDF dát:

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix foaf:  <http://xmlns.com/foaf/0.1/#> .
@prefix sk:  <http://example.com/data/#> .

sk:rasto rdf:type foaf:Person .
sk:rasto foaf:nickname "ZubatyNos" .
```

V príklade vidíme vrchol `sk:rasto`, ktorého vlastnosť `rdf:type` má hodnotu `foaf:Person` (definovaná v ontológií *foaf*). Tento vrchol má ďalej vlastnosť `foaf:nickname` s hodnotou `"ZubatyNos"` (spomínaný literál). Taktiež v príklade vidíme tzv. prefixy, ktoré nám hovoria akú ontológiu sa chystáme použiť a taktiež aký prefix pre rôzne IRI z tejto ontológie budeme používať. Prefix skracuje IRI napr. takto `http://example.com/data#rasto` na `sk:rasto`. Ontológia *rdf* nám ponúka metadáta o vrchole `sk:rasto`.

1.3.3 SPARQL

RDF dáta sa štandardne dopytujú jazykom SPARQL (angl. SPARQL Protocol and RDF Query Language). Syntaxovo je veľmi podobný jazyku SQL. Backend *Courses* generuje SPARQL dopyty pri požiadavkách na jeho REST API.

Príklad SPARQL dopytu:

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix foaf:  <http://xmlns.com/foaf/0.1#> .
@prefix sk:  <http://example.com/data#> .

SELECT ?x WHERE
{
    ?x rdf:type foaf:Person .
}
```

`?x` je v tomto prípade premenná reprezentujúca vrchol, ktorého vlastnosť `rdf:type` má hodnotu `foaf:Person`. Ak by sme spustili tento dopyt na databáze s obsahom ako v ukážke 1.3.2, tak vrcholom `?x` by bol `sk:rasto`

1.4 Relevantné práce

1.4.1 REST API Courses

Pre možnosť CRUD operácií nad dokumentmi je potrebné popísať štruktúru dát dokumentov na backende systému *Courses* a manuálne spustiť SPARQL príkazy na databázovom koncovom bode, čo vidíme v práci *Sémantický dátový model pre podporný kurzový systém* [2].

Komunikácia s rozhraním sa deje posielaním HTTP požiadaviek a spracovaním HTTP odpovedí, k čomu nám poslúži Axios (1.2.3). Frontend *Courses* už má popísane JavaScriptové funkcie, ktoré využívajú Axios a komunikujú s REST API backendu *Courses*. Pre nás je relevantná `axiosRequest` v súbore `src/helperFunctions.js`, ktorá prijíma ako argumenty:

- jednu z HTTP metód `GET, POST, DELETE, PATCH a PUT`,
- URL v tvare `http://www.courses.matfyz.sk/data/*`,
- dáta prislúchajúce k HTTP metóde.

1.4.2 Dokumentový modul systému Courses 2

Efektívnosť práce s dokumentmi v *Courses 2* je popisované v *Electronic Notes* [3]. Funguje viacmenej správne, ale jeho využívanie v praxi poukázalo, že je nutné jeho možnosti rozšíriť, čo rozoberieme v sekcií 2.1.

Dokumenty v *Courses 2* sa môžu do seba vnárať, čím tvoria hierarchické usporiadanie. Pre vytvorenie jedného dokumentu sa musí vytvoriť koreňový dokument a následne do neho vnoriť dokument, ktorý sme chceli vytvoriť. Ďalej pri vytváraní dokumentu

zadávame tzv. *Navigation Index*, čo je číslo, ktoré reprezentuje poradie, akým sa dokument zobrazuje v bočnom kontextovom paneli kurzu. Ak sa rozhodneme vytvorený dokument zmazať, tak sa nám to síce podarí, ale s tým, že sa bude stále zobrazovať ako zmazaný. Pridávanie dokumentov pdf súboru nie je možné. Dá sa to obísť nahratím takýchto súborov inde a ich nalinkovaním.

Ďalej modul umožňuje vidieť históriu zmien dokumentu s možnosťou obnovenia historickej verzie, čím sa z nej stala verzia aktuálna, viditeľná študentmi.

1.5 Kurzový systém Courses

LMS (angl. Learning Management System) systémy sú v princípe navrhnuté správne, plnia svoj účel. Dostupné nástroje týchto systémov nie sú dostatočné flexibilné na naplnenie špecifických požiadaviek, resp. ich úprava nie je možná. Ak tieto požiadavky chceme naplniť, musíme prísť s novým na mieru ušitým riešením.

Preto vznikol v roku 2013 systém *Courses*, fakultný koncepčný derivát LMS systému [4]. Jednou z mnohých požiadaviek bolo napr. možnosť rozsiahlej konfigurovateľnosti kurzov, možnosť integrácie s *blog.matfyz.sk* a modulárna architektúra pre zjednodušenie obohatenia systému v budúcnosti.

Courses prešiel od jeho vzniku mnohými iteráciami po zapracovaní pripomienok študentov a učiteľov. Najnovšia iterácia *Sémantický dátový model pre podporný kurzový systém* [2] sa zatiaľ nepoužíva, pretože je potrebné dopracovanie zásadných modulov ako aj dokumentového modulu, ktorým sa zaoberáme v tejto bakalárskej práci.

1.5.1 Adaptívny vyučovací proces

Jeden z primárnych cieľov pre *Courses* je pozmeniť tradičný vyučovací proces s LMS systémom tak, že sa bude adaptovať potrebám študenta. V publikovanom článku *Timing the Adaptive Learning Process with Events Ontology* je rozobratý systém *Courses* a iný prístup k adaptívnemu vyučovaniu hlavne vo funkcionalite recenzovania kódu medzi študentmi (tzv. peer reviews) [5]. Ďalej v zgrupovaní všetkých potrebných materiálov a všetkých dôležitých deadlinov do logických celkov (blokov), čím sa vytvorí časovo ohraničený kontext kurzu, v ktorom sa vie študent zorientovať. Nakoniec vytvoríme priestor na prepájanie vyučovacích materiálov cez metadáta tak, aby bolo možné vyhodnotiť, ktoré materiály si má používateľ precvičiť pred skúškou, resp. ktoré sa má po midterme doučiť.

1.5.2 Popis ontológie Courses 3

Základnou jednotkou sú kurzy (trieda `Course`). Kurzy obsahujú viacero inštancií (trieda `CourseInstance`) – viacero priebehov kurzu v rozdielnych akademických rokoch. Inštancii kurzu sú priradení registrovaní používatelia (trieda `User`) schválení učiteľom. Každý dokument (trieda `Document`) vždy prislúcha inštancii kurzu.

Ďalej sú definované časovo ohraničené udalosti (trieda `Event`) a medzi ne patria cvičenia a prednášky (podtriedy `Lab` a `Lecture` triedy `Session`), ale taktiež tu patria aj inštancie kurzu. Logické členenie inštancie kurzu na niečo, ako týždne je dosiahnuté pomocou triedy `Block`.

Umožňuje tvorbu zadaní, skúšok, tímov pre kolaboratívne úlohy a mnohé iné.

1.5.3 História systému

Počiatok systému datuje bakalárska práca *Integrovaný LMS systém* [4] z roku 2013. Vtedy systém stál na PHP frameworku CodeIgniter a bol navrhnutý podľa architektonického vzoru HMVC (modulárny, hierarchizovaný vzor MVC). Vzor MVC hovorí o členení kódu a to nasledovne: *Model* slúži na komunikáciu s databázovou vrstvou, *View*-u sa posúvajú dáta z *Model*-u a sú zobrazované používateľovi. *Controller* riadi komunikáciu medzi týmito dvoma celkami.

Medzičasom jazyk PHP začal strácať na popularite, zatiaľ čo jazyk JavaScript dostal mnohé vylepšenia (napr. ES6) a začal sa vo väčšej miere používať, s čím príbudlo aj enormné množstvo voľne dostupných knižníc. Ako to pri vývoji softvéru býva, veci, ktoré sa na počiatku zdali dobre zabehnuté vyžadovali rozšírenie použitia. Konkrétne sa jedná o to, že si moduly definovali vlastný dátový model, vlastné štýlovanie a vlastnú aplikačnú logiku.

Implementácia Courses 3 stavia na tomto systéme. Hlavné zmeny v Courses 3 sú:

- zmena databázového modelu z relačného na sémantický,
- upustenie od jazyka PHP a frameworku CodeIgniter,
- použitie JavaScriptu a moderného frameworku React.

1.5.4 Implementácia systému

1.5.5 Courses 3 ontológia - Dokumenty

V rámci ontológie *Courses 3* využívame päť dokumentových tried: `Document`, `InternalDocument`, `ExternalDocument`, `File`, `Material`. Všetky sú podtriedou triedy Document okrem triedy Document samotnej.

Internal Dokumenty sú dokumenty, ktoré chceme tvoriť na stránke pomocou wysiwyg alebo source code editoru. Sú formátu Markdown, HTML, alebo Textile. External dokument je dokument, ktorého obsah nie je uložený v našej databáze, ale ukladáme si jeho URI. File je akýkoľvek iný súbor, ktorý nie je formátu Markdown, HTML, alebo Textile. Jediný spôsob ako ho upraviť je upraviť ho interne a nahrať zmenenú verziu. Materiál je taký dokument, ktorý sa používa ako výučbový materiál, viditeľný pre študentov.

Tieto termíny budeme ďalej používať v práci. Je dôležité podoktnúť, že každý dokument patrí nejakému existujúcemu kurzu.

Kapitola 2

Implementácia

2.1 Špecifikácia dokumentového modulu Courses 3

Literatúra

[1] FIELDING, R. 2000. *Architectural styles and the design of network-based software architectures* : doctoral dissertation. IRVINE : UNIVERSITY OF CALIFORNIA, 2000.

[2] CIFRA, M. 2018. *Sémantický dátový model pre podporný kurzový systém* : diplomová práca. Bratislava : FMFI UK, 2018.

[3] BILISICS, A. 2014. *Electronic Notes* : bakalárska práca. Bratislava : FMFI UK, 2014.

[4] ČULÍK, J. 2013. *Integrovaný LMS systém* : bakalárska práca, Bratislava : FMFI UK, 2013.

[5] Homola, M., Kl'uka, J., Kubincová, Z., Marmanová, P. and Cifra, M., 2019. Timing the Adaptive Learning Process with Events Ontology. In International Conference on Web-Based Learning (pp. 3-14). Springer, Cham.

[6] Antoniou, G. and Van Harmelen, F., 2004. A semantic web primer. MIT press.

[7] Mozilla, 2021. *Dokumentácia webových technológií pre webových developerov* [online] Mozilla, 2021 dostupné na internete: `https://developer.mozilla.org/en-us/docs` [cit. 2022-01-15].

[8] Axios, 2021. *Dokumentácia http klienta Axios* [online] The Axios Project, 2021 dostupné na internete: `https://axios-http.com/docs/intro` [cit. 2021-12-29].

[9] PDF.js, 2021. *Knižnica PDF.js* [online] Mozilla Foundation, 2021 dostupné na internete: `https://github.com/mozilla/pdf.js/` [cit. 2021-12-28].

[10] React-PDF, 2021. *Knižnica React-pdf* [online] Wojciech, M. 2021 dostupné na internete: `https://github.com/wojtekmaj/react-pdf` [cit. 2021-12-28].

[11] *Moderné vzdelávanie pre vedomostnú spoločnosť/Projekt je spolufinancovaný zo zdrojov EÚ* [online]. Dostupné na internete: `https://cloud7.edupage.org/cloud/Prirucka_Word.pdf?z%3Akzx811n7H09JHxNhMszY%2BBQo7nb4kD2NvSm9mkAh2%2Fd9a%2FjAgXo%2FSDOuej8L3giC` [cit. 2022-01-21].

[12] Rhee, Kyung-Hyune, and Jeong Hyun Yi, eds. *Information Security Applications: 15th International Workshop, WISA 2014, Jeju Island, Korea, August 25-27, 2014. Revised Selected Papers.* Vol. 8909. Springer, 2015. s. 124, ISBN 978-3-030-17982-3. [cit. 2022-01-20].

[13] Matthias Biehl, *API Architecture*, API-University Press, 2015, s. 15-16 ISBN 9781508676645. [cit. 2022-01-22].