FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Analýza a implementácia školského systému domácich prác
Student:	Tibor Engler
Vedoucí:	Ing. Miroslav Prágl, MBA
Studijní program:	Informatika
Studijní obor:	Bezpečnost a informační technologie
Katedra:	Katedra počítačových systémů
Platnost zadání:	Do konce letního semestru 2018/19

Pokyny pro vypracování

- Vykonajte analýzu súčasného súborového úložiska a jeho implementácie na Gymnáziu, Poštová 9, Košice, pre účely odovzdávania domácich prác.
- Porovnajte aktuálne dostupné možnosti lokálneho a cloudového úložiska, vhodného pre tento účel, vrátane už hotových riešení poskytovateľov takýchto služieb.
- Na základe porovnania implementujte najvhodnejšie riešenie, navrhnite vlastné riešenie, alebo skombinujte existujúce riešenie s vlastnými úpravami. Preferujte využitie cloudových technológií.

Seznam odborné literatury

Dodá vedoucí práce.

prof. Ing. Róbert Lórencz, CSc.
vedoucí katedry

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

V Praze dne 18. prosince 2017

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalárska práca

Analýza a implementácia školského systému domácich prác

Tibor Engler

Katedra počítačových systémů
Vedúci práce: Ing. Miroslav Prágl, MBA

9. mája 2018

Poďakovanie

V prvom rade by som chcel poďakovať vedúcemu práce, pánu Ing. Miroslavovi Práglovi, MBA za trpezlivosť a ochotný prístup pri vedení tejto práce. Zároveň by som chcel poďakovať Ing. Petrovi Javnickému za technickú podporu a tak isto RNDr. Monike Molokáčovej z Gymnázia, Poštová 9, Košice za ochotný prístup pri testovaní noviniek vo vyučovacom procese. V neposlednom rade ďakujem svojej rodine a priateľke Michaele za nepoľavujúcu trpezlivosť a podporu.

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 9. mája 2018 .....

Odkaz na túto prácu

Engler, Tibor. *Analýza a implementácia školského systému domácich prác.* Bakalárska práca. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2018.

Abstrakt

Táto bakalárska práca sa zaoberá analýzou požiadaviek základných a stredných škôl na webový systém odovzdávania domácich úloh, prieskumom existujúcich riešení tejto problematiky a vývojom prototypu za použitia cloudových technológií. Účelom vývoja prototypovej aplikácie je oboznámenie čitateľa s vývojom pre cloud a s využívaním cloudových prostriedkov v danej problematike.

Kľúčové slová cloudové úložisko, Microsoft Graph, Office 365, ASP.NET, MVC

Abstract

This bachelor thesis deals with the analysis of the requirements of primary and secondary schools on the web system for submission and collection of home assignments, the research of existing solutions and the development of a prototype using cloud technologies. The purpose of prototype application development is to familiarize the reader with cloud development and cloud usage in the given area.

Keywords cloud storage, Microsoft Graph, Office 365, ASP.NET, MVC

Obsah

Zoznam obrázkov

Zoznam ukážok kódu

Zoznam tabuliek

Úvod

Počítačové technológie sa v poslednom desaťročí prepracovali z veľkých korporátnych firiem do každého aspektu nášho života. Ich využívanie nám pomáha automatizovať určité typy úloh a vďaka nim nám ostáva viac času na dôležitejšie veci, ktoré počítač nezvládne lepšie ako my.

Skvelým príkladom pre integráciu počítačových systémov je bezpochyby školstvo, v ktorom sa uplatnia ako programy s edukačným charakterom, tak aj software, ktorý zjednoduší prácu učiteľov skutočne len na učenie a oberie ich o starosti so zbytočnou byrokraciou, s ktorou sa toľké roky museli trápiť. Preto je veľmi dôležité, aby bol takýto software vysoko intuitívny a aby učitelia dosiahli svoj požadovaný cieľ pri minimálnom úsilí.

Cieľom tejto bakalárskej práce je implementovať aplikáciu pre odovzdávanie domácich úloh formou uploadu súborov do jednotného spravovateľného úložiska. Nakoľko je počítačové vybavenie v základných a stredných školách dlhodobo zanedbávané a školy nemajú dostatok prostriedkov na jeho inováciu, aplikácia bude používať pre tento účel cloudové úložisko.

Hlavnou výhodou používania takéhoto systému je organizované odovzdávanie domácich úloh, nepoužívanie prenosných médií, potenciálne nakazených vírusmi a samozrejme možnosť širšieho nasadenia.

V úvodných kapitolách si povieme niečo o aktuálnej situácii v tejto problematike na našej testovacej škole[1], ktorá však výborne zastupuje situáciu vo väčšine škôl na Slovensku. Zanalyzujeme taktiež dostupné riešenia automatizovaného odovzdávania resp. zberu domácich úloh, ktoré sú už momentálne na trhu. Následne si ukážeme na testovacej aplikácii, na akom princípe takýto software funguje, ako komunikuje s cloudom a ako môžeme pri takomto softwarei využiť už existujúce informačné systémy školy. Na záver jedno z riešení implementujeme do produkčného prostredia a zistíme, či skutočne spĺňa parametre aplikácie, ktorá by sa mohla v školstve využívať dennodenne.

[1]Testovacou školou je pre tento účel Gymnázium, Poštová 9, Košice

Kapitola 1

Cieľ práce

- Analyzovať súčasné súborové úložisko a jeho implementáciu na Gymnáziu, Poštová 9, Košice pre účely odovzdávania domácich prác.
- Porovnať aktuálne dostupné možnosti lokálneho a cloudového úložiska, vhodného pre tento účel, vrátane už hotových riešení poskytovateľov takýchto služieb.
- Vytvoriť ukážkovú aplikáciu a na základe porovnania s existujúcimi riešeniami implementovať najvhodnejšie riešenie s preferenciou cloudových technológií.

Kapitola 2

Analýza

2.1 Analýza východiskového stavu

Gymnázium, Poštová 9, Košice je jedna z najúspešnejších stredných škôl na Slovensku. Každý rok ju navštevuje približne 560 študentov a cca 40 pedagógov. Nakoľko je táto škola zameraná na rozšírené vyučovanie matematiky, je samozrejmé, že sa sem informatizácia výučby dostala oveľa skôr, ako do iných škôl. Za posledné 3 roky pokročila do štádia, v ktorom je klasická triedna kniha a klasifikačný hárok nahradený webovou, resp. mobilnou aplikáciou.

Školská informačná sieť prebehla v poslednom roku značnou renováciou a momentálne sa skladá z racku s virtuálnym serverom Windows Server 2016, ktorý funguje v roli `Active Directory` a `Radius` serveru. Ďalej sa v nej nachádza linuxový server Centos 7, ktorý zastáva rolu `Apache`, takže na ňom beží oficiálna webstránka školy. Ako záložný server pre službu Active Directory slúži bývalý AD server Windows 2008 R2, na ktorý sa pravidelne replikuje globálny katalóg užívateľov a rovnako aj ostatné služby nutné pre chod Windows-based siete. Celú sieť riadi router značky MikroTik, ktorý segmentuje sieť podľa učební (pomocou VLAN), resp. ďalších kritérií. Na tento router je pripojených 6 switchov, do ktorých sú pripojené jednotlivé počítače, resp. wifi access pointy. Všetky sieťové prvky sú značky MikroTik.

Všetci užívatelia (žiaci, učitelia aj nepedagogický personál) majú svoje doménové prihlasovacie meno a heslo, pomocou ktorých sa prihlasujú do školských počítačov a notebookov a zároveň tieto údaje využívajú aj na prihlasovanie sa do školskej wifi siete.

Pre školské zariadenia sa zároveň podľa členstva užívateľa v skupine „Ziaci“ resp. „Zamestnanci“ aplikujú aj korešpondujúce skupinové politiky, ktoré z bezpečnostných dôvodov obmedzujú funkcionalitu počítačov na minimum nutné pre výučbu. V rámci týchto politík sa užívateľom mapujú nasledovné sieťové jednotky:

Jednotka H Disk Home, na ktorý si užívateľ môže ukladať svoje osobné súbory

Jednotka M Disk Share, z ktorého môžu žiaci iba čítať a učitelia môžu čítať aj zapisovať. Slúži pre zdieľanie učebných materiálov.

Jednotka S Disk Write, na ktorý môžu žiaci iba zapisovať (nemôžu čítať) a učitelia nad ním majú plnú kontrolu. Tento disk slúži pre odovzdávanie vypracovaní písomiek písaných na počítačoch.

V pôvodnej (nezrenovovanej) sieti boli tieto disky prístupné mimo školy len cez rozhranie WinSCP (čiže pomocou protokolu SSH), čo predstavovalo pomerne veľké bezpečnostné riziko, keďže ako koncový bod pre tento protokol slúžil server CygWin, nainštalovaný na pôvodnom Windows 2008 R2 serveri. Nesprávne nakonfigurovaný koncový bod mohol týmto spôsobom obchádzať windowsové práva ACL a užívatelia sa tak vedeli dostať aj k súborom, ku ktorým by nemali mať za bežných okolností prístup. Z tohto dôvodu bol prístup pomocou WinSCP zrušený a nahradený webovým rozhraním Pydio, ktoré pristupovalo ku súborom pomocou protokolu SMB.

Po renovácii siete[2] vyvstala otázka, čo urobiť so sieťovými diskmi, ktoré sa pôvodne nachádzali fyzicky nainštalované v starom Windows serveri. Jednak sa ich voľná kapacita rapídne zmenšovala a zároveň bolo nutné myslieť na to, že tieto staré disky, nezaradené v raide, mohli hocikedy zlyhať. Na túto otázku existovali len dve odpovede: Buď škola za nemalé prostriedky nakúpi dostatočné množstvo diskov a vytvorí si tak vlastné, redundandné diskové pole, alebo využije svoje optické pripojenie do akademickej siete a presunie svoje súborové úložisko do cloudu.

Odpoveď sa ponúkla o niekoľko mesiacov sama, keď spoločnosť Microsoft Slovakia uzavrela s ministerstvom školstva multilicenčnú dohodu pre všetky základné a stredné školy, ktorou poskytuje všetky verzie OS Windows a kancelárskych balíkov Office pre žiakov a učiteľov zdarma. K tomu začal Microsoft medzinárodne zdarma ponúkať všetkým vzdelávacím inštitúciám Office 365 pod licenciu A1. Toto zahŕňa aj jednaterabajtové osobné úložisko pre každého užívateľa s licenciou. Preto majú v súčasnosti školy na Slovensku tendencie využívať cloudové úložisko od Microsoftu s oveľa väčšou pravdepodobnosťou, než od akéhokoľvek iného poskytovateľa. Keďže je však povedomie o cloudových službách v regionálnom školstve mizivé, potrvá ešte istý čas, kým sa na takýto typ služieb preorientuje väčšina škôl.

2.2 Zadania a vypracovania

V tejto stati si popíšeme všetky aspekty a parametre zadaní a vypracovaní, ktoré sa v bežnej škole odohrávajú. Tento popis nám pomôže pochopiť, akým

[2]Renovácia zahŕňala výmenu sieťových prvkov a nasadenie nového Windows 2016 serveru

spôsobom tento proces v školstve funguje. Zadania a vypracovania tak môžeme podľa niekoľkých kritérií kategorizovať do nasledujúcich kategórií:

Rozdelenie podľa dĺžky trvania

krátkodobé – najčastejšie písomky, v ktorých sa vyžaduje odovzdanie v krátkom časovom limite,

strednodobé – najčastejšie štandardné domáce úlohy, ktoré sa odovzdávajú na najbližšej hodine v danom týždni, resp. v nasledujúcom týždni,

dlhodobé – najčastejšie referáty, u ktorých je nastavený deadline v predstihu.

Rozdelenie podľa počtu účastníkov

samostatné – najčastejšie písomky, štandardné domáce úlohy,

skupinové – najčastejšie referáty, projekty a skupinové práce.

Rozdelenie podľa formátu odovzdania

papier – najčastejšie vypracovanie písomky formou výpočtov, resp. odpovedí na otázky

formulár – najčastejšie vypracovanie písomky formou krátkych odpovedí na otázky kratšieho rozsahu

poster – najčastejšie vypracovanie referátu alebo projektu formou plagátu

prezentácia – najčastejšie vypracovanie referátu alebo projektu formou počítačovej prezentácie

aplikačné súbory – najčastejšie vypracovanie domácej úlohy z informatických predmetov, v ktorých je požadovaný zdrojový kód a/alebo skompilovaný program

2.2.1 Scenár zadania a odovzdania úlohy

Napriek tomu, že sa môže zdať spracovanie scenáru zadania a odovzdania úlohy ako banalita, sú v ňom určité body, ktoré je pred automatizáciou takéhoto scenára dôležité spomenúť. Práve preto, že sa nám zdajú samozrejmé, na ne môžeme pri implementácii poľahky zabudnúť.

Učiteľ

1. Učiteľ vymýšľa zadanie. Zadanie nesie minimálne nasledujúce atribúty: text zadania, dátum a čas odovzdania, spôsob hodnotenia.
2. Učiteľ dané zadanie zadá jednotlivcovi, študijnej skupine, triede alebo niekoľkým triedam.
3. Učiteľ čaká do termínu odovzdania (deadline) a vyzbiera vypracovania od žiakov alebo deadline predĺží.
4. Učiteľ jednotlivé odovzdania hodnotí.
5. Učiteľ oznamuje žiakom ich výsledky.

Žiak

1. Žiak dostane od učiteľa zadanie.
2. Žiak vypracuje zadanie – vzniká vypracovanie.
3. Žiak svoje vypracovanie odovzdá buď fyzicky alebo elektronicky, prípadne požiada o predĺženie deadlineu.
4. Žiak čaká na vyhodnotenie.

Akým spôsobom by bolo možné preniesť čo najväčšie množstvo týchto úkonov do elektronickej podoby? V prvom rade si to vyžaduje kompaktný a jednoducho ovládateľný software, ktorý by pokryl čo najširšie spektrum kategórií, ktoré sme vyššie spomenuli, tj. aby bol čo najuniverzálnejší a zároveň aby jeho funkcionality pokryli čo najviac (optimálne všetko) z oboch scenárov.

2.3 Papier vs. Počítač

Cieľom nasadenia elektronického systému odovzdaní nemusí byť len automatizácia a uľahčenie procesu distribúcie a zbierania vypracovaní, môže a mala by ním byť aj snaha o čo najmenšiu spotrebu papiera.

Podľa Deshpandeho, ktorý sa odvoláva na štatistiky úradov v New Yorku spotrebuje jedna škola ročne približne 12 kg papiera na osobu. Pri štandardnej hmotnosti jedného listu papiera A4 rovnej 4,5 g nám vyjde na osobu približne 2700 listov papiera ročne (rátame v tom všetky zošity, učebnice, písomky a iné papierové dokumenty). Tieto čísla samozrejme môžeme brať s určitou rezervou, no asymptoticky nám ukazujú, že spotreba papiera v školstve je enormne vysoká. [1]

A teraz z ekonomickej stránky: Pre obvod s 10 000 študentmi nám vychádza približne 27 000 000 listov papiera za rok. Na Heuréka.sk nájdeme najlacnejší

balík papiera (500 listov, hmotnosť cca 5 g na kus) za 2,82 €. Pri vyššie zmienenom počte listov to vychádza približne na 150 000 € ročne.

Predstavme si, že by sme toto všetko elektronizovali. Ak list papiera drží informácie o približnej veľkosti 2 kB, všetkých 2 700 listov by sa zmestilo do 5,5 MB veľkého USB disku. Teda pre obvod s 10 000 študentmi by sa všetky ich informácie zmestili na USB disk o veľkosti 64 GB, ktorý nás vyjde do 30 €. Samozrejme, tieto všetky informácie môžeme úplne zadarmo ukladať v cloude od Microsoftu alebo Googlu. [1]

Aby sme sa však nebavili len v peňažnej rovine, takýmto prístupom by sme ročne zachránili 2 500 stromov a ušetrili 208 000 litrov ropy. [1]

Je úplne jasné, že táto idealizovaná predstava má zatiaľ ďaleko od reality, no pekne nám znázorňuje, ako dokáže niečo tak jednoduché ako napríklad odovzdávanie domácich úloh cez počítač pozitívne ovplyvniť nielen ekonomickú záťaž zriaďovateľa, ale aj životné prostredie.

2.4 Prieskum existujúcich riešení

Nakoľko je v súčasnosti po systémoch na spracovanie žiackych úloh dopyt aj v zahraničí, predstavíme si niekoľko aplikácií, ktoré plnia podobnú úlohu vo vyučovacom procese. V skutočnosti ich na internete nájdeme viac, no väčšina z nich je na stredoeurópskom trhu (najmä kvôli nedostupnej jazykovej mutácii) nepoužiteľná.

2.4.1 aSc EduPage

Aplikácia EduPage vznikla ako doplnok ku pomerne známemu softwareu aSc Agenda, ktorý predstavuje robustný interný informačný systém takmer každej školy na Slovensku. Ten zahŕňa ako evidenciu zamestnancov a žiakov, tak aj tvorbu rozvrhu, klasifikačný hárok, či dochádzkový systém na čipové karty. EduPage v tomto prípade predstavuje zredukované webové rozhranie pre učiteľov, žiakov a ich rodičov pre prístup k elektronickej žiackej knižke. [2]

Služba taktiež ponúka zadávanie domácich úloh s možnosťou PUSH notifikácie žiakov pomocou mobilnej aplikácie, avšak možnosti odovzdania vypracovania sú zredukované na vyplnenie textu vo wysiwyg editore, prípadne nahratie jedného priloženého súboru. Žiaci by tak museli niekoľkosúborové odovzdania zbaliť do zip-u a nahrať ich takýmto spôsobom. Učitelia by zas museli pracne sťahovať každé riešenie do svojho lokálneho počítača a rozbaliť ho. Pri dvoch alebo troch súboroch sa nám to nezdá veľa. Ak má však učiteľ vykonať túto činnosť 60-krát, je takýto spôsob hodnotenia odovzdaní vyslovene nežiadúci. Preto sa nakoniec využije jedine záznam hodnotenia do elektronického systému, zatiaľ čo samotná úloha je stále odovzdaná na papieri alebo v inom informačnom systéme. [3]

Výhody	Nevýhody
integrácia v najrozšírenejšom školskom informačnom systéme	nemožnosť odovzdať viac súborov
jednoduché a intuitívne prostredie	nemožnosť integrácie s Active Directory
notifikácie cez mobilnú aplikáciu	centralizované úložisko – hrozba výpadku
	cena

Tabuľka 2.1: Výhody a nevýhody EduPage

2.4.2 OneNote Class Notebook v Office 365

Aplikácia OneNote z balíku Microsoft Office už v dnešnej dobe prenikla (niekedy aj napriek vôli užívateľov) do každého počítača. Jedná sa o sofistikovaný poznámkový blok, ktorý kombinuje silné stránky všetkých štandardných programov v Office – Wordu, Excelu a PowerPointu. Medzi prednosti OneNoteu patrí možnosť zoradiť si poznámky do rôznych kategórií (napríklad podľa predmetu) a v jednotlivých kategóriách pridávať nové strany (paralela s listmi papiera v zošite). Užívatelia dotykových obrazoviek a tabletov určite ocenia možnosť písania vlastnou rukou a zakresľovania obrázkov.

V cloudovej verzii OneNote ponúka tzv. Class Notebook, ktorý funguje ako nadstavba nad štandardným OneNoteom. Funguje na princípe zdieľaného notebooku, ktorý založí učiteľ pre svoju triedu. Pomocou neho môže následne zdieľať učebné materiály pre žiakov a tak isto vložiť zadanie domácej úlohy. V prípade Class Notebooku však žiaci vypĺňajú nejakú učiteľom preddefinovanú šablónu (napríklad dopĺňanie slovíčok z cudzieho jazyka) a neodovzdávajú nový súbor. Preto sa takýto systém odovzdávania môže hodiť pre určitý typ predmetov, avšak pre odovzdávanie väčšieho množstva súborov nie je vhodný. [4]

Výhody	Nevýhody
integrácia v Office 365, teda aj s Active Directory	nemožnosť odovzdať externe vytvorené súbory
jednoduché a intuitívne prostredie – pochopia aj malé deti	náročnejší setup
možnosť vytvorenia šablóny pre odovzdanie	
uložené v cloude	
zadarmo	

Tabuľka 2.2: Výhody a nevýhody Class Notebooku

2.4.3 Moodle

„Systém Moodle [...] je softwarový balík pre tvorbu výukových systémov a elektronických kurzov na internete. Jedná sa o neustále sa vyvíjajúci projekt, navrhnutý na základe sociálno-konštruktivistického prístupu k vzdelávaniu. Jadrom tohto prístupu je myšlienka, podľa ktorej sa ľudia najviac naučia, ak existujú interakcie medzi nimi a študijnými materiálmi ako aj medzi ľuďmi samotnými.“ [5]

Moodle je jedným z najstarších systémov distribúcie vyučovacieho obsahu. *„Systém je poskytovaný zadarmo ako softvér s otvoreným kódom s všeobecnou verejnou licenciou GNU, ktorý je voľne dostupný. Práve takýto prístup k softvéru zásadne zmenil spôsob jeho vývoja.“* [5] Učitelia a programátori z celého sveta sa už viac ako 15 rokov podieľajú na jeho rozširovaní a skvalitňovaní. Práve preto je dostupný vo viac ako sto rôznych jazykoch. Keďže funguje na modulárnom princípe, môže si ho konkrétna škola do veľkej miery prispôsobiť podľa vlastných požiadaviek.

Učitelia majú možnosť vytvárať tzv. „kurzy“, do ktorých sa žiaci môžu prihlásiť. Následne sú im podľa učiteľom stanoveného plánu dávkované výukové materiály. Okrem toho môžu učitelia vložiť zadanie domácej úlohy. Princíp odovzdávania je však veľmi podobný tomu v EduPage, preto je tento systém použiteľný iba pri menších počtoch odovzdaných súborov.

Výhody	Nevýhody
integrácia s Active Directory	nemožnosť odovzdať viac súborov
modulárne prostredie	menej intuitívne prostredie
stále sa rozširuje	funguje lokálne v škole – hrozba výpadku
zadarmo	škola potrebuje server na prevádzku

Tabuľka 2.3: Výhody a nevýhody Moodlu

2.4.4 TeacherDashboard pre Office 365

Teacher Dashboard je robustný webový systém od britskej spoločnosti Axis12, ktorá sa okrem iného špecializuje na tzv. „content-heavy“ webové aplikácie a na vývoj aplikácií pre školstvo. [6]

Jadrom tohto projektu je úzke prepojenie s cloudom Office 365, vďaka ktorému prevádzka samotného softwareu nie je režijne extrémne náročná – takmer všetky dáta sa totiž ukladajú do cloudového úložiska školy. Systém funguje ako nadstavba nad cloudovou aplikáciou OneDrive a sofistikovane na nej spravuje súbory. Vďaka dobrej integrácii s Officeom má systém prístup k synchronizovaným Active Directory kontám a preto nie je nutné registrovať žiadne nové účty. [7]

Po zadaní domácej úlohy (alebo iného zadania) sa žiakom v ich osobných OneDrive diskoch vytvoria špeciálne priečinky (najprv podľa predmetov a v nich podľa konkrétnych zadaní), do ktorých žiaci nakopírujú svoje vypracovanie. Učiteľ si následne môže manuálne stlačením tlačidla alebo naprogramovaním deadlineu „vyzbierať“ úlohy. V skutočnosti sa vypracovania zo žiackych priečinkov prekopírujú do totožného priečinka na učiteľskom OneDrive a vďaka tomu nikdy nedôjde ku zmene zdrojových súborov alebo zmazaniu nedopatrením. Toto sa môže hodiť v prípade, že učiteľ vpisuje svoje poznámky do svojej kópie daného súboru a následne odovzdanie vráti žiakovi na prepracovanie. Žiak potom vidí svoju aj učiteľskú verziu. [8]

Software tak isto ponúka aj kontrolu proti plagiátorstvu, webové rozhranie pre študentov a rodičov a rôzne ďalšie vychytávky. Toto všetko sa samozrejme odráža aj na cene, ktorá je na skutočnosť, že software nemá slovenskú, ani českú jazykovú mutáciu, skutočne vysoká.

Výhody	Nevýhody
úzko previazané s Office 365	chýba preklad
intuitívne prostredie	cena 900 EUR/rok
rozhranie pre učiteľov, žiakov aj rodičov	
súbory sa ukladajú do cloudu	
možnosť odovzdať viac súborov	

Tabuľka 2.4: Výhody a nevýhody TeacherDashboard365

2.5 Požiadavky na aplikáciu pre zadávanie domácich úloh

Naša testovacia škola po presune súborového úložiska do cloudových služieb Office 365 nemôže a nechce naďalej využívať zraniteľné lokálne sieťové jednotky – nemôže však pre odovzdávanie domácich úloh využívať ani sharepointové stránky. Nakoľko sharepointové súbory neobsahujú rovnakú štruktúru prístupových oprávnení ako Windows ACL, nie je v nich možné nastaviť pre žiakov oprávnenie pre zápis súborov bez oprávnenia ich čítať. Preto je nutné vymyslieť taký systém odovzdávania domácich prác a písomiek, v ktorom si žiaci navzájom svoje vypracovania nebudú môcť pozrieť. Pre jednoduchšie definovanie požiadaviek na aplikáciu uvádzame nižšie zoznam funkčných a nefunkčných požiadaviek, ktoré by aplikácia mala spĺňať:

2.5.1 Funkčné požiadavky

Zobraziť skupiny

Systém umožní užívateľovi zobraziť skupiny užívateľov synchronizované z `Azure Active Directory` spolu s indikáciou, či je daná skupina školská – to znamená použiteľná na priradenie zadania.

Synchronizovať skupiny a užívateľov

Systém umožní užívateľom v role administrátora synchronizovať lokálnu databázu skupín s `Azure Active Directory` a rovnako aj databázu užívateľov patriacich do skupín `Ziaci` a `Zamestnanci`.

Upraviť skupinu

Užívatelia v role administrátora budú mať možnosť upraviť skupine parameter `IsClass` (rozumej „Je školská skupina?“), čím budú môcť zaradiť danú skupinu a jej členov medzi triedy, ktorým možno priradiť zadanie.

Vytvoriť lokálnu skupinu

Užívatelia v role učiteľa budú môcť vytvoriť lokálnu školskú skupinu, do ktorej budú môcť pridať vybraných členov skupiny `Ziaci`. Toto riešenie sa hodí pre predmety, v ktorých sa vyučovanie žiakov delí do skupín a tak by nebolo vhodné zadať zadanie celej triede.

Odstrániť lokálnu skupinu

Užívatelia v role učiteľa budú môcť odstrániť zo systému lokálnu školskú skupinu, ktorú predtým vytvorili, nebudú však môcť odstrániť cudzie lokálne skupiny, alebo skupiny synchronizované z `Azure Active Directory`. Užívatelia v role administrátora budú môcť odstrániť všetky lokálne skupiny.

Zobraziť predmety

Užívatelia v role učiteľa budú môcť zobraziť zoznam predmetov, v ktorých možno zadať zadanie.

Pridať predmet

Užívatelia v role administrátora budú môcť pridať predmety, v ktorých bude možné zadávať elektronické zadania.

Odstrániť predmet

Užívatelia v role administrátora budú môcť odstrániť predmet za podmienky, že v ňom v danom okamihu nebude aktívne[3] žiadne zadanie.

Pridať zadanie

Užívatelia v role učiteľa budú môcť pridať zadanie. Zadanie bude obsahovať názov, predmet, v rámci ktorého bude zadanie zadané, termín odovzdania (deadline), text zadania, v ktorom učiteľ žiakom vysvetlí detaily a nepovinnú súborovú prílohu.

Upraviť zadanie

Užívatelia v role učiteľa budú môcť v prípade potreby zmeniť parametre zadania vlastného zadania, najmä deadline, ak sa rozhodnú termín odovzdania posunúť, alebo text, ak sa rozhodnú zadanie bližšie špecifikovať alebo pozmeniť.

Priradiť zadanie žiakom

Užívatelia v role učiteľa budú môcť už vytvorené vlastné zadanie priradiť jednej alebo viacerým skupinám označeným ako školské.

Zobraziť zoznam zadaní

Systém umožní užívateľom v role učiteľa zobraziť zoznam zadaní, zoradený podľa ich aktuálnosti s tým, že expirované zadania budú farebne odlíšené.

Zobraziť detail zadania

Systém umožní užívateľom v role učiteľa zobraziť detail zadania (aj cudzieho), v ktorom sa okrem základných informácií o zadaní bude nachádzať aj zoznam žiakov, ktorým bolo zadanie pridelené s indikáciou, či už k danému zadaniu niečo odovzdali.

Odoslať notifikačný e-mail

Užívatelia v role učiteľa budú môcť pri priradení zadania skupinám voliteľne odoslať členom týchto skupín notifikačný e-mail o pridaní nového zadania. E-mail by mal obsahovať všetky dôležité informácie o zadaní: jeho názov, predmet, deadline a text.

[3] Aktívnym zadaním rozumieme zadanie, ktorého termín odovzdania je v budúcnosti.

Vyzbierať vypracovania

Užívatelia v role učiteľa budú môcť vyzbierať vypracovania nimi vytvoreného zadania od žiakov, ktorým bolo zadanie priradené. Toto budú môcť robiť nekonečne veľakrát s tým, že sa tieto vyzbierané vypracovania budú verziovať, nie prepisovať.

Odstrániť zadanie

Užívatelia v role učiteľa budú môcť odstrániť svoje vlastné zadanie. Užívatelia v role administrátora budú môcť odstrániť hociktoré zadanie.

2.5.2 Nefunkčné požiadavky

Nefunkčné požiadavky špecifikujú vlastnosti a obmedzujúce podmienky prevádzky aplikácie.

- Aplikácia bude sprístupnená ako webová služba.
- Webová stránka aplikácie bude mať responzívny design – bude možné ju otvoriť aj na tablete, resp. smartfóne.
- Užívatelia sa budú autentifikovať v aplikácii pomocou `Azure Active Directory`
- Aplikácia by mala byť čo najužšie previazaná s existujúcimi dátovými zdrojmi – `Azure Active Directory` a `aSc Edupage`
- Aplikácia by mala byť ľahko rozšíriteľná o ďalšie dátové zdroje.

2.6 Modelovanie prípadov použitia

Modelovaním prípadov použitia (anglicky *use case modelling*, skratka UC) môžeme zachytiť interakciu medzi užívateľom a systémom a zároveň ním pokryť funkčné požiadavky. Model môžeme zachytiť ako textovo, tak aj graficky použitím UML diagramu. V oboch prípadoch sa pracuje s dvoma typmi entít. Prvou z nich je tzv. aktér, resp. zúčastnená strana. Aktérom môže byť užívateľ, systém alebo hardwareové zariadenie, ktoré komunikuje so sledovaným systémom. Druhou entitou je potom prípad užitia, ktorý špecifikuje konkrétnu časť funkcionality systému využívanej aktérom. [9]

K modelovaniu prípadov užitia bol použitý software Enterprise Architect, v ktorom je možné podrobne opísať každý prípad užitia. Na nasledujúcich stranách z nich spomenieme niekoho najzaujímavejších.

UC01: Zobrazenie zoznamu zadaní

Užívateľ sa autentifikuje pomocou svojho doménového konta a následne sa mu kliknutím na odkaz „Zadania“ zobrazí zoznam zadaní v tabuľkovom formáte. Implicitne budú zobrazené len zadania, ktorých je daný užívateľ autorom. Stlačením navigačného tlačidla však bude možné odkryť aj zadania ostatných užívateľov.

V tabuľke budú zobrazené nasledujúce informácie: autor, názov, predmet, dátum vytvorenia, dátum odovzdania (deadline) a tlačidlo Akcie. Stlačením tlačidla akcie sa užívateľovi zobrazí ponuka s nasledovnými akciami: Upraviť, Odstrániť, Prideliť triedam, Vyzbierať, Zobraziť priečinok a Detail. Tieto tlačidlá, s výnimkou tlačidla „Detail“ budú dostupné len autorovi daného zadania.

Užívateľ bude mať tak isto možnosť nechať zobraziť len aktuálne platné zadania, prípadne všetky zadania.

UC02: Vytvorenie zadania

Užívateľ po úspešnom vykonaní prípadu použitia UC01 bude môcť stlačiť tlačidlo „Pridať“, ktorým spustí zobrazenie formuláru pre pridanie nového zadania. Formulár bude mať nasledovné polia: predmet, názov, dátum odovzdania a text zadania. Po vyplnení a stlačení tlačidla „Uložiť“ server skontroluje validitu vpísaných dát a následne záznam uloží do databázy. V prípade, že dáta validné nebudú, upozorní na to užívateľa chybovou hláškou. Po úspešnom uložení dát bude užívateľ presmerovaný opäť na UC01.

UC03: Priradenie zadania

Užívateľ po úspešnom vykonaní prípadu použitia UC02 bude môcť svoje zadanie stlačením tlačidla „Priradiť zadanie“ priradiť jednotlivým triedam. Zobrazí sa mu zoznam tried, z ktorých si bude môcť vybrať jednu alebo niekoľko tried, ktoré označí. Pri tomto výbere bude môcť tak isto označiť, či si praje študentov daných tried notifikovať e-mailovou správou. Stlačením tlačidla „Uložiť“ sa zadanie vytvorí v databáze a žiakom daných tried sa odošlú informačné e-maily. Po úspešnom uložení dát bude užívateľ presmerovaný opäť na UC01.

UC04: Upravenie zadania

Užívateľ kliknutím na tlačidlo „Upraviť“ v UC01 bude presmerovaný na formulár podobný tomu v UC02 už s predvyplnenými údajmi. Určité polia však budú voči úpravám zablokované, vzhľadom na to, či už dané zadanie bolo priradené triedam. Ak bolo, bude možné upraviť iba polia „dátum odovzdania“ a „text“; ak nebolo, bude možné upraviť všetky polia.

Po vyplnení a stlačení tlačidla „Uložiť“ server skontroluje validitu vpísaných dát rovnako ako v UC02 a následne záznam uloží do databázy. V

prípade, že dáta validné nebudú, upozorní na to užívateľa chybovou hláškou. Po úspešnom uložení dát bude užívateľ presmerovaný opäť na UC01.

UC05: Vyzbieranie vypracovaní

Užívateľ kliknutím na tlačidlo „Vyzbierať“ v UC01 spustí úlohu na pozadí, ktorá od jednotlivých žiakov vyzbiera vypracovania daného zadania. O úspešnosti operácie bude užívateľ informovaný notifikáciou.

UC06: Detail zadania

Užívateľ kliknutím na tlačidlo „Detail“ v UC01 bude presmerovaný na stránku s detailnou kartou daného zadania. Stránka užívateľovi vo vrchnej časti zobrazí dôležité informácie o zadaní, ako je jeho názov, či sprievodný text a v spodnej strane bude užívateľovi zobrazovať zoznam študentov, ktorým bolo dané zadanie priradené s aktuálnym stavom ich vypracovania (tj. či už svoje vypracovanie odovzdali alebo nie).

UC07: Zobrazenie zoznamu skupín

Užívateľ sa autentifikuje pomocou svojho doménového konta a následne sa mu kliknutím na odkaz „Skupiny“ zobrazí aktuálny zoznam skupín v aplikácii v tabuľkovom formáte. Ku každej skupine bude uvedený jej názov a indikácia, či môže byť danej skupine priradené zadanie. Užívatelia v role administrátora budú môcť tieto atribúty upraviť stlačením tlačidla „Upraviť“.

UC08: Synchronizácia užívateľov a skupín

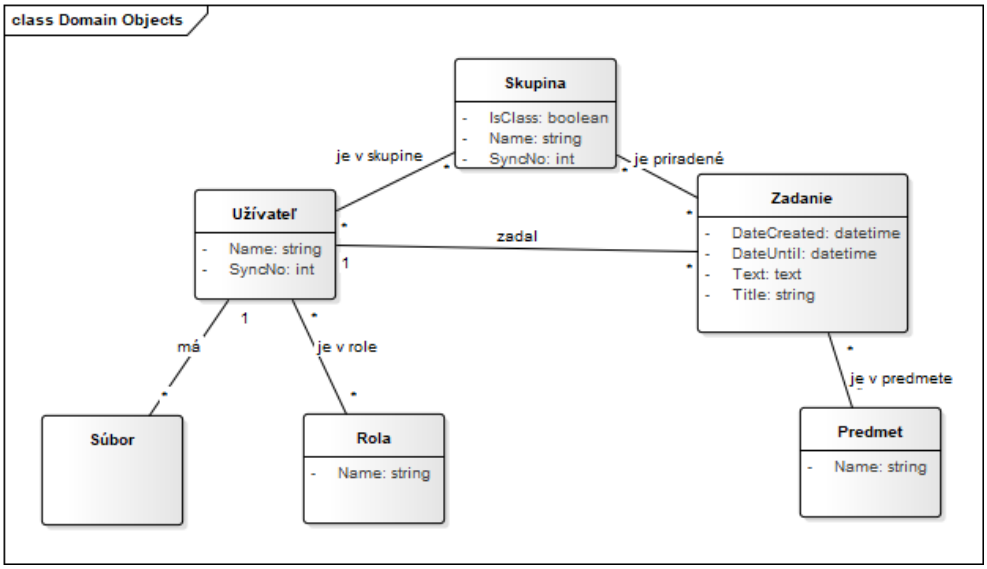
Užívateľ v role administrátora kliknutím na tlačidlo „Synchronizovať“ spustí úlohu na pozadí, ktorá zosynchronizuje lokálnu databázu skupín a užívateľov s databázou `Azure Active Directory`. Po dokončení operácie bude užívateľ informovaný o úspechu, resp. neúspechu notifikáciou.

2.7 Doménový model

Doménovým modelom môžeme znázorniť základné entity našej aplikácie tak, ako logicky vyplývajú z požiadaviek zadávateľa. Najčastejšie sa pre doménový model používa notácia zjednodušeného diagramu jazyka UML. Jednotlivé entity sú v ňom znázornené ako triedy, pričom každá trieda môže obsahovať ľubovoľný počet atribútov. Doménový model je nezávislý na voľbe implementačnej platformy. [9]

V návrhu nášho doménového modelu na obrázku 2.1 vidíme základné entity: Zadanie, Skupina, Predmet, Užívateľ, Rola a Súbor, ktoré prirodzene vyplývajú z funkčných požiadaviek. Trieda Súbor v tomto modeli predstavuje študentské vypracovanie zadania.

Pri návrhu doménového modelu si musíme dávať obzvlášť pozor na to, aby v ňom nevznikli cykly. Ako si iste všimneme, vznikol nám jeden cyklus: Užívateľ – Zadanie – Skupina. V takomto prípade treba cieleným komentárom ošetriť tento cyklus a zdôvodniť, prečo nikdy nenastane. V našom prípade predstavuje vzťah Užívateľ – Zadanie vzťah autora k jeho zadaniu, jedná sa teda o učiteľa, člena skupiny, ktorá nie je triedou. Vzťah Zadanie – Skupina je naopak priradzovací vzťah, ktorým sa priradzuje zadanie skupinám, ktoré triedami sú. Vzťah Užívateľ – Skupina je spoločný ako pre žiakov, tak aj učiteľov. Cyklus teda nikdy nenastane preto, lebo skupine, ktorej členom je učiteľ zadávajúci zadanie nikdy toto zadanie nemožno priradiť.



Obr. 2.1: Doménový model

Kapitola 3

Návrh

3.1 Návrh riešenia

V tejto časti sa budeme venovať rozboru funkčných a nefunkčných požiadaviek zo state 2.5 a budeme na ne navrhovať technické riešenia. V návrhu sa budeme opierať najmä o požiadavku úzkeho previazania našej aplikácie s `Azure Active Directory`.

3.1.1 Správa skupín

Z požiadaviek na aplikáciu vyplýva, že aplikácia by mala okrem štandardných parametrov skupiny, ako sú jej názov, identifikačné číslo a zoznam jej členov uchovávať aj informáciu, či daná skupina môže byť použitá v aplikácii pre priradenie zadania. Znamená to, že aplikácia si niekde tento údaj musí uchovávať. Na výber sú dve možnosti: buď sa bude tento atribút ukladať ako prídavný atribút v `Azure Active Directory`, čo by si potenciálne mohlo vyžadovať rozšírenie schémy, alebo sa skupiny budú synchronizovať s lokálnou databázou, v ktorej budeme môcť pridať ku každej skupine ľubovoľný počet atribútov, ktoré si o nej budeme potrebovať uchovávať.

Pozerajúc sa na to filozofiou „preč z lokálu, všetko do cloudu“ by bolo iste vhodnejšie zvoliť rozšírenie schémy. Pre prototypovú aplikáciu však bude jednoduchšie raz za čas zosynchronizovať tabuľku o pár stĺpcoch, než komplikovane zapisovať do rozšíreného parametru schémy a permanentne sa naň potom pýtať cloudu. Zároveň poznajúc miestne pomery môžeme s kľudom tvrdiť, že skupiny sa do `Active Directory` pridávajú veľmi zriedkavo, maximálne raz do roka, preto koniec koncov nebude potrebné synchronizovať skupiny extrémne často.

Všimnime si, že v predošlom odstavci nie je spomenutá synchronizácia členov jednotlivých skupín. Ak zvážime fakt, že členstvo v skupine môže byť oveľa dynamickejším atribútom, ako údaje o skupine ako takej, prídeme na to,

že práve členov skupiny budeme chcieť poznať vždy v reálnom čase – preto sa na nich budeme vždy pýtať priamo cloudu.

3.1.2 Správa užívateľov

Ako sme si už povedali, oproti skupinám sú užívatelia oveľa dynamickejšou množinou objektov. Z požiadaviek na aplikáciu vyplýva, že u každého užívateľa si bude musieť aplikácia evidovať jeho rolu (alebo viac rolí) na základe jeho členstva v niektorej zo špeciálnych `Active Directory` skupín. Zároveň budú mať užívatelia prepojenie na zadania či už ako učiteľ, ktorý zadanie vytvoril, alebo ako žiaci, ktorí zadanie dostali.

Opäť sa nám ponúkajú dve možnosti: buď rozšírime užívateľskú schému o atribút role a budeme sa pri každom odkaze na užívateľa cez jeho GUID (globálny identifikátor) pýtať cloudu na jeho meno, alebo zvolíme opäť synchronizačnú možnosť, v ktorej si doslova „vycacheujeme“ najpoužívanejšie údaje o užívateľoch do lokálnej databázy. V tomto prípade zvoľme opäť jednoduchšiu možnosť, ktorá jednak ušetrí počet požiadaviek na cloud a zároveň je ľahšie implementovateľná: použijeme synchronizáciu.

3.1.2.1 Užívateľské roly

Aplikácia bude prideľovať užívateľom 3 základné roly, ktoré budú súvisieť s ich členstvom v jednej zo špeciálnych `Active Directory` skupín. Užívateľovi môže byť pridelená viac ako jedna rola, napríklad Učiteľ + Administrátor.

Žiak

Táto rola bude pridelená všetkým členom `Active Directory` skupiny `Ziaci`, teda suverénne najväčšiemu počtu užívateľov. Je to rola s najmenšími oprávneniami, ktorej sú pridelené nasledovné funkcie:

- Zobraziť všetky zadania, ktoré sú mu pridelené.
- Zobraziť detail každého zadania (okrem zoznamu žiakov a ich stavu odovzdania).
- Povolenie zápisu do vyhradeného priečinka pre odovzdanie konkrétneho vypracovania (dôležité: vypracovanie bude s veľkou pravdepodobnosťou obsahovať viac súborov/priečinkov)

Učiteľ

Táto rola bude pridelená všetkým členom `Active Directory` skupiny `Zamestnanci`, teda všetkým učiteľom. Obsahuje oprávnenia pre prístup ku základným funkcionalitám aplikácie, potrebným pre každodenné používanie:

- Zobraziť všetky skupiny a ich členov.

- Vytvoriť a odstrániť lokálnu skupinu.
- Zobraziť predmety.
- Zobraziť všetky zadania.
- Pridať, upraviť a odstrániť vlastné zadanie.
- Priradiť zadanie žiakom.
- Zobraziť detail zadania (aj so zoznamom žiakov a indikáciou stavu odovzdania u každého žiaka.
- Otvoriť každé odovzdané vypracovanie vlastného zadania (bez nutnosti manuálneho sťahovania do lokálneho počítača).

Administrátor

Táto rola bude pridelená vybraným členom personálu manuálne. Užívatelia s touto rolou budú poverení spravovaním aplikácie v konkrétnej škole. Množina povolení neobsahuje povolenia pre rolu Učiteľ, sú v nej definované len povolenia, ktoré rola Učiteľ neobsahuje.

- Pridať, upraviť a odstrániť predmety.
- Synchronizovať skupiny a žiakov.
- Upraviť synchronizované skupiny.
- Pridať, upraviť a odstrániť lokálnu skupinu.
- Upraviť a odstrániť všetky zadania.

3.1.3 Správa predmetov

Z požiadaviek na aplikáciu vyplýva, že aplikácia by si mala ukladať údaje o predmetoch. Keďže pre takúto entitu zatiaľ v cloude neexistuje ekvivalent, bude potrebné si jednotlivé predmety ukladať v spoločnej tabuľke v lokálnej databáze. Keďže predmety vo svojej podstate slúžia iba ako kategórie pre zadania, bude postačovať ukladať si iba ich identifikačné číslo a názov.

3.1.4 Správa zadaní

Asi najdôležitejšou časťou celého systému bude správa zadaní, ktorá by mala všetky predošlé entity a služby spájať dohromady v jeden celok. Keďže zadanie bude komplexná entita, bude sa určite odkazovať na ďalšie entity, ako napríklad na zosynchronizované skupiny, ktoré majú nastavený atribút `IsClass` vo výbere skupín pri priradení zadania, ale tak isto aj na zosynchronizovaných užívateľov pri odkazovaní sa na autora zadania. Nezabudnime na

to, že pri priradení zadania sa tak isto aplikácia spýta cloudu v reálnom čase na členov jednotlivých skupín, ktorým má zadanie priradiť.

3.1.5 Integrácia dát z Azure Active Directory

Kedže celá školská sieť je postavená na windowsovej doméne riadenej `Active Directory` kontrolérom a momentálne je už tento synchronizovaný aj s `Azure Active Directory` v cloude, je pochopiteľné, že v požiadavkách figuruje aj integrácia s týmto systémom. Vďaka nej nebude nutné užívateľov do aplikácie registrovať a vytvárať im nové účty a rovnako nebude nutné zaradzovať ich do vyučovacích skupín (resp. tried) a manuálne im priradzovať roly. Pri správnom napojení na `Azure Active Directory` nám jeho SSO brána autentifikuje aj autorizuje užívateľov – teda overí ich totožnosť a pridelí im podľa prednastavených pravidiel roly. Pre splnenie tejto požiadavky bude musieť aplikácia použiť jedno z Microsoftom poskytovaných API. Komunikácii s cloudom od Microsoftu sa budeme osobitne venovať v kapitole 4.3.

3.1.6 Úložisko vypracovaní

Keďže škola v požiadavkách uviedla svoj negatívny postoj ku lokálnym diskom, sme nútení siahnuť po cloudovom riešení. Ako sme už spomenuli v analýze východiskovej situácie v stati 2.1, Microsoft ponúka všetkým školám jednaterabajtové osobné úložisko pre každého žiaka a učiteľa zdarma. S ohľadom na túto ponuku a na požiadavku autentifikácie a autorizácie cez `Azure Active Directory` zvolíme ako najvhodnejšie riešenie pre túto situáciu použitie cloudu Office 365.

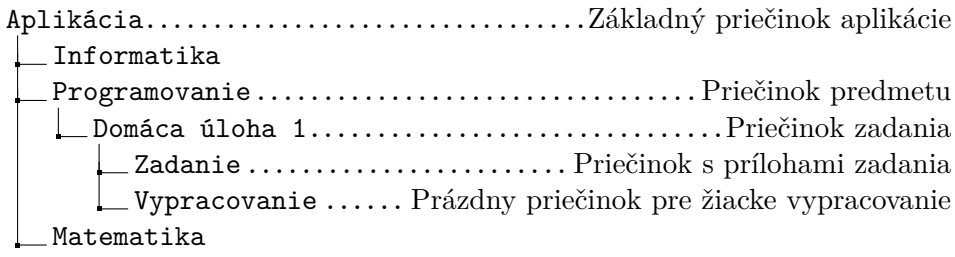
3.1.7 Proces zadania a vypracovania

Keďže sme si už definovali, na akých technológiách budeme stavať, porozmýšľajme nad tým, ako by sme mohli preniesť scenár zadania a odovzdania domácich úloh zo state 2.2.1 do návrhu našej aplikácie.

K dispozícii v Office 365 máme dva typy úložísk: osobné OneDrive úložisko pre každého užívateľa a skupinové SharePoint stránky. Z hľadiska autenticity odovzdaných úloh by bolo nanajvýš vhodné, aby žiaci nemali prístup ku odovzdaniam svojich spolužiakov – preto možnosť využitia sharepointových stránok zamietneme.

Ostávajú nám teda osobné OneDrive úložiská, ktoré už momentálne naša testovacia škola využíva ako osobné užívateľské disky. Toto môže byť koniec koncov výhodou, pretože sú všetci užívatelia na používanie tohto typu úložiska už zvyknutí.

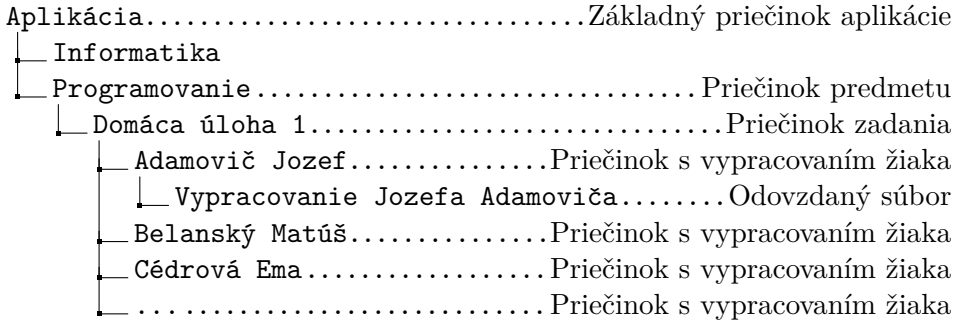
V osobných úložiskách užívateľov bude teda stačiť vytvoriť špeciálny priečinok pre našu aplikáciu a v ňom adresárovú štruktúru podľa predmetov a jednotlivých zadaní. Adresárovú štruktúru aplikačného priečinku v žiackom úložisku znázorňuje obrázok 3.1.



Obr. 3.1: Adresárová štruktúra u žiakov

Keďže si aplikácia bude v databáze uchovávať názov zadania, bude môcť v príhodný čas od každého žiaka, ktorému bude toto zadanie priradené, skopírovať súbory z priečinku `Vypracovanie` na osobný OneDrive učiteľa, ktorý dané zadanie zadal. Adresárová štruktúra aplikačného priečinku u učiteľa sa bude líšiť len v tom, že namiesto priečinkov `Vypracovanie` a `Zadanie` sa v adresári konkrétneho zadania budú nachádzať priečinky s menami žiakov a ich skopírovanými vypracovaniami (viď obrázok 3.2).

Vďaka tomuto prístupu budú mať učitelia všetky odovzdané vypracovania zhromaždené na jednom mieste, ku ktorému budú mať prístup ako z domu, tak aj zo školy. Autenticita súborov ostane zachovaná, pretože ich bude k učiteľovi kopírovať naša aplikácia, nie žiaci samotní. Zároveň vďaka taktike kopírovania súborov namiesto ich presúvania sa nebude môcť stať prípad, v ktorom by odovzdanie kvôli zozbieraniu „záhadne“ zmizlo zo žiackeho priečinka. Aby si mohol učiteľ vypracovania vyzbierať niekoľkokrát, budeme pri každom ďalšom zbere vypracovaní vytvárať nový priečinok zadania u učiteľa s priradeným číslom – takto bude mať učiteľ prehľad, ktorý zber prebehol ako prvý a ktorý ako posledný.



Obr. 3.2: Adresárová štruktúra u učiteľov

3.2 Dátové entity

Kedže sme si už vyššie popísali, akým spôsobom môžeme preniesť požiadavky zadávateľa do praxe, rozoberieme si dátové entity, ktoré budú zastupovať jednotlivé zdroje, s ktorými budeme pracovať. Pre prehľadnosť budeme pri každej entite uvádzať, kde sa jej dáta ukladajú, aby sme jasne odlíšili lokálne aplikačné entity od tých cloudových.

Skupina – Lokálna databáza

Aplikácia musí pre priradenie zadania vyučovacej skupine evidovať tieto skupiny a atribút `IsClass`, ktorý bude nastavený podľa toho, či danej skupine môže byť priradené zadanie. Keďže budú skupiny do databázy pridávané jednostrannou synchronizáciou z `Azure Active Directory`, musí byť ku každej skupine priradený aj atribút ukazujúci, či už konkrétna skupina bola v danom synchronizačnom behu synchronizovaná. Atribúty entity `Groups` teda budú:

Id Jedinečný identifikátor skupiny, skopírovaný z `Azure Active Directory`.

Name Názov skupiny, skopírovaný z `Azure Active Directory`.

IsClass Parameter indikujúci, či môže byť daná skupina použitá pre priradenie zadania.

Assignments Množina zadaní, ktoré sú skupine priradené.

SyncNo Synchronizačné číslo, indikujúce, či bol daný objekt synchronizovaný.

Skupina – Microsoft Graph

Aplikácia musí kvôli synchronizácii a načítaniu aktuálnych členov vyučovacej skupiny kontaktovať cloud Office 365 cez rozhranie Microsoft Graph. Ten vracia ako odpoveď na požiadavku entitu `Microsoft.Graph.Group`, skrátene `Group`, ktorá obsahuje atribúty ako: názov, identifikátor, vlastník atď. Pre účely našej aplikácie nás budú zaujímať len nasledovné atribúty:

Id Jedinečný identifikátor `GUID` prenesený z lokálneho školského `Active Directory`.

displayName Názov skupiny.

Members Aktuálni členovia skupiny.

Užívateľ – Lokálna databáza

Aplikácia si musí kvôli priradeniu rol uchovávať aj databázu užívateľov. Každému užívateľovi je teda priradená minimálne jedna rola. Zároveň lokálna databáza užívateľov slúži ako akási cache na mená užívateľov, aby sme sa na ne nemuseli podľa ich `Id` vždy dotazovať cloudu. Atribúty entity `Users` teda budú:

Id Jedinečný identifikátor užívateľa, skopírovaný z `Azure Active Directory`

Name Meno a priezvisko užívateľa, skopírované z `Azure Active Directory`

Assignments Množina zadaní, ktorých je užívateľ autorom.

Roles Množina rol, ktoré sú užívateľovi priradené.

SyncNo Synchronizačné číslo, indikujúce, či bol daný objekt synchronizovaný.

Užívateľ – Microsoft Graph

Aplikácia musí kvôli synchronizácii a kvôli prístupu do osobných úložísk užívateľov pravidelne kontaktovať Microsoft Graph. Ten vracia ako odpoveď na požiadavku entitu `Microsoft.Graph.User`, skrátene `User`, ktorá obsahuje množstvo informácií o každom užívateľovi. Pre účely našej aplikácie nas však budú zaujímať len nasledovné atribúty:

Id Jedinečný identifikátor `GUID` prenesený z lokálneho školského `Active Directory`.

displayName Meno a priezvisko užívateľa.

givenName Krstné meno užívateľa.

surname Priezvisko užívateľa.

mail E-mailová adresa užívateľa.

drive Referencia na osobné OneDrive úložisko. Pomocou tejto referencie bude vedieť aplikácia narábať so súbormi a priečinkami na užívateľskom OneDrive.

Rola – Lokálna databáza

Aplikácia si v lokálnej databáze ukladá číselník rol, ktoré sú následne užívateľom podľa ich členstva v niektorej z `Active Directory` skupín pridelené. Entita `Roles` si uchováva nasledovné údaje:

Id Jedinečný identifikátor role – číselný.

Name Názov role – jedinečný identifikátor pre párovanie so skupinami v `Azure Active Directory`

Users Množina užívateľov, ktorým je daná rola priradená.

Predmet – Lokálna databáza

Aplikácia si v lokálnej databáze ukladá predmety, ktoré slúžia pre kategorizáciu zadaní. Kedže sa jedná len o pomocnú entitu, entita `Subjects` nemá veľa antribútov:

Id Jedinečný identifikátor predmetu.

Name Názov predmetu.

Assignments Množina zadaní, ktoré sú zadané v danom predmete.

Zadanie – Lokálna databáza

Zadanie je samozrejme najkomplexnejšou entitou, ktorá v sebe sústreďuje informácie zo všetkých ostatných pomocných entít. O zadaní si aplikácia neeviduje len základné informácie ako sú názov, čas odovzdania a podobne, ale aj dátum vytvorenia, autora a skupiny, ktorým je zadanie pridelené. Entita `Assignments` obsahuje nasledovné údaje:

Id Jedinečný identifikátor zadania.

AuthorId Jedinečný identifikátor autora zadania.

SubjectId Jedinečný identifikátor predmetu, v ktorom je zadanie vytvorené.

Title Názov zadania.

DateCreated Dátum a čas vytvorenia zadania.

DateUntil Dátum a čas odovzdania vypracovania daného zadania.

Text Rozšírujúci text zadania.

Groups Množina skupín, ktorým je dané zadanie priradené.

Poznámka: Dôsledkom používania EntityFrameworku v implementačnej časti sa namapovali lokálne entity na názvy tabuliek v lokálnej databáze, preto tie názvy „Groups“, „Users“ a podobne. Zároveň sa nešťastnou náhodou volajú cloudové entity rovnako ako singulár názvov lokálnych entít, teda „Group“, „User“, preto nebolo možné lokálne entity premenovať na ich singulár. Autor si túto nepríjemnosť uvedomuje a verí, že mu ju čitateľ odpustí.

Kapitola 4

Použité technológie

Na základe vyššie rozanalyzovaných požiadaviek našej testovacej školy, s ktorou sa vo veľkej miere zhodujú aj požiadavky väčšiny základných a stredných škôl na Slovensku, sa javí ako najvhodnejšie riešenie vytvorenie aplikácie komunikujúcej s cloudovými službami Office 365, ktorá by rozumným spôsobom spravovala úložiská jednotlivých žiakov a učiteľov k dosiahnutiu požadovaného výsledku.

Microsoft pre tieto účely ponúka svoje API s názvom Microsoft Graph, ktoré funguje svojím spôsobom ako brána ku jednotlivým cloudovým službám (napr. k Outlooku, Sharepointu, či OneDriveu). Aby sa vývojári nemuseli zaoberať formovaním HTTP požiadavkov na Graph, sú im k dispozícii tzv. „SDK“ – Software Development Kit pre nasledujúce platformy:

- Android
- Angular
- ASP.NET MVC
- iOS Swift
- Node.js
- PHP
- Python
- Ruby
- Universal Windows Platform (.NET)
- Xamarin

Keďže máme za cieľ vyvinúť serverovú aplikáciu s webovým rozhraním, ktorá by ideálne mala bežať v cloude, ako najvhodnejšia voľba sa javí framework ASP.NET MVC, v ktorom môžeme využiť silné stránky platformy .NET a pretaviť ich vďaka návrhu MVC do ľahko udržiavateľného a pochopiteľného kódu.

4.1 ASP.NET

ASP.NET je opensourcový serverový framework určený na vývoj webových aplikácií, ktorý produkuje dynamické webové stránky. Jeho predchodcom až do jeho vydania v roku 2002 bol skriptovací jazyk ASP (vo svojej podstate podobný PHP), ktorý sa však veľmi neuchytil. Oproti svojmu predchodcovi má ASP.NET 2 nesporné výhody: Jednou je to, že kód je predkompilovaný do jedného alebo niekoľko málo dll súborov, vďaka čomu sú ASP.NET stránky rýchlejšie, zatiaľ čo pri starom ASP sa pri každom volaní stránky musel kód znovu a znovu parsovať. Druhou je to, že ASP.NET je postavený na princípe Common Language Runtime, ktorý umožňuje programátorom písať kód pre ASP.NET v hociktorom z .NET frameworkom podporovaných jazykov (napríklad C#, Visual Basic, JScript.NET, Managed C++ a ďalšie).

V ASP.NET sa dá šikovne naprogramovať asi všetko – od malej osobnej stránky až po veľký korporátny portál. Aj keď si stránky môžeme naprogramovať vo Visual Studiu zadarmo, webhostingov podporujúcich ASP.NET je ako šafránu. Ak si pre porovnanie vezmeme webstránky bežiace na PHP, zistíme, že stránok naprogramovaných v ASP.NET je veľmi málo. Ak sa však zameriame len na veľké projekty a korporátne portály, beží ich na ASP.NET pomerne veľa.

Ako sme si už povedali, ASP.NET beží na strane serveru a na základe klientských požiadaviek generuje webstránku, ktorú klientovi pošle. Ten už vidí iba výsledné HTML, v ktorom nie je po ASP.NET ani stopy. Prevádzkovať tento framework možno optimálne na serveri IIS od Microsoftu (podmienkou je teda OS Windows) alebo na webserveri Apache s prídavným modulom Mono. Toto je práve dôvodom, prečo tak málo webhostingov podporuje túto technológiu.

ASP.NET ponúka 3 frameworky na vytváranie webových aplikácií: ASP.NET Web Forms, ASP.NET MVC a ASP.NET Web Pages. Všetky tri sú stabilné a už dlhší čas používané, no každý z nich je vhodný pre iný typ aplikácie. [10], [11]

4.1.1 ASP.NET Web Forms

Web Forms je určený pre programátorov, ktorí preferujú deklaratívne programovanie založené na *„controls“* (čiže formulárových políčkach, tlačidlách a pod.). Jedná sa o pokus preniesť WinForms (teda štandardné formulárové aplikácie, ako ich poznáme z desktopu) na web. Hlavná myšlienka spočíva vo

wysiwyg editore, v ktorom si programátor vyskladá formulár z jednotlivých controllov z toolboxu a potom im priradí udalosti. Aplikácia sa navonok tvári ako desktopová, ale na pozadí beží zložitejšia logika. Úloha nasimulovať desktopovú aplikáciu je pomerne ťažká, nakoľko protokol HTTP je bezstavový.

Stavom sa myslia napríklad vyplnené hodnoty, zaškrtnuté checkboxy a tak podobne. Server vôbec nevie, v akom stave je aplikácia až dovtedy, kým mu nepríde od klienta požiadavka, na ktorú reaguje generovaním stránky. Po odoslaní stránky klientovi väčšina dát na serveri zaniká. Napríklad pri odoslaní formulára by sme očakávali, že nám jeho hodnoty ostanú vpísané v políčkach, najmä ak nám server vygeneruje ako výsledok operácie ten istý formulár. Stav formulára sa teda musí niekde ukladať, čo ASP.NET Web Forms rieši takzvaným ViewState. Nejedná sa o nič iné, než o skryté formulárové pole, v ktorom sa nachádza zakódovaný reťazec indikujúci serveru stavy jednotlivých formulárových polí. Vďaka tomuto procesu si formulár pamätá hodnoty svojich políčok aj po odoslaní a užívateľ má pocit, že pracuje s desktopovou aplikáciou.

Výhodou takéhoto prístupu je enormná rýchlosť tvorby aplikácií, najmä tých, ktoré obsahujú množstvo formulárov. Nevýhodou je samozrejme zvýšená veľkosť HTML stránky kvôli odosielaniu ViewState a nemožnosť upravovať HTML zobrazenie určitých controllov. ASP.NET Web Forms je starší koncept, ktorý sa napriek tomu ešte stále využíva vo firmách. [10], [11]

4.1.2 ASP.NET Web Pages

ASP.NET Web Pages je framework určený pre programátorov, ktorým záleží na čo najjednoduchšom vývoji. Vývoj vo Web Pages je veľmi podobný písaniu kódu v PHP. Najprv si vytvoríme HTML, do ktorého neskôr vpisujeme serverový kód, ktorý pri spracovaní dynamicky kontroluje vygenerovanú stránku. Framework Web Pages je špecificky nadizajnovaný ako ľahšia alternatíva ku ostatným dvom ASP.NET frameworkom a stal sa najľahšími dverami do sveta ASP.NET či už pre programátorských nadšencov alebo vývojárov zvyknutých na PHP.

Rovnako ako Web Forms je aj Web Pages orientovaný na čo najrýchlejší vývoj. Programátorom ponúka tzv. *„helpers“* (slovensky pomocníkov), ktorí ponúkajú už hotové riešenia pre komplexné programovacie úlohy ako napríklad prihlásenie používateľov prostredníctvom Facebooku, vytiahnutie dát z databázy a tak podobne.

Web Pages ponúkajú jednoduchší prístup, než Web Forms. Ak sa pozrieme na jednotlivé súbory, môžeme si vo všeobecnosti predstaviť logiku programu ako procedurálnu, teda zhora nadol, podobne ako v PHP. A nakoľko sú súbory .cshtml a .vbhtml stále vo svojej podstate HTML, veľmi jednoducho sa do nich implementujú aj funkcionality na klientskej strane ako je napríklad javascriptová jQuery validácia. [10], [11]

4.1.3 ASP.NET MVC

ASP.NET MVC je určené pre vývojárov, ktorým záleží na vzoroch a princípoch ako test-driven development, separation of concerns, inversion of control a dependency injection. Tento framework podporuje oddelenie logickej vrstvy od prezentačnej vrstvy aplikácie.

V ASP.MVC prichádzajú vývojári viac do kontaktu s HTML a HTTP ako vo Web Forms. Zatiaľ čo Web Forms sa snažili čo najviac napodobňovať správanie desktopovej aplikácie napríklad automatickým používaním ViewState, v MVC si programátori musia všetko naprogramovať sami. Na druhej strane im to dáva plnú kontrolu nad tým, ako sa aplikácia bude v konkrétnych prípadoch správať.

MVC bolo dizajnované tak, aby ponúklo vývojárom možnosť prispôsobiť si framework podľa svojich požiadaviek, resp. požiadaviek aplikácií. Preto je im zdrojový kód ASP.MVC k dispozícii na neustály rozvoj pod OSI licenciou. [10], [11]

4.2 MVC – Model-View-Controller

MVC je softwareová architektúra, ktorá rozdeľuje dátový model aplikácie, užívateľské rozhranie a riadiacu logiku do troch nezávislých komponent tak, že modifikácia niektorej z nich má len minimálny vplyv na ostatné.

Motiváciou pre používanie tejto architektúry je snaha o oddelenie logiky od výstupu. Rieši teda problematiku tzv. „špagetového“ kódu, v ktorom sú v jednom súbore pomiešané logické operácie a zároveň renderovanie výstupu. Súbor teda obsahuje databázové požiadavky, logiku a rôzne poprehadzované HTML tagy. Všetko je do seba zamotané ako špagety.

Kód sa samozrejme ťažko udržiava, nieto ešte rozširuje. Pri jeho prezeraní sa strácame, pretože si IDE nevie dať rady so zvýrazňovaním tagov a stromová štruktúra kódu je tak komplikovaná, že nemáme tušenia, čo daná funkcionalita robí. Našim cieľom je, aby zdrojový kód s logikou vyzeral ako štandardný zdrojový kód v C# a výstup vyzeral ako HTML stránka s čo najmenšou prímesou ďalšieho kódu.

Celá aplikácia je rozdelená na komponenty 3 typov, ktoré sú uložené podľa svojej kategórie spoločne na jednom mieste. Jedná sa o Modely, Viewy (slovensky pohľady) a Controllery.

4.2.1 Model

Model obsahuje aplikačnú logiku a všetko, čo s ňou súvisí. Môže sa jednať o dátové štruktúry, databázové dotazy, business logiku atď. V našom prípade sa teda jedná o klasické C# triedy. Dôležitou myšlienkou je, že model vôbec nevie o výstupe – netuší, ako bude jeho výstup spracovaný alebo akým spôsobom budú jeho dáta zobrazené. Jeho funkcia spočíva jedine v prijatí parametrov

zvonku a vrátení dát naspäť von. Parametrami v tomto prípade myslíme parametre jednotlivých metód v triede modelu.

V našej aplikácii budeme používať Entity Framework, ktorý nám namapuje jednotlivé modely na databázové tabuľky. Inštancie jednotlivých modelov budú obsahovať atribúty databázových tabuliek a vďaka tejto vlastnosti bude naša aplikácia nezávislá od konkrétnej použitej databázy. Je to obrovská výhoda, pretože pri presune na iný databázový systém nie je nutné meniť v programe takmer nič okrem pripojovacieho reťazca identifikujúceho konkrétny databázový server.

4.2.2 View

View je časť aplikácie zodpovedná za zobrazovanie výstupu užívateľovi. Vo svojej podstate sa jedná o HTML šablónu doplnenú tagmi syntaxe Razor, ktorá umožňuje do šablóny vkladať premenné, cykly a podmienky. Vďaka tomu je možné doplniť do statickej šablóny konkrétne hodnoty.

Viewov máme v aplikácii veľa, keďže pre každú funkcionalitu potrebujeme mierne odlišnú štruktúru výstupu. Napríklad pre funkcionality s entitou Predmet budú existovať samostatné viewy: Predmet_Pridat, Predmet_Upravit, Predmet_Zobrazit atď.

Šablóny je samozrejme možné vkladať do seba, aby sme spoločné prvky stránky nemuseli kopírovať do každého viewu. Jedná sa napríklad o layout stránky s navigáciou, šablónu stránky so zoznamom a podobne. Takýmto vkladaným viewom hovoríme *„Partial Views“*.

View nie je len šablóna, ale je zároveň aj zobrazovač výstupu. Znamená to, že obsahuje aspoň minimálne množstvo logiky potrebnej pre sformátovanie dát pre výpis, ale aj validačné mechanizmy pred odoslaním formuláru, ktoré sa však opierajú o definície jednotlivých atribútov v modeli.

Podobne ako Model, ani View netuší, odkiaľ mu dáta na zobrazenie prišli. Stará sa len o ich zobrazenie užívateľovi.

4.2.3 Controller

Controller je tým chýbajúcim článkom, ktorý všetko prepája dohromady. Jedná sa o akéhosi prostredníka, s ktorým komunikuje užívateľ, model aj view. Drží teda celý systém pohromade a komponenty prepojuje.

Controller spracováva prichádzajúce požiadavky, užívateľský vstup a interakcie s webstránkou a k nim spúšťa príslušnú aplikačnú logiku.

4.2.4 Životný cyklus stránky ASP.NET MVC

Životný cyklus stránky štartuje užívateľ, ktorý zadá do prehliadača URL adresu webu s nejakými parametrami, ktorými aplikácii dá najavo, ktorú stránku si praje zobraziť. Príklad nájdeme v ukážke 4.1.

```
http://www.stranka.sk/Subjects/Details/2
```

Ukážka kódu 4.1: Príklad užívateľského požiadavku na ASP.NET MVC

Požiadavku ako prvý zachytí tzv. *„router“*, ktorý podľa parametrov v URL adrese zistí, ktorý controller, ktorú akciu (metódu) v ňom a s akým parametrom má zavolať. Štandardným vzorcom URL v ASP.MVC je totiž:

```
http://www.stranka.sk/{Controller}/{Action}/{Id}
```

Ukážka kódu 4.2: Rozparsovanie požiadavku 4.1

V našom prípade sa teda volá controller *Subjects*, jeho metóda *Details* a identifikátorom konkrétnej inštancie (Id) je *2*.

Controller po zavolaní spustí konkrétnu akciu, v našom prípade detail predmetu s Id = 2. Zavolá model, ktorý predmet s daným Id nájde v databáze a vráti jeho údaje. Následne zavolá ďalšiu metódu modelu, ktorá napríklad spočíta, koľko žiakov má daný predmet zapísaný. Tieto údaje si controller ukladá do premenných. Nakoniec pre danú akciu vyrenderuje View, ktorému sú odoslané premenné s príslušnými dátami. Controller teda poslúchol užívateľa a podľa zadaných parametrov zaobstaral dáta od modelu, ktoré ďalej posunul Viewu.

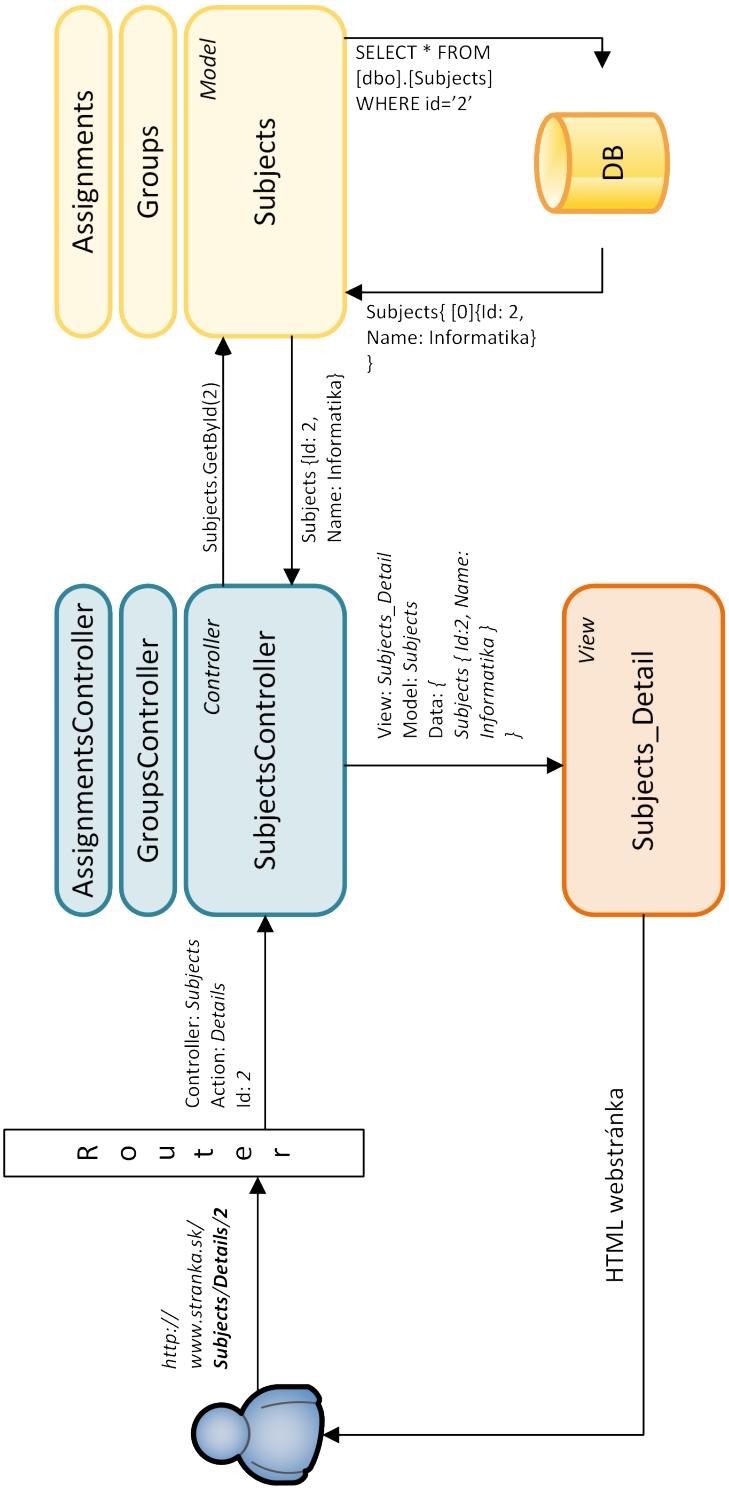
View prijme tieto dáta od controlleru a vloží ich do pripravenej šablóny. Hotová stránka sa potom zobrazí užívateľovi, ktorý často o tejto celej kráse ani len netuší. Vyššie popísaný životný cyklus nájdete na obrázku 4.1.

4.3 Microsoft Graph

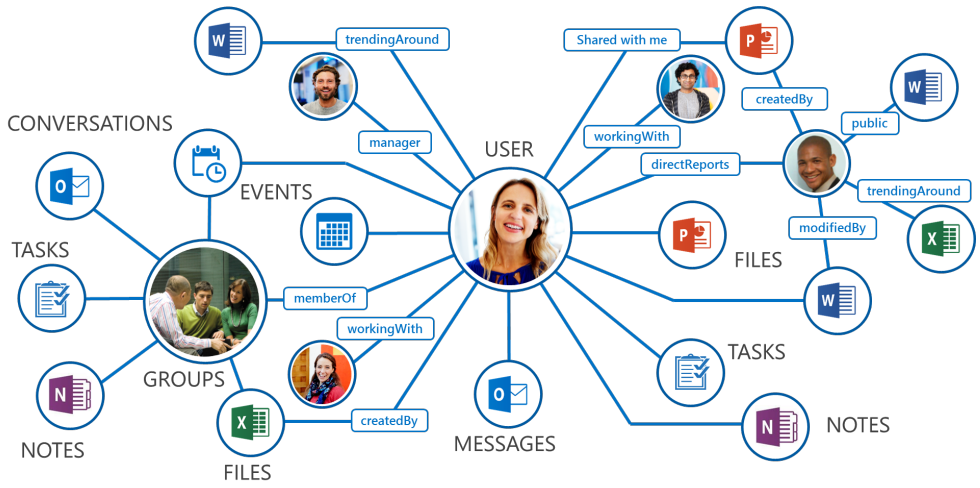
Microsoft Graph je RESTful API pre vývojárov aplikácií, ktoré komunikujú s cloudom Office 365. Vzniklo v roku 2015, takže je ešte stále pomerne novým a vyvíjajúcim sa projektom. Pred jeho existenciou bolo pre komunikáciu s jednotlivými produktmi Office 365[4] nutné naučiť sa používať API každého jedného produktu.

Microsoft Graph však funguje ako univerzálna brána do Office 365, ktorá sprostredkúva užívateľské požiadavky jednotlivým *„resourceom“* (slovensky zdrojom). Graph si môžeme v tomto prípade skutočne predstaviť ako graf s vrcholmi, v ktorých sa nachádzajú jednotlivé entity (napríklad Užívateľ, Kalendár, Míting a pod.) a hranami, ktoré predstavujú vzťahy medzi jednotlivými entitami (napríklad Anna Nováková je menežérom Jozefa Starého alebo Kristián Malý má Míting o 14.00 v Pracovnom kalendári). Vďaka takémuto grafovému prístupu je pre vývojárov oveľa ľahšie pristupovať komplexne ku ponúkaným zdrojom a vyťažiť z ponúkaných dát absolútne maximum. Takúto databázovú schému môžeme označiť ako *„user-centric“*, čiže zameranú na užívateľa a všetko s ním súvisiace.

[4]Napríklad SharePoint, OneDrive, Skype, Word a podobne.



Obr. 4.1: Ukážka požiadavku v ASP.NET MVC



Obr. 4.2: User-centric databáza Microsoft Graph [12]

Požiadavky na Graph API sú pomerne jednoduché a priamočiare. Ku komunikácii sa využíva štandard `REST`, ktorý je určený ku uskutočňovaniu tzv. `CRUD`[5] operácií pomocou HTTP požiadavkov ako `GET`, `POST`, `PUT` a `DELETE`. Výhodou je, že v jednej požiadavke môžeme požadovať niekoľko resourceov, ktoré s užívateľom súvisia. Príkladom jednoduchej požiadavky môže byť: *„ukáž môj kalendár“* v ukážke 4.3:

```
GET https://graph.microsoft.com/v1.0/me/calendar
```

Ukážka kódu 4.3: Požiadavka na Microsoft Graph

Odpoveďou Microsoft Graphu potom je:

```
HTTP/1.1 200 OK
Content-type: application/json
{
    "@odata.context": "https://graph.microsoft.com/v1.0/
        ↪ $metadata#me/calendars/$entity",
    "@odata.id": "https://graph.microsoft.com/v1.0/users('...')/
        ↪ calendars('...')",
    "id": "AAMkAGI2TGuLAAA=",
    "name": "Calendar",
    "color": "auto",
    "changeKey": "nfZyf7VcrEKLNoU37KWlkQAAAOx0+w==",
    "canShare":true,
    "canViewPrivateItems":true,
    "canEdit":true,
```

[5]Create, Read, Update, Delete

```
"owner":{
    "name":"Samantha Booth",
    "address":"samanthab@adatum.onmicrosoft.com"
} }
```

Ukážka kódu 4.4: Odpoveď Microsoft Graphu na požiadavku 4.3

Zároveň dokážeme Microsoft Graph využiť pre autentifikáciu a autorizáciu užívateľov v našej aplikácii. Pre prácu so zdrojmi v rámci graphu je nutné autentifikovať sa voči serveru takzvaným *„access tokenom“*, vďaka ktorému server vie, či sme autorizovaní k danému zdroju pristúpiť. Na tento účel sa využíva OpenID Connect a protokol OAuth 2.0.

4.3.1 Autentifikácia a autorizácia

Aby sme mohli z Microsoft Graphu získať hociaké informácie, je potrebné sa v prvom rade voči nemu autentifikovať. Toto sa deje autentifikáciou s `Azure Active Directory` (Azure AD), ktorý funguje ako služba pre overovanie identity v rámci cloudových služieb od Microsoftu. `Access token`, ktorý vráti autentifikačný endpoint ako odpoveď obsahuje zoznam práv (anglicky *claims*), ktoré obsahujú informácie o našej aplikácii a o jej povoleniach a rozsahu prístupu ku konkrétnym resourceom. Aby aplikácia skutočne `access token` aj dostala, musí byť schopná autentifikovať sa voči `Azure AD` a musí dostať autorizačné oprávnenie na konkrétne zdroje buď od užívateľa (v rozsahu jeho užívateľských oprávnení) alebo od administrátora (v rozsahu na celú organizáciu).

4.3.2 Access Token v skratke

`Access token`, ktorý vydáva `Azure AD` je reťazec formátu JSON Web Tokens (JWT) zákodovaný algoritmom `base 64`. Obsahuje informácie (*claims*), ktoré používajú rôzne webové API, ako napríklad Microsoft Graph na validáciu volajúcej aplikácie a na overenie oprávnení na volanú operáciu. Pri volaní Microsoft Graphu sa `access token` pripája ku autorizačnej hlavičke v HTTP požiadavke. Nahrádza tak v komunikácii s Graphom užívateľské meno a heslo[6], ktoré si aplikácia nemusí uchovávať. `Access token` má iba obmedzenú platnosť a pre správny chod aplikácie musí byť pravidelne obnovovaný. [13]

4.3.3 Povolenia v Microsoft Graphe

Microsoft Graph chráni jednotlivé zdroje pomocou kontroly povolenia pre daný zdroj. Aplikácia si vždy pred prvým použitím vypýta od užívateľa resp.

[6]Login a heslo zadáva užívateľ na prístupovej bráne cloudu, nie konkrétnej aplikácie.

administrátora povolenie používať dané zdroje a až následne s týmto povolením k nim má prístup. Existujú dva typy povolení, ktoré môže aplikácia vyžadovať a používať:

delegované Delegované povolenia sú používané aplikáciami, ktoré vo svojom konaní zastupujú používateľa – teda konajú v jeho mene. Tieto povolenia môže odsúhlasiť užívateľ na prihlasovacej obrazovke alebo administrátor, ak sa jedná o povolenia s väčším dopadom. Aplikácia však smie pristupovať ku daným zdrojom pod užívateľskou identitou len kým je užívateľ prihlásený.

Príkladom delegovaného povolenia môže byť `Calendars.Read`, ktoré dovoľuje aplikácii čítať s užívateľského kalendára: *„Zobraz udalosti v mojom kalendári medzi 23.4.2017 a 29.4.2017.“*

```
GET /me/calendarView?startDateTime=2017-04-23T00:00:00&
    ↪ endDateTime=2017-04-29T00:00:00
```

Ukážka kódu 4.5: Ukážka delegovaného povolenia

aplikačné Aplikačné povolenia sú používané aplikáciami, ktoré nepotrebujú alebo z nejakých príčin nechcú vystupovať ako prihlásený užívateľ. Môže sa jednať o rôzne služby na pozadí, prípadne o aplikácie, ktoré potrebujú mať dosah na viacerých užívateľov (aj napriek tomu, že prihlásený užívateľ by na to za normálnych okolností oprávnenie nemal). Aplikačné povolenia môže udeliť iba administrátor pomocou tzv. *„admin consent“* (čiže administrátorského súhlasu).

Príkladom aplikačného povolenia môže byť `Group.Read.All`, ktoré dovoľuje aplikácii čítať vo všetkých Azure AD skupinách: *„Nájdi všetky skupiny, ktorých názov začína na 'Sales'“*

```
GET /groups?$filter=startswith(displayName,'Sales')
```

Ukážka kódu 4.6: Ukážka aplikačného povolenia

Efektívne povolenia sú skutočné povolenia, ktoré aplikácia pri komunikácii s Microsoft Graphom dostane. Pri delegovaných povoleniach sa jedná o minimálny prienik s užívateľskými povoleniami. Znamená to, že aplikácia konajúca v mene užívateľa nemôže použiť vyššie povolenia, ako má samotný užívateľ (aj keď na ne môže mať súhlas). Naopak, pri aplikačných povoleniach sa efektívnymi povoleniami stáva maximálna množina povolení odsúhlasených administrátorom. [14]

4.3.4 Registrácia aplikácie a získanie povolení

Pred spustením aplikácie komunikujúcej s Microsoft Graphom je potrebné ju zaregistrovať na developerskom portáli app.dev.microsoft.com. Následne je aplikácii vygenerované jedinečné identifikačné číslo `Application Id` a tajné

heslo `Application Secret`. Tieto dva údaje sú používané aplikáciou pri nadviazaní autentifikačného spojenia s Azure AD.

Zároveň je nutné na tomto portáli nastaviť všetky povolenia, ktoré si bude aplikácia vyžadovať ku jednotlivým resourceom – či už delegované, alebo aplikačné. [15]

4.3.5 Autentifikačný proces

OpenID Connect je jednoduchá identitná vrstva postavená nad protokolom OAuth 2.0. OAuth 2.0 definuje základné mechanizmy pre získanie access tokenov ku chráneným resourceom, ale chýbajú mu štandardné metódy pre poskytovanie informácií o identite užívateľa. OpenID Connect implementuje autentifikáciu ako rozšírenie ku autorizačnému procesu protokolu OAuth 2.0 a poskytuje informácie o koncovom užívateľovi prostredníctvom `id_token`, ktorý overuje identitu užívateľa a obsahuje o ňom základné informácie. [16]

```
// Zaciatok nespracovaneho id_token-u
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsIng1dCI6ImtyaU1QZG1Cd...

// Zaciatok spracovaneho a dekodovaneho id_token-u
{
    "name": "John Smith",
    "email": "john.smith\@gmail.com",
    "oid": "d9674823-dffc-4e3f-a6eb-62fe4bd48a58"
    ...
}
```

Ukážka kódu 4.7: Časť id_token-u

Teraz si ukážeme autentifikáciu pomocou OpenID Connect krok po kroku. Priebeh tohto procesu nám tak isto znázorňuje obrázok 4.3.

1. Užívateľ klikne v aplikácii na link odkazujúci na zabezpečenú sekciu.

2. Aplikácia vygeneruje autorizačnú URL, na ktorú presmeruje užívateľa.

3. Užívateľ zadá na zabezpečenej stránke poskytovateľa identít (v našom prípade Microsoftu) svoje prihlasovacie údaje.

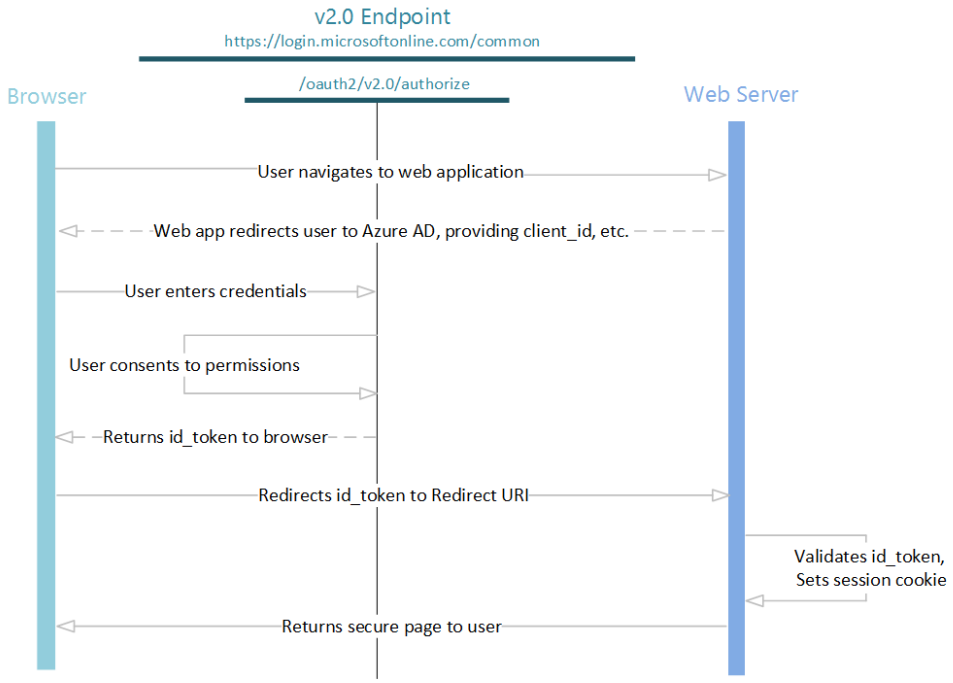
4. Po ich úspešnom overení je užívateľ vyzvaný na súhlas s povoleniami, ktoré si aplikácia pýta. Tento bod nastane len ak administrátor organizácie neodsúhlasil paušálne povolenia pre celú organizáciu.

5. Na základe povolenia vydá poskytovateľ identít `id_token`, v ktorom sa nachádzajú základné informácie o koncovom užívateľovi a pošle ho naspäť do prehliadača užívateľa.

6. Prehliadač presmeruje užívateľa aj s novovydaným `id_token`om naspäť na stránku aplikácie.

7. Aplikácia overí pravosť `id_token`u, uloží si informácie do session cookie a zobrazí užívateľovi zabezpečenú stránku.

Po prechode celým procesom je užívateľ úspešne prihlásený do aplikácie a tá o ňom vďaka `session cookie` má základné informácie. Momentálne však užívateľ ani aplikácia ešte stále nemajú `access token` potrebný pre pracovanie s resourceami.



Obr. 4.3: Autentifikačný proces OpenID Connect [16]

4.3.6 Autorizačný proces

Ku získaniu `access token`u musí aplikácia vyvolať autorizačný proces protokolu OAuth 2.0. Podľa typu aplikácie a typu povolení, ktoré sa aplikácia snaží získať môže vyvolať jeden z nasledujúcich 4 procesov:

OAuth 2.0 authorization code flow Tento proces používajú najmä aplikácie nainštalované na zariadeniach ako napríklad mobilné alebo desktopové aplikácie, ktoré často potrebujú pristupovať ku prostriedkom nejakého webového API v mene prihláseného užívateľa. Počas prihlasovania sa dostane aplikácia okrem iného aj `authorization_code`, ktorý reprezentuje povolenie ap-

likácie pristupovať ku back-endovým službám v mene prihláseného užívateľa. Aplikácia môže následne tento autorizačný kód vymeniť za `access token`, ktorý môže použiť na autorizáciu svojich požiadavkov a `refresh token`, vďaka ktorému môže získať nový `access token` po vypršaní platnosti starého. [17]

OAuth 2.0 implicit flow Tento proces používajú najmä tzv. *Single Page* aplikácie, v ktorých väčšinu obslužnej činnosti vykonáva Javascript (AngularJS, Ember.js, React.js a iné). Hlavnou výhodou tohto procesu je, že aplikácia si k získaniu `access token`u nepotrebuje vymieňať prístupové údaje cez svoj back-endový server. Vďaka tomu dokáže prihlásiť užívateľa, uložiť si jeho session a získavať `access token`y pre webové API pomocou Javascriptu len zo strany klienta (z prehliadača) bez nutnosti kontaktovať back-end. [18]

OAuth 2.0 client credentials flow Tento proces používajú aplikácie, ktoré sa voči Graphu identifikujú nie užívateľskou, ale vlastnou identitou. Využíva sa v spojení s aplikačnými povoleniami, vďaka ktorým aplikácia nekoná v mene konkrétneho prihláseného užívateľa, ale vo svojom vlastnom mene. Vďaka týmto povoleniam môže pracovať nad viacerými užívateľmi a pristupovať ku ich resourceom bez toho, aby bol nejaký užívateľ vôbec prihlásený. Tento typ aplikácií zväčša beží ako služby na pozadí (anglicky *daemons*). [19]

OAuth 2.0 on-behalf-of flow Tento proces sa používa v prípadoch, v ktorých aplikácia, ktorá sa autentifikuje voči jednému webovému API sa potrebuje následne cez toto API identifikovať voči ďalšiemu webovému API. Hlavnou myšlienkou je propagácia delegovanej užívateľskej identity a povolení cez reťaz požiadavok. [20]

V prípade našej aplikácie potrebujeme dostať `access token` pre identitu aplikácie, pretože potrebujeme pracovať s viacerými užívateľmi súčasne (napríklad pri zadávaní úlohy), preto zvolíme **client credentials flow**. Teraz si krok po kroku ukážeme priebeh tohto procesu, ktorý verne znázorňuje aj obrázok 4.4.

1. *Administrátor organizácie je presmerovaný na stránku *admin consent*.

2. *Administrátor organizácie skontroluje a udelí požadované povolenia aplikácii.

3. *Administrátor organizácie je presmerovaný naspäť na stránku aplikácie. Odteraz si už aplikácia na dané povolenia nebude pýtať súhlas administrátora.

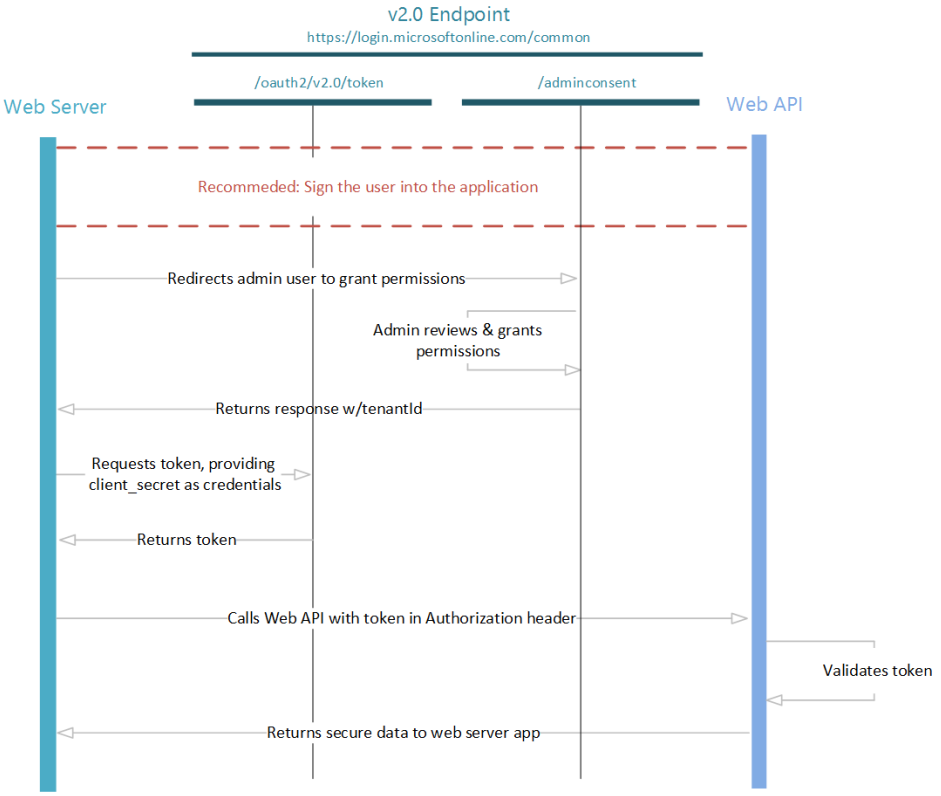
4. Aplikácia sa pripojí na endpoint vydávajúci `access token`y a prezentuje sa svojím `Application Id` a `Application Secret`[7]. Vďaka nim poskytovateľ identít overí, že sa skutočne jedná o zaregistrovanú aplikáciu.

5. Poskytovateľ identít na základe overenej identity aplikácie a zaregistrovaného administrátorského súhlasu s povoleniami vydá `access token` obsahujúci tieto povolenia.

6. Aplikácia môže volať jednotlivé zdroje Microsoft Graphu, ku ktorým pripája vygenerovaný `access token`.

Výsledkom úspešného prechodu týmto procesom je aplikácia, ktorá sa môže pripájať ku vopred odsúhlaseným resourceom bez ďalšieho pýtania si povolenia. Je dôležité podotknúť, že administrátor organizácie môže toto povolenie aplikácii kedykoľvek zrušiť cez portál Azure Active Directory. Body označené hviezdičkou (*) vykonáva administrátor organizácie len raz, pri každom ďalšom spojení s autorizačným serverom sa začne proces vykonávať až od bodu číslo 4.

[7]Pri iných autorizačných procesoch sa `Application Secret` nepoužíva, pretože sa aplikácia neprihlasuje vlastnou identitou.



Obr. 4.4: OAuth 2.0 Client Credentials Flow [19]

Kapitola 5

Implementácia

V tejto časti sa budeme venovať vytvoreniu aplikácie **Cloud Home Work** (v skratke **Cloud HW**), na ktorej si ukážeme základné princípy komunikácie s cloudom Office 365 a zostavíme tak *„proof of concept“*, teda funkčnú aplikáciu dokazujúcu, že dané technológie môžeme pre túto problematiku využiť.

5.1 Beh aplikácie

V tejto stati si v krátkosti rozoberieme proces spracovania požiadavkov, prichádzajúcich na aplikáciu, od ktorého sa neskôr budú odvíjať ďalšie state vysvetľujúce jednotlivé funkcionality.

1. **Prijatie prvého požiadavku na aplikáciu**

   V súbore `Global.asax` sú objekty typu `Route` pridané do `RouteTable` – čiže do routovacej tabuľky.

```
void Application_Start(object sender, EventArgs e)
{
    RegisterRoutes(RouteTable.Routes);
}

public static void RegisterRoutes(RouteCollection routes)
{
        routes.MapRoute(
        name: "Default",
        url: "{controller}/{action}/{id}",
        defaults: new { controller = "Home", action = "Index
            ↪ ", id = UrlParameter.Optional }
    );
}
```

Ukážka kódu 5.1: Registrácia v routovacej tabuľke

2. **Routing**

   Modul `UrlRoutingModule` použije prvý vhodný objekt typu `Route` z routovacej tabuľky na vytvorenie `RouteData` objektu, ktorý následne použije na vytvorenie objektu typu `RequestContext (IHttpContext)`. Ako sme už spomenuli v stati 4.2.4, routing je zodpovedný za rozparsovanie užívateľského požiadavku a zavolanie príslušného controlleru.

3. **Vytvorenie MVC request handleru**

   Objekt `MvcRouteHandler` vytvorí inštanciu triedy `MvcHandler` a pošle jej inštanciu `RequestContext` vytvorenú v predchádzajúcom kroku.

4. **Vytvorenie controlleru**

   `MvcHandler` použije inštanciu `RequestContext` na identifikáciu objektu typu `IControllerFactory` (typicky sa jedná o inštanciu triedy `DefaultControllerFactory`) na vytvorenie inštancie konkrétneho controlleru.

5. **Spustenie controlleru**

   `MvcHandler` zavolá metódu `Execute` na vytvorenom controlleri.

6. **Vyvolanie akcie**

   Väčšina controllerov sú zdedené triedy od rodičovskej triedy `Controller`. Controllery, ktorých sa to týka, obsahujú objekt `ControllerActionInvoker`, ktorý rozhoduje, ktorú akčnú metódu má daný controller zavolať a následne ju aj zavolá.

7. **Spustenie výsledku**

   Typická akčná metóda zväčša prijíme užívateľský vstup, pripraví si k nemu vhodné výstupné dáta a následne spustí výsledok vrátením návratovej hodnoty. Vstavané typy návratových hôdnot, ktoré môžu byť spustené sú: `ViewResult` (ktorý vyrenderuje View a je najčastejšie používaný), `RedirectToRouteResult`, `RedirectResult`, `ContentResult`, `JsonResult` a `EmptyResult`. [21]

5.2 Asynchrónne programovanie v ASP.NET

Technológia ASP.NET ponúka jednu nespornú výhodu oproti bežným skriptovacím jazykom ako sú napríklad PHP alebo Javascript a je ňou používanie asynchrónneho programovania, ktoré plne využíva procesorovú paralelizáciu. Hodiť sa nám môže v prípadoch, v ktorých bude naša aplikácia pracovať s veľkým množstvom objektov, napríklad užívateľmi, u ktorých bude vykonávať rovnakú operáciu. Takáto asynchrónna metóda používa namiesto štandardnej návratovej hodnoty *T* hodnotu `<Task<T>>`. [22]

Skôr, ako si ukážeme na krátkej ukážke výhodu takéhoto asynchrónneho programovania, predstavíme si dve kľúčové slovíčka, vďaka ktorým tento celý princíp funguje.

async – Týmto kľúčovým slovom označíme v deklarácii metódu, v ktorej tele aspoň raz použijeme kľúčové slovíčko **await**, vďaka ktorému budeme aktuálnu metódu implementovať ako kompozíciu behu asynchrónneho workflow zostavenú z volania iných asynchrónnych funkcií. Tým zmeníme štandardný spôsob prekladu tejto metódy na stavový automat, ktorý bude zaisťovať spúšťanie častí kódu za a medzi jednotlivými volaniami **await** ako callback (continuation) jednotlivých asynchrónnych volaní.

await – Toto kľúčové slovo uvedieme pred asynchrónne volanie, u ktorého chceme, aby sa ďalší beh kódu umiestneného za **await** naplánoval a spustil až po dokončení volanej operácie. V dôsledku toho budú v kóde za **await** už k dispozícii výsledky vykonanej asynchrónnej metódy. [23]

Teraz si ukážeme použitie týchto slovíčok na malej ukážke:

```
1  public async Task<User> getUserById(GraphServiceClient
       ↪ graphClient, string userId)
2  {
3      return await graphClient.Users[userId].Request().GetAsync();
4  }
5
6  public async Task<List<User>> ProccessUsers(GraphServiceClient
       ↪ graphClient, List<int> userIdList)
7  {
8    var users = new List<User>(); // prazdny zoznam uzivatelov
9    var taskList = new List<Task<User>>(); // prazdny zoznam uloh
10   foreach(string uid in userIdList){
11         taskList.Add( getUserById(graphClient, uid) ); //
               ↪ pridame novu ulohu
12   }
13   // tu mozeme vykonavat cinnosti nesuvisiace s uzivatelmi
14   // ... ...
15   while (taskList.Count > 0)
16   {
17    var task = await Task.WhenAny(taskList); // ak je hociktora z
         ↪  uloh hotova
18    users.Add(await task); // prevezmi vysledok dokoncenej ulohy
19    taskList.Remove(task); // odstran dokoncenu ulohu zo zoznamu
20   }
21
22   return users;
23 }
```

Ukážka kódu 5.2: Použitie asynchrónnej metódy

Všimnime si, že v riadku číslo 9 vytvárame prázdny *„pool“* pre budúce asynchrónne úlohy. Pridávaním úloh v riadku 11 sa dané úlohy automaticky spúšťajú a vykonávajú nezávisle od seba. Medzi zavolaním asynchrónnej metódy (úlohy) a vyvolaním jej výsledku pomocou `await` môže program pokračovať v behu, ak narába s premennými nesúvisiacimi s výsledkami danej úlohy.

Ak by sme chceli počkať na dokončenie jednej konkrétnej úlohy, zavoláme podobne, ako v riadku číslo 3 `await Úloha()`. V tom momente sa kontext prepne naspäť do volajúcej metódy až do momentu, kým nie je výsledok danej úlohy pripravený, čo je maximálne výhodné. V našom prípade čakáme v riadku č. 17 na dokončenie hociktorej z úloh, aby sme jej výsledok (teda objekt typu `User` mohli vložiť do nášho výsledného zoznamu). Podobne by sme mohli pomocou direktíva `Task.WhenAll(taskList)` počkať naraz na dokončenie všetkých úloh a až tak ich výsledky spracovať. V našom prípade nám však nezáleží, v akom poradí budú užívatelia v zozname uložení a preto je rýchlejšie použiť direktívum `Task.WhenAny(taskList)`, ktoré nám ako návratovú hodnotu vráti prvú dokončenú úlohu v zozname.

5.3 Microsoft Graph .NET SDK

`Microsoft Graph .NET Software Development Kit` je sada knižníc, ktoré programátorom uľahčujú komunikáciu s cloudovým prostredím Office 365. Keďže naša aplikácia bude úzko s cloudom spolupracovať, použijeme na komunikáciu práve metódy ponúkané touto knižnicou. Jej nespornou výhodou je, že Microsoft Graph ako univerzálna brána do cloudu Office 365 komunikuje so všetkými zdrojmi, ktoré cloud ponúka. V opačnom prípade by sme museli používať knižnice ku každému zo zdrojov zvlášť.

Pre úspešné zavedenie do aplikácie je potrebné v prvom rade nainštalovať balíček `Microsoft.Graph` a mať zaregistrovanú aplikáciu na developerskom portáli apps.dev.microsoft.com.

Aby boli požiadavky zo strany aplikácie spracované, musí sa voči serverom Microsoftu autentifikovať a autorizovať. Túto funkcionalitu však SDK neponúka, preto si musíme pomôcť sami – na autentifikáciu použijeme už zmienenú knižnicu `OpenID Connect`. Na autorizáciu pomocou `OAuth 2.0 client credentials flow` použijeme zabudovanú knižnicu `Microsoft.Identity`.

5.3.1 Autentifikácia

Autentifikačný proces je nastavovaný v súbore **Startup.Auth.cs**. V rámci neho je nakonfigurované prostredie `OpenID Connect`, ktoré v prípade potreby overuje užívateľa voči Microsoft Graphu. Na uchovávanie informácií o užívateľovi používa zašifrované cookie súbory. Po úspešnom priebehu autentifikácie sa zavolá metóda `Application_PostAuthenticateRequest`, ktorá prihlásenému užívateľovi do jeho *claims* (pozri stať 4.3.1) priradí roly z lokálnej databázy.

5.3.2 Autorizácia

Aby sa ku určitým metódam alebo dokonca až ku celým controllerom nedostali neprihlásení užívatelia, dekorujú sa tieto metódy, resp. controllery anotáciou `[Authorize]`. Vstavaná implementácia tohto atribútu zabezpečí kontrolu s používanou autentifikačnou knižnicou a na základe prítomnosti/neprítomnosti *claims* užívateľa do chránenej sekcie vpustí alebo nevpustí.

Ak však chceme niektoré metódy sprístupniť len užívateľom s konkrétnym povolením (teda priradenou konkrétnou rolou), musíme použiť vlastný autorizačný atribút `[AccessDeniedAuthorize(Roles=(...)]`, ktorý implementujeme ako rozšírenie štandardného autorizačného atribútu `AuthorizeAttribute`.

5.3.3 Získavanie Access Tokenov

Implementácia procesu získavania `access token`ov sa nachádza v metóde `GetUserAccessTokenAsync`, ktorá na základe pravidiel pre `OAuth 2.0 client credentials flow` komunikuje s Microsoft Graphom. Túto metódu volá trieda `SDKHelper` pri zostavovaní každého požiadavku na Microsoft Graph. Aby sa aplikácia nemusela vždy na `access token` pýtať Graphu, používa tzv. *„Token Cache“*, ktorá v sebe ukladá `access token`y pre jednotlivých prihlásených užívateľov.

Pri vývoji bola implementácia tejto funkcionality asi najťažšia, pretože dokumentácia ku použitým triedam nie je na dobrej úrovni. Tak isto neexistuje štandardná implementácia *Token Cache* a preto bola v tejto ukážke použitá `SessionTokenCache`, ktorá nie je nutne najbezpečnejšou variantou. Pre implementáciu bezpečnejšej, databázovej *Token Cache* by bolo potrebné inštalovať separátny server, čo sa k rozsahu danej práce nevyplatilo.

5.3.4 Požiadavky na Microsoft Graph

Ako sme si už v stati 4.3 povedali, požiadavky na Microsoft Graph sú veľmi jednoduché a priamočiare. SDK nám v tomto pomáha ešte viac, pretože programátorov oslobodzuje od formulácie `REST`-ových požiadaviek a od parsovania odpovedí Microsoft Graphu do zmysluplnej podoby. Toto všetko za nás urobí SDK. Preto potom požiadavku

```
GET https://graph.microsoft.com/v1.0/me/calendar
```

Ukážka kódu 5.3: Požiadavka na Microsoft Graph

môžeme poľahky vďaka SDK zapísať ako:

```
var mycalendar = await graphClient.Me.Calendar.Request().
    ↪ GetAsync();
```

Ukážka kódu 5.4: Požiadavka na Microsoft Graph v SDK

Návratou hodnotou je v tomto prípade objekt `Microsoft.Graph.Calendar`, ktorý pre nás vytvorilo SDK z odpovede serveru formulovanej vo formáte `JSON`.

5.4 Modely

Modely v našej aplikácii slúžia ako reprezentanti entít a služieb, s ktorými aplikácia operuje. Aplikácia používa tri typy modelov: entitné modely, modely zastupujúce jednotlivé cloudové služby a tzv. *view modely*.

5.4.1 Entitné modely

Entitné modely reprezentujú jednotlivé lokálne databázové tabuľky v prostredí ASP.NET MVC skrz `Entity Framework`. Vďaka nim môže aplikačná logika pracovať s položkami v databázových tabuľkách bez jediného SQL príkazu. Entitné modely okrem samotných atribútov v sebe nesú aj dôležité popisné znaky jednotlivých atribútov ako sú napríklad zobrazovaný názov atribútu, či jeho validačné pravidlá. Tieto popisné znaky je výhodné uložiť v modeli, pretože sa použijú pri každom kontakte s daným modelom (hodí sa, ak viacero formulárov používa ten istý model). V ukážke kódu 5.5 vidíme štandardný atribút `Title`, ktorý je dekorovaný dátovou anotáciou `[Display(-...)]`, definujúcou zobrazovaný názov daného atribútu vo formulároch.

```
[Display(Name = "Title", ResourceType = typeof(Resources.
    ↪ Resource))]
    public string Title { get; set; }
```

Ukážka kódu 5.5: Dátová anotácia v entitnom modele

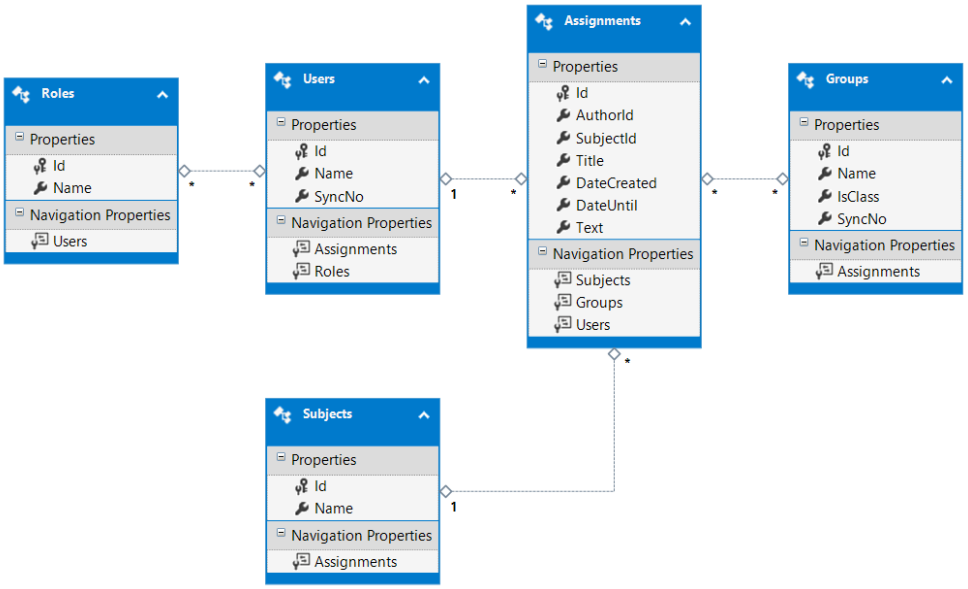
V tomto konkrétnom prípade serveru hovoríme: „*Nájdi v triede* `Resources.Resource` *položku s názvom 'Title' a dosaď jej hodnotu za zobrazovaný názov tohto atribútu.*“ Prečo nie je názov atribútu natvrdo nakódovaný? Z jednoduchého dôvodu – pretože si pripravujeme pôdu pre podporu viacerých zobrazovacích jazykov.

5.4.2 Grafové modely

Grafové modely v našej aplikácii obsahujú metódy pre prácu s Microsoft Graphom. Sú rozdelené podľa resourceov, ku ktorým pristupujú. V našom prípade sa jedná o **FilesService**, **GroupsService** a **MailService**.

Peknou a jednoduchou ukážkou práce s Graphom môže byť asynchrónna metóda `GetMembers` na ukážke 5.6, ktorá vracia zoznam členov danej skupiny[8]. Nakoľko je Graph „zvyknutý“ vracať enormné množstvá objektov ako odpoveď na požiadavku, rozhodli sa tvorcovia pri implementácii jeho API

[8]Poznámka: Niektoré riadky sme pre šetrenie miesta zámerne vynechali.



Obr. 5.1: Entitné modely

použiť stránkovanie. Preto iba jedna požiadavka na konkrétny resource nestačí k tomu, aby sme dostali všetky položky. Ku zisteniu, či existuje aj nasledujúca stránka slúži atribút `NextPageRequest`, ktorý metóde indikuje, či má pokračovať v dotazovaní na Graph, alebo má vyskočiť z cyklu.

```
public async Task<List<User>> GetMembers(GraphServiceClient
    ↪ graphClient, string id)
{
 List<User> items = new List<User>();
 var members = await graphClient.Groups[id].Members.Request().
    ↪ GetAsync();
 while (true) {
   foreach (DirectoryObject mem in members)
     {
        if (mem is User) {
           User member = mem as User;
           items.Add(member);
        }
     }
     if (members.NextPageRequest != null){
          members = await members.NextPageRequest.GetAsync();
     }else{
          break;
     }
```

```
  }
  return items;
}
```

Ukážka kódu 5.6: Metóda GetMembers

V nasledujúcich odstavcoch si predstavíme jednotlivé metódy, implementované v grafových modeloch.

5.4.2.1 FilesService

Model **FilesService** obsahuje metódy obsluhujúce prácu so súbormi na užívateľských OneDriveoch. Implementuje nasledovné metódy:

GetAssignmentStatus

Vráti informáciu, či už daný užívateľ konkrétne zadanie odovzdal alebo nie.

GetNumberOfFilesInAssignmentDirectory

Vráti počet súborov v priečinku odovzdania daného užívateľa.

GetDrive

Vráti referenciu na osobný OneDrive daného užívateľa.

CopySubmission

Skopíruje súbory z odovzdávacieho priečinka do učiteľského priečinka.

DeleteFoldersForAssignment

Odstráni priečinok konkrétneho zadania u daného užívateľa.

RenameFoldersForAssignment

Premenuje priečinok konkrétneho zadania u daného užívateľa.

CreateFoldersForAssignment

Vytvorí priečinok, resp. potrebnú adresárovú cestu ku konkrétnemu zadaniu u daného užívateľa.

CreateFoldersForAssignmentAndCopy

Najprv zavolá `CreateFoldersForAssignment` a následne do vytvoreného priečinku nakopíruje prílohu k zadaniu od učiteľa.

GetLink

Vráti URL adresu priečinku so zadaním na OneDrive daného užívateľa.

5.4.2.2 GroupsService

Model **GroupsService** obsahuje metódy obsluhujúce prácu so skupinami a užívateľmi z `Azure Active Directory` a stará sa o synchronizáciu týchto entít s lokálnou databázou. Implementuje nasledovné metódy:

GetGroups

Vráti zoznam objektov typu `Microsoft.Graph.Group`, teda zoznam všetkých skupín z `Azure Active Directory`.

GetStudentsWithAssignment

Vráti zoznam študentov, ktorým bolo priradené dané zadanie. Najprv si z parametru `Assignment` vytiahne všetky skupiny, ktorým bolo zadanie priradené a následne na každú z týchto skupín zavolá `GetMembers`, aby dostala jej zoznam členov z Microsoft Graphu.

GetSubmissionStatus

Vytvorí kolekciu informácií o odovzdaní zadania jednotlivými žiakmi. Využíva pri tom opakované asynchrónne volanie metódy `FilesService.GetAssignmentStatus`, ktorú si ukladá v zozname úloh podobne, ako v ukážke kódu 5.2. Po dokončení každej úlohy jej výsledok vloží do výsledného zoznamu a až po skončení všetkých úloh tento zoznam zoradí podľa priezviska a mena študentov.

GetMembers

Vráti z Microsoft Graphu zoznam študentov, ktorí sú členmi danej skupiny.

5.4.2.3 MailService

Model **MailService** obsahuje metódy obsluhujúce prácu s e-mailami. V prípade našej aplikácie sa jedná len o odoslanie notifikačného e-mailu študentom, ktorým bolo zadanie priradené. Implementuje nasledovné metódy:

SendMail_NewAssignment

Odošle všetkým študentom, ktorí sú členmi skupín, ktorým bolo zadanie priradené, notifikačný e-mail s predpripraveným textom a použije pri tom Office 365 e-mailovú adresu učiteľa.

5.4.3 View Modely

View Modely sú špeciálne druhy modelov, ktoré špecificky slúžia ku „zhromaždeniu“ dát potrebných pre výstup z iných modelov. Sú používané pri renderovaní Viewov, na ktorých potrebujeme niekoľko konkrétnych atribútov z viacerých rôznych modelov.

Pekným príkladom nám môže byť `GroupsDetailsViewModel` z našej aplikácie, ktorý v sebe spája dva modely – `Groups` z lokálnej databázy a `IEnumerable<User>`, čiže kolekciu užívateľov z Microsoft Graphu.

```
namespace CloudHW.ViewModels
{
    public class GroupsDetailsViewModel
    {
        public virtual Groups SchoolGroup { get; set; }
        public IEnumerable<User> Students { get; set; }
    }
}
```

Ukážka kódu 5.7: Ukážka ViewModelu

Takýto model si najprv controller naplní dátami z iných modelov a následne ho hotový odovzdá Viewu na vyrenderovanie.

5.5 Resources

Resources slúžia v našej aplikácii ku šikovnému ukladaniu statických textov, ktoré sa užívateľovi zobrazia. Hlavným dôvodom oddelenia zobrazovaných textov od controllerov a viewov je otvorenie možnosti rôznych jazykových mutácií. Vďaka tomuto prístupu máme všetky hlášky a statické texty pohromade na jednom mieste v súbore `Resource.resx`, ktorého kópie môžeme následne prekladať do ďalších jazykov. Správnym nastavením našej aplikácie môžeme potom detekovať užívateľský jazyk, nastavený v prehliadači a k nemu korešpondujúco zobraziť jazykovú mutáciu aplikácie.

5.6 Controllery

Ako sme si už v stati 4.2.3 povedali, controllery sú spájacím článkom, ktorý získava dáta z jednotlivých modelov a služieb a ďalej ich posúva Viewom, ktoré ich užívateľovi zobrazia. Zo zvyku platí, že pre každú oddelenú sekciu alebo funkcionalitu stránky existuje samostatný controller. Naša aplikácia tak implementuje nasledovné controllery:

5.6.1 AccountController

Controller **AccountController** slúži pre správu užívateľskej relácie. Implementuje nasledovné metódy:

SignIn

Táto metóda je zavolaná po stlačení tlačidla „Prihlásiť sa“ a spúšťa autentifikačný proces a presmeruje užívateľa na prihlasovaciu obrazovku na stránke Microsoft SSO.

SignOut

Táto metóda je zavolaná po stlačení tlačidla „Odhlásiť“ a uzatvára autentifikačný proces – vymaže obsah *claims* a užívateľa presmeruje na obrazovku pre neprihlásených.

ConnectAADTenant

Táto metóda je zavolaná po stlačení tlačidla „Admin Consent“ a presmeruje užívateľa na stránku Microsoft SSO, na ktorej sa nachádza administrátorský súhlas s povoleniami, ktoré si aplikácia pýta. Tento súhlas stačí dať len raz.

AADTenantConnected

Na túto metódu je užívateľ presmerovaný po udelení administrátorského súhlasu. Metóda skontroluje, či bol súhlas úspešný a podľa výsledku tejto kontroly užívateľa presmeruje.

5.6.2 AdminController

Controller **AdminController** slúži len pre zobrazenie tlačidla s administrátorským súhlasom. Implementuje jedinú metódu `Index`:

Index

Táto metóda skontroluje, či má aplikácia administrátorský súhlas a na základe výsledku zobrazí o tom zobrazí príslušnú hlášku a prípadne tlačidlo, odkazujúce na metódu `AccountController/ConnectAADTenant`.

5.6.3 AssignmentsController

Controller **AssignmentsController** je rozhodne najkomplexnejším controllerom v celej aplikácii. Implementuje všetky metódy na vytváranie a správu zadaní. Implementuje nasledovné metódy:

Index

Zavolá View pre úvodnú obrazovku.

Index_LoadTable

Asynchrónne načíta tabuľku so zadaniami z databázy a vráti partial view, ktorý sa dynamicky doplní už do zobrazenej stránky. Táto metóda je volaná pomocou ajaxového volania javascriptovej knižnice jQuery.

Details

Zavolá View pre detail konkrétneho zadania.

Details_LoadTable

Podobne ako `Index_LoadTable` načíta asynchrónne detaily konkrétneho zadania a zoznam žiakov, ktorým bolo toto zadanie priradené spolu so stavom odovzdania. Využíva pritom metódu `GetSubmissionStatus` modelu `GroupsService` a jej výsledky formátuje do View Modelu `AssignmentsDetailsViewModel`. Následne tento výsledok dynamicky doplní už do zobrazenej stránky.

Collect

Táto metóda zozbiera vypracovania od všetkých žiakov, ktorým bolo dané zadanie priradené. Najprv si pomocou `GetStudentsWithAssignment` z Graphu vytiahne všetkých študentov, ktorým bolo zadanie priradené. Potom overí, či sa na učiteľskom OneDrive nachádza priečinok ku danému zadaniu – ak nie, vytvorí ho. Následne vytvorí zoznam úloh, do ktorého postupne popridáva metódu `CopySubmission` z modelu `FilesService` pre každého žiaka, ktorému bolo zadanie priradené. Súbory sa tak kopírujú na učiteľský OneDrive nie synchrónne a tým pádom pomaly, ale paralelne od všetkých žiakov naraz. Pripomeňme si, že naša aplikácia pritom reálne s danými súbormi ani nepríde do styku – kopírovanie sa deje až na úrovni cloudu.

Assign

Táto metóda priradí zadanie skupinám, ktoré v príslušnom Viewe vybral učiteľ[9]. Najprv pomocou `GetNumberOfFilesInAssignmentDirectory` skontroluje počet súborov v priečinku zadania u učiteľa. Ak sa v ňom nejaké súbory nachádzajú, znamená to, že učiteľ pridal ku zadaniu aj prílohu a táto sa bude musieť nakopírovať všetkým žiakom. Následne si pripraví zoznam študnetov zo všetkých skupín, ktorým bolo dané zadanie priradené. Toto sa všetko deje paralelne.

[9] Učiteľ môže vybrať len zo skupín, ktoré majú nastavený parameter `IsClass` na 1

V momente, keď sú všetky úlohy na prípravu zoznamu študentov hotové, môže začať priraďovanie. Aplikácia pre každého študenta v predpripravenom zozname vytvorí asynchrónnu úlohu `CreateFolders`, ktorá je implementovaná v tom istom controlleri. Táto metóda vytvorí pre každého študenta priečinky na jeho osobnom OneDrive korešpondujúce s adresárovou štruktúrou, ktorú sme si predstavili v stati 3.1.7. Tieto všetky úlohy bežia paralelne.

Medzitým sa aplikácia pozrie, či učiteľ zaškrtol možnosť odoslania notifikačného e-mailu. Ak áno, zavolá `SendMail_NewAssignment`, ktorý notifikuje študentov o novom zadaní.

Na konci metódy počkáme, aby sa všetky tieto asynchrónne úlohy dokončili a následne presmerujeme užívateľa späť na `Index` tohto controllera s notifikáciou o úspešnom dokončení operácie.

CreateFolders

Táto metóda je asynchrónne volaná metódou `Assign` a slúži na vytvorenie adresárovej cesty u každého užívateľa, ktorému bolo dané zadanie priradené.

Z parametrov volania metóda vie, či sa budú priečinky len vytvárať, alebo sa do nich bude zároveň kopírovať aj učiteľská príloha k zadaniu. Podľa tohto rozhodnutia do pripraveného zoznamu úloh pridá buď `CreateFoldersForAssignment` alebo `CreateFoldersForAssignmentAndCopy`. Priečinky sa teda u žiakov vytvárajú paralelne. V momente, ako sú všetky priečinky vytvorené sa metóda ukončí – toto správanie je zabezpečené direktívom `Task.WhenAll()`.

Create

Táto metóda vytvorí na základe učiteľom vyplneného formulára nové zadanie, ktoré si uloží v databáze. Zároveň však na učiteľskom OneDrive pomocou volania metódy `CreateFoldersForAssignment` vytvorí adresárovú cestu k danému zadaniu, aby do nej učiteľ mohol prípadne nakopírovať nejaké prílohy skôr, než zadanie priradí žiakom.

Edit

Táto metóda slúži na úpravu parametrov konkrétneho zadania. Kým nie je zadanie priradené žiadnym skupinám, môže učiteľ upravovať všetky jeho parametre. V prípade, že sa rozhodne zmeniť predmet zadania, je pri spracovaní jeho požiadavku odstránený pôvodný priečinok z jeho OneDriveu a následne je vytvorený nový priečinok v zmenenom predmete. Ak sa však rozhodne zmeniť len názov zadania, metóda `RenameFoldersForAssignment` premenuje príslušný priečinok na nový názov.

V momente, ako zadanie učiteľ priradí nejakej skupine sa možnosti zmeniť predmet alebo názov zablokujú, keďže priečinky pre vyzbieranie sú „mapované“ na názov predmetu a názov zadania. Učiteľ však stále môže zmeniť popisný text, prípadne predĺžiť dátum odovzdania.

Delete a DeleteConfirmed

Tieto metódy dohromady implementujú odstránenie zadania z databázy. Jediné, čo sa v skutočnosti pri odstránení zadania stane je to, že sa jeho záznam odstráni z lokálnej databázy. **Aplikácia zámerne neodstraňuje žiadne súbory ani zo žiackych, ani učiteľských OneDriveov.** Je to preto, aby si učiteľ mohol pozrieť vyzbierané odovzdania aj po tom, čo zadanie z databázy odstráni a zároveň preto, aby nebolo možné zvaliť zmiznutie súborov z užívateľských OneDriveov na aplikáciu. Upratovanie v súboroch je teda v osobnej réžii každého užívateľa.

ShowDirectory

Táto metóda otvorí v užívateľskom prehliadači novú kartu s priečinkom daného zadania na užívateľskom OneDrive. Ku získaniu adresy využíva metódu `GetLink` modelu `FilesService`.

5.6.4 GroupsController

Druhým najdôležitejším controllerom v našej aplikácii je **GroupsController**, ktorý úzko komunikuje s modelom `GroupsService`. Implementuje všetky metódy, ktoré spravujú užívateľov a skupiny v aplikácii:

Index

Zobrazí štandardný View so zoznamom synchronizovaných skupín z lokálnej databázy.

Details

Zavolá View pre detail konkrétnej skupiny.

Details_LoadTable

Asynchrónne načíta informácie o konkrétnej skupine z lokálnej databázy a doplní ich aktuálnym zoznamom členov z Microsoft Graphu. Spoločne tak vytvorí novú inštanciu `GroupsDetailsViewModel`, ktorú naplní asynchrónne získanými údajmi a vráti partial view, ktorý sa dynamicky doplní už do zobrazenej stránky. Táto metóda je volaná pomocou ajaxového volania javascriptovej knižnice jQuery.

Edit

Táto metóda umožňuje upraviť údaje o skupine v lokálnej databáze. Jedná sa najmä o parameter `IsClass`, ktorý indikuje, či danej skupine môže byť priradené zadanie.

SyncGroups

Táto metóda je najdôležitejšou v celom controlleri. Má za úlohu zosynchronizovať všetky skupiny a užívateľov z vopred preddefinovaných skupín v `Azure Active Directory` s lokálnou databázou.

Aplikácia si najprv definuje synchronizáčné číslo `SyncNo` a nastaví ho na aktuálnu časovú známku (timestamp). Toto synchronizačné číslo ostane počas celého procesu synchronizácie rovnaké.

Následne si zavolaním metódy `GetGroups` zaistí zoznam všetkých skupín v `Azure Active Directory`. Pre každú skupinu overí, či sa už nachádza v databáze – ak áno, vykoná update atribútu `SyncNo` u daného záznamu na aktuálne synchronizačné číslo – ak nie, vloží nový záznam do databázy.

Ak aplikácia zistí, že sa názov skupiny nachádza na zozname špecialnych skupín pre synchronizáciu užívateľov, začne spracovávať jej užívateľov: Najprv si pomocou metódy `GetMembers` načíta aktuálny zoznam členov tejto skupiny. Následne pridá do zoznamu úloh metódu `ProcessUserInGroup` pre každého užívateľa. Keď sú všetky tieto úlohy hotové, pokračuje synchronizácia ďalšou skupinou.

Po precyklení všetkých skupín pristúpi aplikácia k poslednému kroku – vymazaniu záznamov z lokálnej databázy, ktoré sa už v Microsoft Graphe nenachádzajú. Tieto záznamy dokáže aplikácia zistiť veľmi jednoducho: v databáze majú v atribúte `SyncNo` ešte staré synchronizačné číslo. Preto označí k vymazaniu všetky skupiny a všetkých užívateľov, ktorých synchronizačné číslo je menšie ako to, ktoré sa nastavilo na začiatku synchronizačného procesu.

Keďže databáza pracuje transakčným spôsobom, až po úspešnom ukončení všetkých týchto operácií prebehne tzv. „commit“, tj. prenesenie výsledkov do databázy.

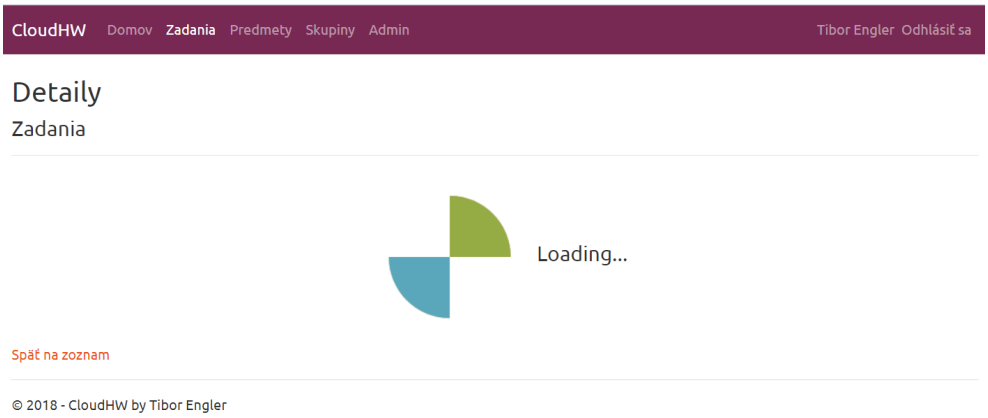
ProcessUserInGroup

Táto metóda je pomocou metódou k `SyncGroups`. U daného užívateľa zistí, či sa nachádza v lokálnej databáze. Ak áno, dvihne mu synchronizačné číslo na aktuálnu hodnotu a skontroluje, či má záznam užívateľa v lokálnej databáze priradenú rolu rovnomennú s názvom kontrolovanej skupiny. Ak nie, pridá mu ju.

Ak užívateľ v lokálnej databáze neexistuje, vytvorí mu nový záznam a tomuto záznamu priradí rolu rovnomennú s názvom kontrolovanej skupiny.

5.6.5 HomeController

Tento controller implementuje jedinú triedu `Index`, ktorá zobrazuje rôzne privítanie pri prihlásených a neprihlásených užívateľov.



Obr. 5.2: Loading Screen pri dynamickom načítaní dát

5.6.6 SubjectsController

Tento controller je zodpovedný za správu predmetov, v ktorých sa zadania vytvárajú. Implementuje štandardné `CRUD` operácie, ktorými sa predmety pridávajú, upravujú a mažú z lokálnej databázy.

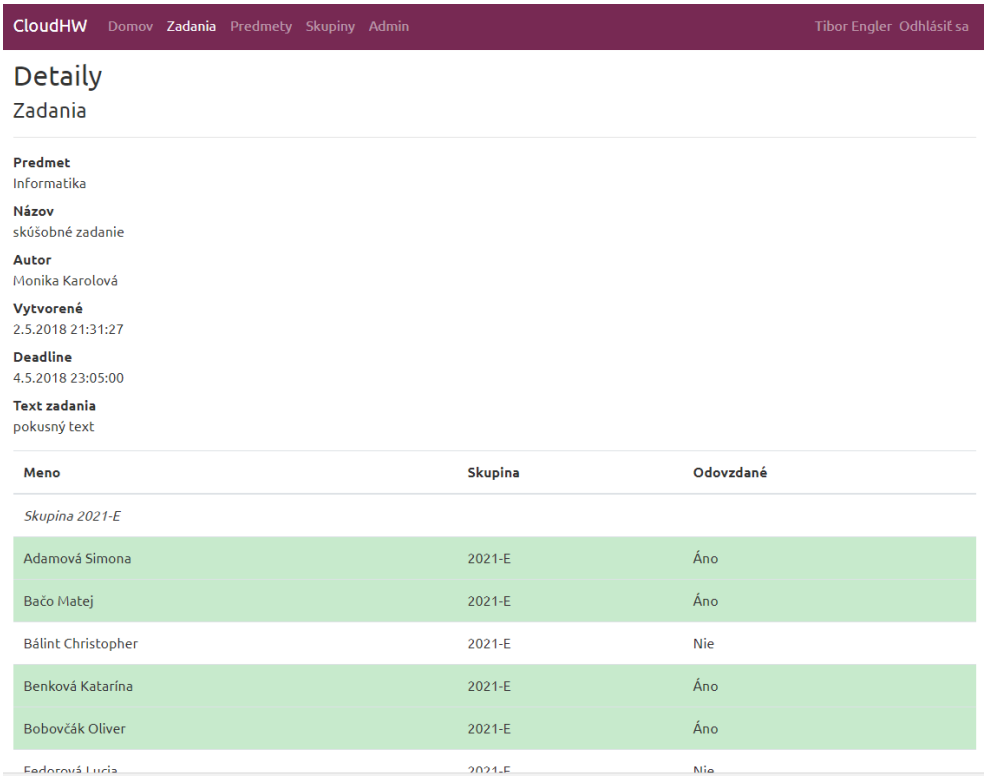
5.7 Viewy a prezentačná vrstva

Viewy sú tou peknou fasádou, ktorú naša aplikácia zobrazuje užívateľom. Všetky controllery majú takmer ku každej akcii priradený vlastný view, ktorý sa volá rovnako ako daná akcia a je uložený v priečinku s názvom daného controlleru. Všetky controllerové viewy sú vkladané do šablóny `_Layout.cshtml`, ktorá implementuje základnú kostru stránky s navigáciou, CSS štýlmi a podobne.

K peknému vizuálu stránky dopomáha bootstrapová téma, ktorá sa stará okrem iného aj o responzívnosť webu na mobilných zariadeniach.

Aby načítanie stránky pôsobilo pre užívateľa svižnejšie, najčastejšie požiadavky, ako napríklad zobrazenie zoznamu zadaní, prípadne zobrazenie stavu vypracovania u jednotlivých žiakov, sú riešené dynamickým doplňaním dát pomocou javascriptového jQuery (technológia `Ajax`). Stránka sa teda najprv načíta bez dát (takmer okamžite) a dáta sa po malej chvíľke za sympatického krútenia „Loading“ obrázku do stránky doplnia. Tento proces nám znázorňujú obrázky 5.2 a 5.3.

V našej aplikácii v sebe viewy neobsahujú (takmer) žiadny výstupný text. Pri textovom výstupe sa totiž odkazujú na jazykový súbor `Resources`, ktorý sme si prednedávnom predstavili. Vďaka nemu nie je nutné vyrábať samostatné viewy pre každý nový jazyk, ale stačí jeden view pre danú funkcionalitu, do ktorého sa už text podľa užívateľského jazyka doplní.



Obr. 5.3: Obrazovka s dynamicky doplnenými údajmi

5.7.1 Sekcia Zadania

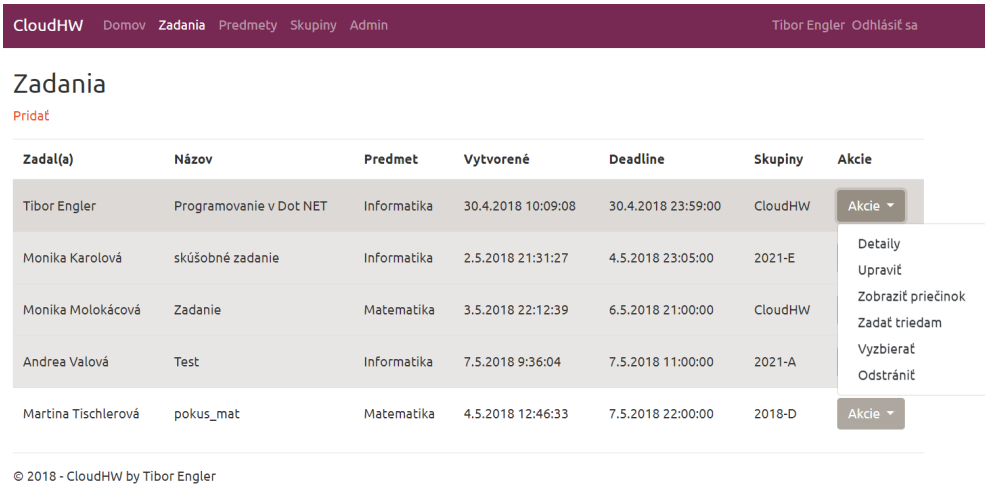
Sekcia stránky **Zadania** odkazuje na controller `AssignmentsController`. Po kliknutí na navigačné tlačidlo sa zobrazí zoznam zadaní zoradený podľa dátumu odovzdania. Užívatelia majú možnosť zobraziť detaily každého zadania, no ostatné akcie sú prístupné len autorovi konkrétneho zadania. Jednotlivé odkazy odkazujú na akcie controlleru `AssignmentsController` (viď obrázok 5.4).

5.7.2 Sekcia Predmety

Sekcia stránky **Predmety** odkazuje na controller `SubjectsController`. Po kliknutí na navigačné tlačidlo sa zobrazí zoznam predmetov zoradený podľa názvu. Ku každému predmetu sú priradené navigačné tlačidlá „Upraviť“, „Detaily“, „Odstrániť“, ktoré spolu s centrálnym tlačidlom „Pridať“ reprezentujú štandardné `CRUD` operácie.

5.7.3 Sekcia Skupiny

Sekcia stránky **Skupiny** okdazuje na controller `GroupsController`. Po kliknutí na navigačné tlačidlo sa zobrazí zoznam synchronizovaných skupín z lo-



Obr. 5.4: Sekcia Zadania

kálnej databázy zoradený podľa toho, či je skupina používaná v našej aplikácii (má nastavený atribút `IsClass`) a následne sú tieto skupiny zoradené podľa názvu. Ku každej skupine sú priradené tlačidlá „Upraviť“ a „Detaily“.

Pre užívateľov v roli administrátora je prístupné tlačidlo „Synchronizovať“, ktoré spúšťa metódu `SyncGroups` príslušného controlleru. Po úspešnom priebehu tejto metódy je užívateľ notifikovaný zelenou notifikáciou v pravom hornom rohu okna.

Kapitola 6

Nasadenie a testovanie

6.1 Nasadenie aplikácie v produkčnom prostredí

Pre účely testovania učiteľmi na našej testovacej škole musíme aplikáciu **CloudHW** sprístupniť verejnosti. Na to poslúži dedikovaný virtuálny server `Windows Server 2016` v serverovom racku na našej testovacej škole.

Na tomto serveri je nainštalovaný server `Microsoft Internet Information Services`, ktorý funguje ako webový a zároveň aj aplikačný server. Pre účely ukladania lokálnej databázy beží na tom istom virtuálnom serveri aj `Microsoft SQL Express` – lightweightová verzia robustného databázového serveru od Microsoftu. K zabezpečeniu HTTPS pripojenia používa server zadarmo vydaný certifikát od certifikačnej autority LetsEncrypt.

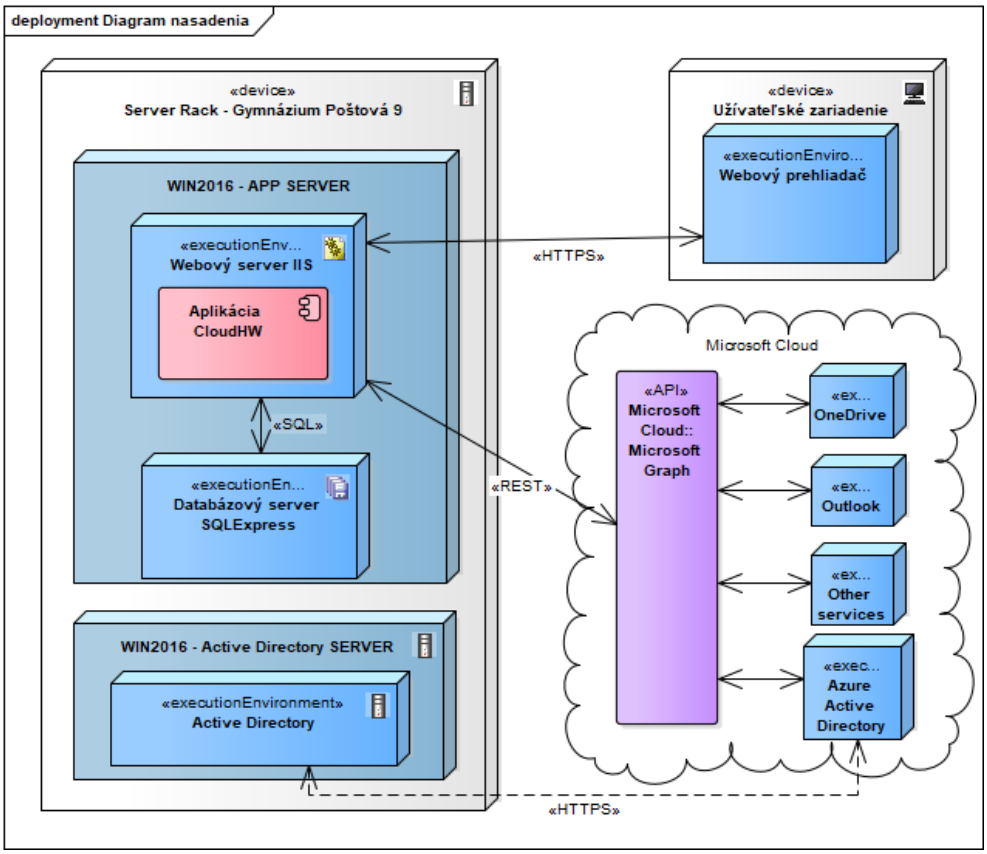
Čitateľ sa iste spýta, prečo sme aplikáciu, ktorá pracuje výhradne s cloudom nedeployovali do cloudu. Odpoveďou, okrem finančnej stránky veci, je možnosť vyladenia všetkých parametrov behu aplikácie na vlastnom serveri. V budúcnosti však autor zvažuje presun aj samotnej aplikácie do cloudu od Microsoftu.

Diagram nasadenia aplikácie na virtuálnom serveri v testovacej škole nájdeme na obrázku 6.1. Na diagrame je prerušovanou čiarou znázornené synchronizačné spojenie medzi lokálnym `Active Directory` a cloudovým `Azure Active Directory`, ktoré navzájom synchronizuje tieto dva adresáre niekoľkokrát do hodiny.

Online verziu aplikácie nájdeme na **https://app.gympos.sk**.

6.2 Testovanie

Testovanie aplikácie prebiehalo počas vývoja jednotlivých funkcionalít ich systematickým skúšaním podľa vzorov jednotlivých prípadov použitia. Najdôležitejšie akcie controllerov `Assignemnts` a `Groups` boli testované Unit testami vo vývojárskom prostredí Visual Studio. Toto všetko na lokálnom, vývojárskom počítači.



Obr. 6.1: Diagram nasadenia aplikácie CloudHW

Počas testovania prezentačnej vrstvy bola aplikácia v režime debuggingu, takže bolo možné pozorovať všetky druhy správania a výnimiek, ktoré sa vyskytli. Určité typy výnimiek, ako napríklad neexistujúca užívateľská OneDrive stránka, boli zámerne potlačené, aby nedošlo k prerušeniu komplexnejšieho užívateľského procesu, no ich výsledok je užívateľovi prezentovaný po dokončení tohto procesu.

Po deploymente do produkčného prostredia bol na produkčný server nainštalovaný balík `Visual Studio Remote Debugger`, vďaka ktorému sme mohli aplikáciu debuggovať aj nadialku. Táto funkcionalita je vhodná najmä preto, že hoci malé, ešte stále existujú drobné rozdiely medzi lokálnym `IIS Express` a vzdialeným `IIS` serverom. Najcennejším výstupom z testovania sú však reakcie samotných laických užívateľov, ktoré nasledujú v ďalšej stati.

6.2.1 Testovanie v produkčnom prostredí

Keďže žiadna aplikácia by nemala navždy ostať v autorskom počítači, ponúkli sme túto aplikáciu na testovanie do našej testovacej školy – Gymnázia Poštová 9

v Košiciach. Požiadali sme učiteľov predmetu informatika, aby počas jedného týždňa intenzívne skúšali našu aplikáciu spolu so žiakmi a povzbudili sme ich aj do skúšania extrémnych prípadov. Na záver týždňa sme ich poprosili, aby odpovedali na nižšie uvedené otázky. Do odovzdania bakalárskej práce sa k nám dostali odpovede od troch zo štyroch oslovených učiteľov. Pre zachovanie ich súkromia ich označíme U1 až U3.

Je podľa Vás táto myšlienka využiteľná v praxi? Pomáha Vám tento software uľahčiť vyučovanie?

U1: „Čo sa týka využitia, tak je to použiteľné a celkom dobre sa to hodí na domáce úlohy, keďže k tomu vedia žiaci pristupovať z domu, nakopírovať tam svoje práce, rovnako aj pre učiteľov je fajn prístup z domu.

Pokiaľ to chceme využiť priamo v škole napr. na odovzdávanie písomky, tak to bude asi dosť otravné, pretože sa žiaci musia prihlásiť na OneDrive či už na začiatku, aby videli priložené súbory, prípadne iba na konci, aby nakopírovali to, čo chcú odovzdať."

U2: „Myslím si, že po vyladení drobných chybičiek krásy bude Vaša aplikácia vo vyučovaní použiteľná. Prístup k domácim úlohám žiakov z domu je ako pre mňa, tak aj pre žiakov pohodlný, pretože ich nemusia do školy nosiť na USB kľúčoch."

U3: „Navrhnuté cloudové riešenie je v porovnaní s bežným cloudovým úložiskom veľkým prínosom pre organizáciu vyučovacieho procesu, a to predovšetkým pri hodnotení žiakov a pri zdieľaní dát učiteľa so žiakmi jednotlivých skupín (tried). Veľkou výhodou je plná kompatibilita vytvorenej aplikácie s cloudovým úložiskom Office 365 používaným vo vyučovaní, čo zaručuje dostupnosť odovzdaných žiackych prác v personalizovanom cloudovom prostredí učiteľa, dostupnom po prihlásení tak v prostredí školskej siete ako aj mimo nej. Rovnako žiakom sú zadania alebo zdieľané výučbové materiály dostupné v ich personalizovanom cloudovom prostredí. Systém odovzdávania prác stiahnutím priamo zo žiackeho konta autentifikuje prácu žiaka a v porovnaní so spoločným úložiskom pre hodnotenú skupinu je nástrojom dôvernosti odovzdaných riešení medzi učiteľom a jednotlivými žiakmi. Autentifikácia a dôvernosť procesu zdieľania zadaní a riešení medzi žiakom a učiteľom je na úrovni e-mailovej komunikácie, s nesporným odbúraním administratívnej záťaže na oboch stranách."

Reakcia autora: Odpovede učiteľov na túto otázku sú viac-menej pozitívne, čo je dobrým znakom budúceho širšieho nasadenia aplikácie. Autor nevidí ako veľkú prekážku to, že by sa žiaci museli prihasovať na OneDrive počas písomiek, pretože túto funkciu je možné nakonfigurovať vo vstavanom klientovi OneDrive for Business. V každom prípade je však táto aplikácia primárne stavaná pre odovzdávanie domácich úloh s dlhšou časovou platnosťou. Nebudeme sa však vyhýbať vývoju modifikácie, vhodnej aj pre krátkodobé zadania.

Je tento software jednoduchý a intuitívny? (ak nie, prečo?)

U1: „Skúsila som vyrobiť zadanie, prideliť mu triedu, vyzbierať zadania od testovacej triedy. Žiaľ zo strany žiaka som to nevidela. Čo sa týka ovladánia, tak tam problem nevidím, ide to vcelku jednoducho. Samozrejme, ako na všetko nové si bude potrebné zvyknúť."

U2: „Aplikácia je jednoduchá a rýchla, nie je tam nič rušivého ani naviac, takže sa pri práci s ňou človek nemôže pomýliť. Všetko je tam, kde by som to očakával. Oceňujem aj responzívny dizajn, vďaka ktorému mi aplikácia bežala aj na telefóne."

U3: „Áno, funkcionalita navrhnutých nástrojov je aj pre nového používateľa intuitívne zrozumiteľná."

Reakcia autora: Dali sme si špeciálne záležať na tom, aby bola aplikácia navrhnutá v minimalistickom štýle. Keďže sa jedná o nástroj, ktorý v prvom rade spravuje obsah, nemal by svojim vizuálom daný obsah prekričať. Podľa reakcií učiteľov sa to podarilo.

Aké zlepšenia alebo rozšírenia by ste si vedeli predstaviť? Čo by Vám pomohlo v procese zbierania vypracovaní?

U1: „Pre ďalšie používanie by bolo vhodné deliť triedu na skupiny, tým by sa čiastočne eliminoval počet vygenerovaných zadaní a hlavne žiaci by mali zadania len od svojho učiteľa. A taktiež si viem predstavit, že v časti Zadania by každý učiteľ videl len svoje zadania, resp. by si mohol vyselektovat svoje zadania, pripadne selektovat svoje zadania pre príslušnú triedu/skupinu. Ak si to zoberieme, že počas týždňa má INF/PRG/CIF[10] 44 skupín, tak sa kľudne môže stať, že za týždeň vygenerujeme celkovo 44 zadani a to už nemusí byť práve také prehľadné... "

U2: „Rozhodne by sa hodila možnosť filtrovať zadania podľa skupín, ktorým bolo zadanie zadané, zároveň by bolo vhodné mať možnosť už existujúce zadanie duplikovať len s pozmenenou deadlineou. Bolo by fajn prepojiť aplikáciu s EduPage kvôli hodnoteniu žiackych prác na jednom mieste. Pri rozšírení používania Vašej aplikácie aj na iné predmety by bolo vhodné dorobiť kontrolu proti opisovaniu na písomkách, resp. plagiátorstvu v domácich úlohách."

U3: „V ideálnom prípade by sa hodila integrácia s aSc Edupage kvôli udeľovaniu bodov za jednotlivé odovzdania. Určite by bolo vhodné mať možnosť skupiny rozdeliť do menších podľa reálnych vyučovacích skupín, aby sa zadania zbytočne nevytvárali aj žiakom, ktorých učiteľ, vyučujúci inú skupinu, neučí. A maličkosť na záver: nastaviť implicitne čas odovzdania na 24.00.00 h v prípade, že údaj zostane učiteľom nevyplnený."

Reakcia autora: Aplikácia v čase testovania skutočne neposkytovala možnosť filtrovať zobrazené zadania – táto funkcionalita pribudla až po ukončení testovania hlavných súčastí, pretože sme ju nepovažovali za nutnú k u-

[10]Informatika, Programovanie, Cvičenia z informatiky

skutočneniu testovania. Čo sa duplikácie zadania týka, bude veľmi jednoduché takúto funkcionalitu doplniť. Integrácia s EduPage bude bohužiaľ ťažšia, keďže firma aSc nemá otvorené API pre komunikáciou s ich softwareom, ale čo nie je, môže byť. Čo sa kontroly proti plagiátorstvu týka, jednalo by sa o novú, enormne veľkú funkcionalitu, ktorá už nie je predmetom tejto práce. V každom prípade je to zaujímavý námet na budúce rozšírenie už zabehnutého softwareu.

Vyskytli sa nejaké problémy počas Vášho používania? Ak áno, aké?

U1: „Čo sa týka problémov, tak sa mi dvakrát stalo, že som sa prihlásila zo vstupnej stránky app.gympos.sk, ale nemala som k dispozícii menu ... pôvodne som si myslela, že to bolo len v utorok a že som to začala skúšať príliš skoro, ale potom sa mi to isté stalo aj v piatok, v priebehu dňa."

U2: „Jediným problémom je rýchlosť nahrávania súborov. Žiakom trvalo nahrávanie do cloudu dlhšie, než to trvá na sieťové disky. Vyzbieranie súborov však aj napriek tejto nízkej rýchlosti prebehlo pomerne svižne."

U3: „Nie, k žiadnym problémom nedošlo."

Reakcia autora: „Čo sa problémov učiteľa U1 týka, mohlo dôjsť ku vymazaniu bezpečnostného cookie niektorým z čistacich softwareov – zvážime ukladanie týchto dát na strane serveru. Čo sa rýchlosti uploadu súborov týka, bude to pravdepodobne spôsobené používaním protokolu `WebDAV`, ktorý je pomalý a zastaralý, nakoľko škola má privedené optické pripojenie a interné sieťové prvky sú takmer nové (rýchlosť uploadu je okolo 50 Mbit/s). Zavedením OneDrive for Business by sa rýchlosť nahrávania mala rapídne zvýšiť."

Záver

Cieľom tejto práce bolo nájsť vhodné riešenie problematiky odovzdávania školských domácich prác cez internet. Ukázali sme si aktuálnu situáciu vo väčšine škôl a prišli sme na to, prečo je vhodným riešením v tomto sektore práve využitie cloudových technológií.

Vďaka rešeršu existujúcich riešení sme mali možnosť inšpirovať sa pri vytváraní našej aplikácie už fungujúcimi projektmi a zároveň nám to spolu s užívateľskými požiadavkami dalo dobrý prehľad o tom, ktorým smerom sa vo vývoji uberať.

Následne sme si ukázali, na akých princípoch postaviť modernú webovú aplikáciu, ktorá bude efektívna a ľahko udržiavateľná. Zároveň sme sa zoznámili s možnosťami autentifikácie a autorizácie voči poskytovateľovi cloudu, so spôsobom komunikácie s cloudovými službami a prišli sme na záver, že ak je aplikácia vhodne implementovaná, využije minimálne alebo žiadne lokálne prostriedky.

Testovanie na vzorke žiakov a učiteľov, ktorí majú blízko k IT nám zas prezradilo, ako sa aplikácia správa pri reálnej záťaži a prečo je v takomto prípade výhodnejšie použiť asynchrónne programovanie a Ajax.

Ako nám prieskum spokojnosti testovacej vzorky ukázal, ešte stále existuje priestor pre rozšírenia. Určite by bolo pohodlné, keby učitelia mohli zadávať hodnotenie odovzdaných prác priamo v našom systéme a pomocou API by sa výsledky prenášali do aSc Edupage – to by si však vyžadovalo vytvorenie API zo strany aSc, ktorá svojím spôsobom tomuto produktu konkuruje. Učitelia taktiež navrhli vylepšenia v rámci organizácie skupín, ktorým sú zadania rozdeľované a taktiež vyjadrili požiadavku na automatizovanú kontrolu proti plagiátorstvu.

Aplikácia CloudHW je v súčasnosti intenzívne používaná v rámci pilotného testovania na Gymnáziu, Poštová 9 v Košiciach a škola pravidelne komunikuje s autorom a dodáva nové nápady na vylepšenia aplikácie tak, aby mohla byť v krátkom čase uvedená do ostrej prevádzky.

Verím, že rozšírením tejto aplikácie o ďalšie funkcionality sa otvoria dvere

aj do ďalších škôl, ktoré po príklade našej testovacej školy prejdú plynulo do cloudu. Vďaka elektronickému odovzdávaniu domácich úloh v nich učitelia budú mať konečne poriadok a všetci ušetria niekoľko ton papiera – a tým aj naše životné prostredie.

Literatúra

[1] DESHPANDE, A.: How much paper do schools use? [online]. 2014, [cit. 2018-03-20]. Dostupné z: http://web.archive.org/web/20141027214519/https://blog.frevvo.com/2014/10/27/how-much-paper-do-schools-use/

[2] asc – academic solutions: ascEdupage Funkcie [online]. 2017, [cit. 2018-03-20]. Dostupné z: http://web.archive.org/web/20171025185948/http://www.ascagenda.com/

[3] aSc Applied Software Consultants: ascEdupage Funkcie [online]. 2018, [cit. 2018-03-20]. Dostupné z: http://web.archive.org/web/20180106173553/https://www.edupage.org/?lng=sk&ctn=sk

[4] Microsoft Corporation: OneNote Class Notebook [online]. 2018, [cit. 2018-03-20]. Dostupné z: http://web.archive.org/web/20180416092938/https://www.onenote.com/classnotebook

[5] FCHPT STU v Bratislave: *Moodle 1.8: príručka učiteľa [online].* [cit. 2018-03-28]. Dostupné z: http://www.kirp.chtf.stuba.sk/moodle/pluginfile.php/13521/mod_page/content/2/m18uc.pdf

[6] AxisTwelve Limited: About the creators of Teacher Dashboard [online]. 2015, [cit. 2018-03-20]. Dostupné z: http://web.archive.org/web/20150519041238/http://www.teacherdashboard365.com/about-us

[7] AxisTwelve Limited: Azure Active Directory group sync [online]. 2017, [cit. 2018-03-20]. Dostupné z: http://web.archive.org/web/20170923202329/http://www.teacherdashboard365.com:80/aadsync

[8] AxisTwelve Limited: TD Basic Features [online]. 2017, [cit. 2018-03-20]. Dostupné z: http://web.archive.org/web/20170830222650/http://www.teacherdashboard365.com:80/features

[9] TRČKA, J.: Aplikace pro hodnocení a recenzování produktú internetového obchodu. 2014.

[10] Microsoft Corporation: ASP.NET Overview [online]. 2018, [cit. 2018-03-28]. Dostupné z: https://msdn.microsoft.com/en-us/library/4w3ex9c2.aspx

[11] ČÁPKA, D.: 1. díl - Úvod do ASP.NET [online]. 2014, [cit. 2018-03-28]. Dostupné z: https://www.itnetwork.cz/csharp/asp-net/tutorial-uvod-do-asp-dot-net

[12] AAD Graph team: Microsoft Graph or Azure AD Graph [online]. 2016, [cit. 2018-04-02]. Dostupné z: https://blogs.msdn.microsoft.com/aadgraphteam/2016/07/08/microsoft-graph-or-azure-ad-graph/

[13] BRANNIAN, J.; ALTIMORE, P.; JACKETT, B. T.; aj.: Get access tokens to call Microsoft Graph [online]. 2017, [cit. 2018-04-02]. Dostupné z: https://developer.microsoft.com/en-us/graph/docs/concepts/auth_overview

[14] BRANNIAN, J.; GRAHAM, L.; KOVAL, E.; aj.: Microsoft Graph permissions reference [online]. 2018, [cit. 2018-04-02]. Dostupné z: https://developer.microsoft.com/en-us/graph/docs/concepts/permissions_reference

[15] BRANNIAN, J.; GRAHAM, L.: Register your app with the Azure AD v2.0 endpoint [online]. 2018, [cit. 2018-04-02]. Dostupné z: https://developer.microsoft.com/en-us/graph/docs/concepts/auth_register_app_v2

[16] SINGHAL, H.; MOHANRAM, P.; TILLMAN, M.; aj.: Azure Active Directory v2.0 and the OpenID Connect protocol [online]. 2018, [cit. 2018-04-24]. Dostupné z: https://docs.microsoft.com/en-us/azure/active-directory/develop/active-directory-v2-protocols-oidc

[17] SINGHAL, H.; TILLMAN, M.; ATWAL, G.; aj.: v2.0 Protocols - OAuth 2.0 Authorization Code Flow [online]. 2018, [cit. 2018-04-24]. Dostupné z: https://docs.microsoft.com/en-us/azure/active-directory/develop/active-directory-v2-protocols-oauth-code

[18] SINGHAL, H.; MOHANRAM, P.; TILLMAN, M.; aj.: v2.0 Protocols - SPAs using the implicit flow [online]. 2018, [cit. 2018-04-24]. Dostupné z: https://docs.microsoft.com/en-us/azure/active-directory/develop/active-directory-v2-protocols-implicit

[19] ALTIMORE, P.; aj.: Azure Active Directory v2.0 and the OAuth 2.0 client credentials flow [online]. 2017, [cit. 2018-04-02]. Dostupné

z: https://docs.microsoft.com/en-us/azure/active-directory/develop/active-directory-v2-protocols-oauth-client-creds

[20] SINGHAL, H.; TILLMAN, M.; CANUMALLA, N.; aj.: Azure Active Directory v2.0 and OAuth 2.0 On-Behalf-Of flow [online]. 2018, [cit. 2018-04-24]. Dostupné z: https://docs.microsoft.com/en-us/azure/active-directory/develop/active-directory-v2-protocols-oauth-on-behalf-of

[21] Microsoft; PASIC, A.; DYKSTRA, T.: Understanding the ASP.NET MVC Execution Process [online]. 2009, [cit. 2018-04-02]. Dostupné z: https://docs.microsoft.com/en-us/aspnet/mvc/overview/older-versions-1/overview/understanding-the-asp-net-mvc-execution-process

[22] CARTER, P.; WENZEL, M.; LUZGAREV, A.; aj.: Asynchronous programming [online]. 2016, [cit. 2018-04-03]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/csharp/async

[23] HOLAN, T.: princip async/await z jazyka c# 5.0 [online]. 2012, [cit. 2018-04-03]. Dostupné z: https://www.dotnetportal.cz/blogy/15/Null-Reference-Exception/5304/Princip-async-await-z-jazyka-C-5-0

Dodatok A

Zoznam použitých skratiek

ACL Access Control List

AD Active Directory

Ajax Asynchronous JavaScript + XML

API Application programming interface

ASP Active Server Pages

Azure AD Azure Active Directory

CRUD Create, Read, Update, Delete

GUID Globally Unique Identifier

HTML Hypertext Markup Language

HTTP Hypertext Transfer Protocol

HTTPS Secure Hypertext Transfer Protocol

IIS Internet Information Services

OAuth Open standard for access delegation

REST Representational state transfer

SMB Server Message Block

SQL Structured Query Language

SSH Secure Shell

UC Use Case

WinSCP Windows Secure Copy

WYSIWYG What You See Is What You Get

Dodatok B

Obsah priloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `impl` ..... zdrojové kódy implementácie
    - `CloudHW`
    - `CloudHW.sln`
    - `Nuget.config`
  - `thesis` ..... zdrojová forma práce vo formáte LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce vo formáte PDF
  - `thesis.ps` ..... text práce vo formáte PS