Technická univerzita v Liberci

FAKULTA PŘÍRODOVĚDNĚ-HUMANITNÍ A PEDAGOGICKÁ

Katedra: Ústav nových technologií a aplikované inf. (FM)

Studijní program: Informatika

Studiní obor: Informatika se zaměřením na vzdělávání /
Anglický jazyk se zaměřením na vzdělávání

ELEKTRONICKÁ TŘÍDNÍ KNIHA
AN ELECTRONIC CLASS REGISTER

Bakalářská práce: 11-FP-NTI-01

Autor: **Podpis:**

Miloš SEDLÁČEK

Adresa:

SNP 853, Hradec Králové, 500 03

Vedoucí práce: Mgr. Jiří Vraný, Ph.D.

Konzultant:

Počet

stran	grafů	obrázků	tabulek	pramenů	příloh
64	0	5	0	31	1

V Liberci dne: 27.4.2011

Čestné prohlášení

Název práce: Elektronická třídní kniha

Jméno a příjmení autora: Miloš Sedláček

Osobní číslo: P09001116

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, zejména § 60 – školní dílo.

Prohlašuji, že má bakalářská práce je ve smyslu autorského zákona výhradně mým autorským dílem.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím bakalářské práce.

Prohlašuji, že jsem do informačního systému STAG vložil elektronickou verzi mé bakalářské práce, která je identická s tištěnou verzí předkládanou k obhajobě, a uvedl jsem všechny systémem požadované informace pravdivě.

V Liberci dne: 27.4.2011 _____

Miloš Sedláček

Poděkování

Rád bych poděkoval Mgr. Jiřímu Vranému, Ph.D., svému vedoucímu práce, za to, že na mne i po několika letech nezapomněl, a ve chvíli, kdy jsem potřeboval jeho konzultaci, si na mne vždy udělal čas. Práce by bez něho zcela jistě nebyla tím, čím je, resp. by pravděpodobně nebyla vůbec.

Rád bych také poděkoval Mgr. Davidu Kmochovi, jelikož to je právě on, komu vděčí zkušební verze vytvořené aplikace za její umístění na serveru TUL.

Anotace

Tato práce se zabývá především vývojem webové aplikace zastupující funkci klasické třídní knihy. Aplikace byla vytvořena za pomoci Django web frameworku, aplikace pracuje s SQLite3 databází, a je určena jak pro PC, tak pro mobilní zařízení. Text samotné práce pak popisuje jednak teoretické pozadí projektu, konkrétně jaké má využití elektronických nástrojů v této oblasti výhody, a jaké nástroje jsou již na českém trhu k nalezení, hlavně ale popisuje vytvořenou aplikaci samotnou. Čtenář zde tedy najde, jak a proč byla vytvořena, či jaké má taková aplikace další možné využití.

Klíčová slova: elektronická správa výuky, školní IS, Django, Python, web

Annotation

This work deals mainly with the development of a web application supposed to substitute the function of a normal class register. The application was created using the Django web framework; it works with the SQLite3 database, and is aimed at both the PC and mobile platforms. The text of the work describes not only the theoretical background of the project, meaning the advantages of using electronic tools in such a field, and which other similar projects are already available, but also, and mainly, the created application itself. This means the reader will find why and how the application was created, or what other usages this particular application may have.

Key words: electronic class work administration, school IS, Django, Python, web

Obsah

Úvod

Situace, kdy školy sahají k elektronickým nástrojům pro správu výuky, či dokonce správu celé školní administrativy, nejsou dnes již žádnou výjimkou. Není tedy ani překvapující, že se na českém trhu objevila v několika posledních letech řada nástrojů řešících tuto problematiku. Poněkud zklamáním pak ovšem je, že zde prakticky neexistuje žádný, minimálně ne zcela, volně šiřitelný a modifikovatelný nástroj, který by školám poskytl alternativu stávajících komerčních řešení.

Toto, spolu se značnou zajímavostí celé myšlenky elektronických nástrojů pro správu výuky, bylo důvodem k vytvoření této práce.

Práce samotná má v podstatě dva hlavní cíle. Prvním z nich je, jak definuje zadání práce, vytvoření webové aplikace zastupující funkci klasické třídní knihy. Tato část práce, resp. kompletní zdrojový kód aplikace, se nalézá na nosiči přiloženém k této práci. Samotnou aplikaci si lze potom, pravděpodobně však pouze dočasně, také vyzkoušet na URL http://147.230.76.12.

Druhým cílem této bakalářské práce bylo poté vytvoření tohoto textu. Text samotný je rozdělen do dvou částí, teoretické a praktické. Teoretická část zběžně popisuje elektronické nástroje pro správu výuky a nastiňuje současný stav obdobných produktů na českém trhu. Praktická potom pochopitelně rozebírá jednotlivé aspekty vytvořené aplikace. V praktické části jsou umístěny kapitoly, které popisují aplikaci jako celek, přibližují využité nástroje a technologie, které ukazují možné praktické využití aplikace, pak ale také kapitoly zabývající se konkrétním popisem mechanizmů ovládajících aplikaci, tedy kapitoly o model, view, a template, kapitolu o nasazení aplikace na mobilních zařízeních, o zabezpečení aplikace, a na závěr samozřejmě kapitolu popisující konkrétní postup, jak vytvořenou aplikaci zprovoznit.

Závěrem je nutné pouze podotknout, že je tento text míněn jako doplněk vytvořené aplikace, ne naopak. Pro jeho dostatečné pochopení je tedy nutná alespoň základní znalost programovacího jazyka Python, a je také důrazně doporučeno současně s textem zkoumat i zdrojový kód aplikace. Kód aplikace je velmi snadno pochopitelný,

v textu proto nebudou popsány konkrétní řádky kódu, nýbrž buď jen principy jeho fungování, nebo dokonce pouze jeho zajímavosti.

Zároveň je nutné jasně říci, že nebylo cílem této práce vytvoření komplexního systému pro správu výuky, v jakékoli jeho možné podobě. Zadání jasně definuje cíl práce jako „*vytvoření webové aplikace, jenž bude zastupovat funkci klasické třídní knihy (tzn. evidence známek studentů, docházky, popis hodiny, ...).*“

Teoretická část

Předtím, než je možné přistoupit k praktické části této práce, je třeba alespoň částečně popsat teoretickou část dané problematiky, tedy teorii kolem elektronické správy výuky. Jde hlavně o jisté zdůvodnění, proč vůbec přistoupit k vytvoření nástroje pro elektronickou správu výuky, jaké jsou jinými slovy výhody v porovnání s lety ověřenou a ozkoušenou klasickou formou, jaká je vlastně autorova představa o ideálním systému pro správu výuky obecně, tedy k jakému ideálu se přímo či nepřímo snaží přiblížit, a na závěr jaké takové systémy, ať více či méně úspěšné a oblíbené, lze dnes na českém trhu již najít. Zmíněná témata tvoří pomyslné základy části praktické, následující kapitola se bude tedy každému z nich alespoň krátce věnovat.

1 Elektronická správa výuky

Pojmem „*elektronická správa výuky*“ jsou míněny především softwarové nástroje pro správu třídní knihy, zápis známek, správu rozvrhů hodin, či třeba nástroje pro správu a plánování výuky. Ač jsou tyto nástroje často shlukovány do systémů mající ještě množství dalších funkcí, často spojené s administrativou samotné školy, následující text se bude věnovat pouze samotné „*elektronické správě výuky*“.

1.1 Výhody / nevýhody oproti klasické formě

Výhody a nevýhody spojené s elektronickou formou správy výuky závisí pochopitelně na konkrétním návrhu nástrojů, obecně vzato je lze však rozdělit v podstatě do tří kategorií – výhody / nevýhody pro uživatele, tedy učitele, pro školu, která k využití takových nástrojů přistoupí, a pro žáky, resp. jejich rodiče. V této kapitole budou zmíněny alespoň některé z nich.

Výhody a nevýhody elektronické správy výuky jsou bezesporu nejdůležitější z pohledu učitele, tedy člověka, který bude s takovými nástroji denně pracovat. Nemělo-li by mu takové nástroje jeho každodenní práci nijak ulehčit, zcela jistě by nemělo

význam o jejich využití ani uvažovat. Právě v této oblasti však mají elektronické nástroje hodně co nabídnout. Mezi výhody patří například jejich téměř absolutní nevázanost na klasická fyzická média. Server, na kterém jsou elektronické nástroje umístěny, je samozřejmě také druh fyzického média, jedná se však o jedno jediné médium, nikoli o celou řadu třídních knih, sešitů, či všelijakých papírů. Nutnost starat se o velké množství takovýchto knih, sešitů, či papírů s využitím elektronických nástrojů tedy odpadá. Odpadá také nutnost je zabezpečit proti, u žáků a studentů tolik oblíbenému, zcizení či zničení. Odpadá ale také situace, kdy nemají k požadovaným materiálům přístup všichni ti, kdo by to potřebovali, jelikož není pochopitelně možné, aby mělo jednu fyzickou knihu u sebe naráz více lidí.

Jediné, co je k práci s elektronickými nástroji potřeba, je počítač s nainstalovaným klientským softwarem, díky němuž je možné se kdykoli k potřebným materiálům dostat. Jsou-li pak navíc použité nástroje webové, odpadá i nutnost jakýchkoli instalací a spravování klientského softwaru, jelikož je do elektronické aplikace přistupováno skrze klasické webové prohlížeče. Aplikace podporují práci více uživatelů, s dostupností potřebných materiálů tedy není potíž. Samotná aplikace je poté umístěna na serveru, nejlépe v zázemí školy, ke kterému je omezený přístup, šance na fyzickou krádež či poškození dat jsou tedy malé. Jelikož jsou data samotná v elektronické podobě, je extrémně jednoduché je zálohovat, a možnost nenávratného poškození obsahu se tak ještě více redukuje.

Další nespornou výhodou softwarového řešení této problematiky je pak možnost kombinace různých nástrojů do jediného systému, což umožní učiteli dělat veškerou práci na jediném místě. To je výhoda nejenom proto, že se nemusí učitel starat o množství malých aplikací, také ale proto, že zároveň vidí veškeré informace v jejich vzájemných souvislostech, což může značně zkvalitnit jeho rozhodování.

Hlavní výhodou je však v podstatě stoprocentní automatizace při vytváření různých reportů, výkazů, či seznamů. Zatímco by musel například učitel dlouze kontrolovat třídní knihu jen proto, aby zjistil, kolik měli žáci ve třídě za měsíc absencí, lze tyto informace pomocí softwaru získat z elektronických dat během pár vteřin. S vhodně na-

psaným programem lze z elektronických dat obdobně získat prakticky jakýkoli výstup a znatelně tak urychlit učitelovu práci. Uspořený čas pak může učitel využít například ke kvalitnější přípravě výuky.

Mohlo by se zdát, že nemají z pohledu učitele elektronické nástroje snad žádné nevýhody; není to však zcela pravda, jelikož zde alespoň jedna, značně zásadní, nevýhoda existuje. Jde o nutnost se s takovým nástrojem naučit pracovat. Ač totiž může být daný nástroj sebelepší a nabízet uživateli nespočet kvalitních funkcí a usnadnění, nebude-li jej schopen, či ochoten, uživatel efektivně používat, nejenom že mu práci neulehčí, může mu ji ale dokonce i zkomplikovat. Takový nástroj pak zcela ztrácí svůj význam.

Výhod a nevýhod elektronické správy výuky je však nutné řešit nejenom z pohledu uživatele, nýbrž i z pohledu školy, která k využití elektronického řešení správy výuky přistoupí.

Mezi hlavní výhody tohoto přístupu patří pro danou školu opět především možnost značné automatizace správy výuky. Jedná se především o ulehčení typu programového řešení rozvrhů a suplování, které bývají již ze své podstaty často velkým problémem, či například značné zjednodušeného vytváření reportů o jednotlivých třídách, předmětech, práci učitelů apod. Možnosti typu zcela automatizovaných výstupů stavících na všemožných datech žáků, jako například generování vysvědčení, jsou pro školu jistě také velkou výhodou, jelikož se však nejedná o „*správu výuky*“, nebudou zde blíže rozebírány. Tím se bohužel výčet výhod značně redukuje, a ukazuje tak na fakt, že jsou elektronické nástroje správy výuky přínosem opravdu spíše pro učitele, či, jak bude zmíněno dále, žáky a jejich rodiče.

Není to pochopitelně způsobeno neexistencí výhod, ostatně již jich bylo několik jmenováno, nýbrž nutností takový systém, či nástroje, spravovat a vytvořit pro ně potřebné zázemí. Toto, spolu s nutností takový elektronický systém pro správu výuky zakoupit, ve většině případů, jsou bezesporu nevýhody, které musí daná instituce před rozhodnutím přistoupit k elektronickým nástrojům zvážit.

Poslední skupinou, u které je vhodné zmínit výhody a nevýhody využití elektronických nástrojů pro správu výuky jsou žáci a jejich rodiče. Dle názoru autora jsou to právě žáci a jejich rodiče, kteří při použití takových nástrojů získají nejvíce, resp. pro ně nemá takový přístup žádné nevýhody.

Mezi hlavní výhody patří možnost kdykoli sledovat aktuální průběh studia. Toto je výhoda pro žáky či studenty, hlavně pak ale pro jejich rodiče, jelikož tak mají nezkreslený přehled o známkách jejich dětí a důvodech k jejich udělení. Odpadají tak situace, kdy rodiče nemají přehled o skutečném množství absencí jejich dětí, či nevědí, kdy se konají školní akce, jako např. rodičovské schůzky, jim určené; ke všem těmto informacím se totiž mohou dostat skrze systém elektronická správy výuky školy. Klasická situace, kdy žáci „*ztrácí*“ žákovské knížky, či zatajují termíny rodičovských schůzek jen proto, aby se jejich rodiče nedozvěděli o jejich chabém prospěchu či záškoláctví, jsou díky tomu již nepředstavitelné.

1.2 Popis ideálního systému

Jelikož se představy o vlastnostech ideálního systému pro správu výuky mnohdy značně liší, je poněkud obtížné jej zcela definovat. Z pohledu autora je však možné definovat alespoň některé z jeho klíčových vlastností, jako jsou modularita, dostupnost, použitelnost, modifikovatelnost a komplexnost. Tato kapitola tedy alespoň krátce popíše každou z vlastností autorova ideálního systému.

První ze zmíněných vlastností je modularita. Dle autora nemůže být ideální systém vystavěn jako monoblok. Takový přístup by jej činil jednak zbytečně složitým na vytvoření, zároveň by ale zcela nepodporoval další z klíčových vlastností takového systému, a to modifikovatelnost. Ideální systém by mělo být zcela jisté možné jednoduše rozšířit o různé funkce, ideální systém tedy nemusí ve své podstatě ihned obsahovat vše, stačí, když bude dostatečně snadno rozšiřitelný. Právě z tohoto důvodu je nutné, aby byl daný systém postaven modulárně. Umožňuje tak nejen rozšířit svoji funkčnost do potřebné míry, zároveň ale také snadno odpojit funkce, které jsou nadbytečné.

Druhou důležitou vlastností ideálního systému je jeho dostupnost. Dostupností je zde myšlena především finanční dostupnost. Tato vlastnost je velmi důležitá z toho důvodu, že bude-li pořízení aplikace příliš finančně náročné, nebo bude např. svázáno s ne zcela přijatelnými licenčními podmínkami, jen těžko se taková aplikace rozšíří mezi uživatele, a bude prakticky využívána. Nebude-li ji pak samozřejmě nikdo používat, lze se jen těžko bavit o jakémkoli ideálu. Ideální systém by měl být tedy proto dostupný za smysluplnou cenu, či docela zdarma.

Třetí zmíněnou vlastností je použitelnost. Použitelností je zde myšlena použitelnost v rámci instituce, která má daný systém používat. Jde tedy hlavně o to, že musí systém odpovídat požadavkům takové instituce. Situace, kdy se musí změnit požadavky, či představy o použitelnosti, na systém, kvůli tomu, že není daný systém postavem právě pro ně, je zcela nepřijatelný. Ideální systém požadavky odráží, mění se podle nich, ne naopak. Tato vlastnost je těsně spojena s modifikovatelností, jelikož právě modifikovatelnost může použitelnost systému značně zvýšit.

Předposlední ze zmíněných vlastností je právě modifikovatelnost. Jak bylo již řečeno, může modifikovatelnost značně zvýšit použitelnost systému. K tomu, aby byl však systém plně modifikovatelný, musí mít jeho uživatel přístup k jeho zdrojovým kódům, nebo musí být tvůrce systému, za přijatelných podmínek, schopen a ochoten systém měnit dle konkrétních požadavků. Jelikož je však druhá z možností poněkud nereálná, přenáší totiž veškeré úsilí s modifikací, resp. se všemi modifikacemi, aplikace na jedinou osobu, resp. subjekt, je vhodnější přístup uvolnění zdrojových kódů, díky čemuž není již upravitelnost takového systému ničím a nikým omezena. Ideální systém by měl být tedy jinými slovy open source projekt.

Poslední z vlastností, které budou probrány, je komplexnost systému. Komplexnost systému je zde zmiňována především kvůli tomu, že ačkoli by měl být ideální systém systémem komplexním, poskytujícím tedy kvalitní řešení všech na něj kladených požadavků, neměl by být takový systém složitý. Poskytnutí přehršle funkcí na úkor ovladatelnosti systému, a tím pádem tedy i jeho použitelnosti, zcela jistě není ideální. Je zkrátka třeba dbát na to, že jej mají používat lidé. Požadavek na uživatelsky

příjemné ovládání aplikace by měl být tedy dodržován stejně důsledně, jako požadavek na funkčnost celého systému.

Závěrem je nutné dodat, že je výše popsaný soupis vlastností pouze jednou z možných variant *„ideálu“*. Ač je totiž pro autora klíčovým prvkem např. přístup ke zdrojovým kódům takového systému, a tedy možnost jeho naprosté modifikace, může být taková vlastnost pro někoho jiného zcela druhořadá. Výše popsaný ideál je tedy nutné brát opravdu pouze jako autorův pohled, nikoli univerzální pravdu.

1.3 Obdobné systémy na trhu

Situaci ohledně školních informačních systémů je relativně výstižně popsána ve článku Ondřeje Neumajera, Školní informační systémy [1]. Obecně lze však říci, že již v dnešní době existuje na českém trhu takovýchto systémů celá řada. Jako příklady jmenujme IS Gaudeamus [2], Bakaláři [3], eTřídnice [4], iŠkola [5], nebo například SAS [6].

IS Gaudeamus je poměrně mladý školní informační systém, který se snaží komplexně řešit školní administrativu. Jedná se o webový systém poskytující uživatli nástroje suplující žákovskou knížku, třídní knihu, nástroje starající se o suplování, nebo třeba evidenci žáků a učitelů. Aplikace je dostupná zdarma, platí se však za technickou podporu. Ač má mít aplikace "*intuitivní*" ovládání, vypadala její zkušební verze spíše chaoticky.

Bakaláři jsou v Čechách pravděpodobně nejrozšířenějším systémem pro správu školní administrativy vůbec. Školám nabízí vše od evidence žáků a zaměstnanců až po vytváření rozvrhů či evidenci úrazů. Jedná se o čistě komerční produkt. Dle některých uživatelů se stali Bakaláři příliš složitým systémem. Nejedná se o webovou aplikaci.

eTřídnice je pravděpodobně nejbližší současné verzi aplikace vytvořené pro tuto práci. Stejně jako vytvořená aplikace se, na rozdíl od předchozích řešení, zaměřuje převážně na třídní knihu. Jde o zcela komerční webový produkt, nabízí 30 denní zkušební verzi, a dle některých je to příliš jednostranně zaměřená aplikace.

iŠkola je poměrně komplexní systém nabízející školám nejen správu např. hodnocení, docházky, či rozvrhů hodin, ale i velmi zajímavou možnost vytvářet a spravovat online testy či domácí úkoly. Jde o webové komerční řešení, jeho cena je však přijatelná.

SAS je systém kompletně řešící školní administrativu. Nabízí nejenom funkce pro elektronickou správu výuky, nýbrž i nástroje pro správu majetku školy nebo třeba výpočet daní. Jedná se o komerční klient-server řešení s možností přístupu k datům aplikace i přes internet.

Pro bližší informace o jednotlivých systémech je doporučeno navštívit jejich oficiální webové stránky.

Praktická část

Jelikož bylo hlavním cílem této bakalářské práce vytvoření funkční webové aplikace suplující tradiční třídní knihu, je praktická část tohoto textu z pochopitelných důvodů věnována právě jejímu zpracování a možnostem jejího využití. V následujících kapitolách tudíž čtenář nalezne jak obecný popis zpracování dané problematiky, tedy popis vytvořené aplikace jako celku v souvislosti se zadáním této práce, tak přiblížení jednotlivých nástrojů a mechanizmů využitých při jejím zpracování a zdůvodnění jejich použití v dané podobě.

2 Popis aplikace

2.1 Obecný popis aplikace

Tato kapitola se zabývá obecným popisem vytvořené aplikace v souvislosti se zadáním bakalářské práce. Jde o popis řešení v zadání stanovených požadavků na aplikaci jako celku, nikoli o popis konkrétních mechanizmů aplikace.

Zprvu je vhodné přiblížit technické požadavky na aplikaci a jejich řešení. Ačkoli bylo v zadání stanoveno hned několik důležitých bodů, které musela aplikace splňovat, nebyly nijak definovány prostředky, jimiž by mělo být těchto cílů dosaženo. Pro tvorbu samotné aplikace byly tedy během návrhu vybrány následující nástroje a standardy – na Pythonu postavený Django web framework verze 1.2.5 spolu s odlehčenou databází SQLite3 jakožto základ celého projektu, XHTML 1.1 Basic a CSS 2.1 pro výstup vhodný i pro mobilní zařízení, knihovna JQuery 1.5 pro AJAX volání. Díky těmto prostředkům, a důrazu na zachování validity kódu, bylo docíleno splnění všech zadáním stanovených cílů. Výsledná aplikace tedy zaručuje prakticky stejné zobrazení v drtivé většině dnes využívaných webových prohlížečů, podporuje zobrazení na mobilních zařízeních a využívá technologii AJAX pro příjemnější uživatelský zážitek.

Dalším důležitým bodem popisu vytvořené aplikace jsou její funkce. Jelikož byl při tvorbě kladen důraz na dodržení zadání a držení se tématu práce, neobsahuje vytvo-

řená webová třídní kniha prakticky žádné funkce, které by nebyly vysloveně zmíněné v zadání, nebo z něho alespoň nevyplývaly. Vzhledem k tomuto faktu aplikace obsahuje pouze následující moduly – administrátorské prostředí, třídní knihu, zápis známek a přehled studia studenta.

Administrátorské prostředí slouží ke správě dat v aplikaci. Ačkoli lze skrze toto prostředí do databáze zadávat jakákoli data podporovaná databázovým modelem, není pravděpodobné, že by bylo využito pro samotné zalidnění databáze. Při případném reálném použití aplikace budou spíš uživatelská data do databáze nahrána dávkově, a toto rozhraní tak zůstane hlavně jako nástroj pro kontrolu dat a jejich případnou korekci, či jako nástroj pro správu uživatelů. Do tohoto rozhraní nemá z pochopitelných důvodů přístup nikdo jiný než administrátor.

Modul třídní knihy je pomyslným středem celého projektu. Právě tato část aplikace totiž zajišťuje hlavní požadavek zadání, a to suplování funkce klasické třídní knihy. Stejně jako klasická třídnice, i tento modul nabízí možnost vytvářet zápisy hodiny v závislosti na datu, třídě a vyučovaném předmětu, či možnost zapisovat absence žáků a studentů. Třídní učitel má pak přístup k detailnějším údajům jeho třídy, jako jsou především seznamy celkové absence třídy. Třídní učitel je také jediný, kromě administrátora samozřejmě, kdo může absence omlouvat. Pro přiblížení dalších možných výstupů z třídnice obsahuje detail třídy funkci generující pdf soubor se seznamem žáků. Ačkoli neobsahuje modul třídní knihy všechny funkce, které by mohla takováto aplikace ze své podstaty poskytovat, obsahuje to, co třídnice klasická; splňuje tedy zadání.

Jelikož byla evidence známek studentů, ač ne přímo spojená s myšlenkou třídní knihy, zařazena do požadavků na tuto aplikaci, musel vzniknout i modul zabývající se právě touto oblastí. Výsledkem je nástroj, který umožňuje učiteli rychlou orientaci v dosavadních známkách žáků z daného předmětu, dává mu pochopitelně možnost žáky známkami hodnotit, a navíc mu dovoluje zapisovat žákům poznámky.

Poslední částí tohoto projektu je zobrazení průběhu studia pro žáky či studenty, resp. jejich rodiče. Ačkoli nebylo toto rozhraní jedním z požadavků na výslednou apli-

kaci, bylo do webové třídnice zahrnuto z důvodu velmi dobré praktické využitelnosti. Toto rozhraní dává studentovi, resp. jeho rodičům, přehled o všech známkách, poznámkách a absencích daného studenta. Jednoduchou formou může ukázat průměry z předmětů či celkové počty zameškaných hodin. Obecně vzato je takovéto zobrazení logickou součástí jakékoli podobné aplikace, jelikož snadno spojí veškeré informace o žákovi a s vynaložením minima úsilí dá žákům, rodičům, kvalitní nástroj pro kontrolu průběhu studia.

2.2 Uživatelský popis aplikace

Tato kapitola je v podstatě krátkou uživatelskou příručkou k aplikaci, a je určena lidem, kteří s aplikací pracují poprvé, či si nejsou jisti jejím ovládáním.

Přihlášení a odhlášení

Přihlášení do aplikace se provádí kliknutím na tlačítko *Login* v pravém horním rohu Indexu aplikace. Po stisknutí tohoto tlačítka se uživatel dostane k zadání samotných přihlašovacích údajů, které potvrdí stisknutím tlačítka *Login* pod polem pro heslo. Přihlašovací jméno je uživatelovo evidenční číslo, a heslo je, nenechal-li si jej administrátorem změnit, X následované jeho rodným číslem, např. X8609122520. Po úspěšném přihlášení je uživatel přesměrován do hlavní stránky aplikace.

Odhlášení z aplikace proběhne stisknutím tlačítka *Logout* v pravém horním rohu kterékoli stránky aplikace. Po ukončení práce s aplikací je doporučeno se z ní touto cestou odhlásit.

Vytvoření a upravení zápisu hodiny

Pro zápis hodiny klikne učitel na tlačítko *Třídní kniha* v pravém horním rohu aplikace, čímž se dostane do modulu třídní knihy. Zde učitel vyplní třídu, se kterou hodlá pracovat, datum, a předmět, jenž vyučuje. Vybrané údaje potvrdí tlačítkem *Vyberte*. Všechny tři údaje jsou povinné.

Po tomto výběru se učiteli zobrazí dva sloupce dat. Levý obsahuje zápisy hodin pro daný den, pravý seznam absence pro daný den. Seznam zápisů obsahuje data o jednotlivých zápisech. Tedy číslo hodiny, datum, předmět, téma hodiny, poznámku (poznámka se zobrazí najetím kurzoru myši na písmeno *P*), a kdo zápis vytvořil.

Zápisy hodiny lze filtrovat výběrem umístěným pod seznamem samotným, a to na zápisy pro daný den, pro daný předmět, a všechny zápisy. Vybraný filtr uživatel potvrdí tlačítkem *Filtrovat*.

Pro vytvoření nového zápisu stiskne učitel tlačítko *Nový* vpravo pod seznamem, čímž se mu zobrazí formulář pro zadání zápisu. Učitel vybere číslo hodiny, pro kterou zápis vytváří, a vyplní téma hodiny. Má-li učitel k hodině nějaké poznámky, může je zapsat do podle poznámek. Pole *Č. hodiny* a *Téma hodiny* jsou povinná. Pro jednu danou hodinu, tedy kombinaci třída, datum, předmět, č. hodiny, lze vytvořit vždy jen jediný záznam. Po vyplnění všech povinných polí učitel potvrdí zápis tlačítkem *Zapsat* vpravo pod formulářem. Chce-li učitel zápis zrušit, stiskne tlačítko *X*.

Chce-li učitel předchozí záznam hodiny upravit, postupuje zcela obdobně. Vybere požadovaný zápis, a stisknutím tlačítka *Upravit* zobrazí formulář pro zápis hodiny; tento formulář obsahuje data daného zápisu. Upraví jej, a stiskne tlačítko *Zapsat*, či změny zruší tlačítkem *X*. Učitel může upravovat pouze své vlastní zápisy. Výjimkou k tomuto pravidlu je třídní učitel, který má právo měnit veškeré zápisy ve své třídě.

Vytvoření a upravení absence

Po vstupu do modulu třídní knihy a vyplnění třídy, data, a předmětu (viz předchozí bod) uvidí učitel v pravém sloupci seznam absencí. Tento seznam zobrazuje pouze absence pro daný den, a nelze v něm nijak filtrovat. Červená pole označená písmenem *X* jsou neomluvené absence, zelená pole označená písmenem *O*, jsou omluvené absence.

Učitel může do seznamu přidat absenci stisknutím tlačítka *Nová* pod seznamem absencí, čímž se mu zobrazí formulář pro zadání údajů absence. Ve formuláři učitel vybere ze seznamu jméno chybějícího žáka, a zaškrtne požadovaná pole hodiny, kdy

chyběl. Takto vytvořenou absenci potvrdí tlačítkem *Zapsat*, či zruší tlačítkem *X* umístěným pod formulářem.

Chce-li učitel upravit stávající absenci, vybere ji ze seznamu, a klikne na tlačítko *Upravit*. Zobrazí se mu tak již vyplněný formulář absence, který může upravit a následně, obdobně jako při vytváření nové absence, potvrdit tlačítkem *Zapsat*, či zruší tlačítkem *X*. Učitel má právo upravovat všechny absence. Stávající absence nelze v tomto zobrazení zcela smazat.

Vytvoření a upravení známek

Pro zápis známek klikne učitel na tlačítko *Zápis známek* v pravém horním rohu aplikace, čímž se dostane do modulu zápisu známek. Zde učitel vyplní třídu, se kterou hodlá pracovat, datum, a předmět, jenž vyučuje. Vybrané údaje potvrdí tlačítkem *Vyberte*. Všechny tři údaje jsou povinné.

Po tomto výběru se učiteli zobrazí dva sloupce dat. Levý obsahuje seznam žáků ve třídě, ten pravý seznam známek všech žáků. Seznam známek je řazen od nejmladších po nejstarší zleva doprava. Po najetí kurzorem myši na známku se učiteli zobrazí datum udělení známky a poznámka ke známce.

Pro zadání známky žákovi vybere učitel daného žáka ze seznamu vlevo a klikne na tlačítko *Vyberte žáka* umístěné pod seznamem. Seznam známek třídy tím bude nahrazen detailním seznamem známek žáka, nahoře, a seznamem jeho poznámek, dole. Zde klikne učitel na tlačítko *Nová*, čímž zobrazí formulář pro zadání známky. Ve formuláři je učitel povinnen vybrat známku ze seznamu, a zapsat, za co byla známka udělena. Takto vyplněný formulář je potvrzen stisknutím tlačítka *Zapsat*, či zrušen tlačítkem *X*.

Chce-li učitel stávající známku upravit, postupuje obdobně. Vybere známku ze seznamu žákových známek, klikne na tlačítko *Upravit*, čímž zobrazí formulář obsahující data známky, upraví jej, a potvrdí tlačítkem *Zapsat*, či zruší tlačítkem *X*. Učitel má právo upravovat pouze známky, které sám zadal.

Vytvoření a upravení poznámek

Po vstupu do modulu zápisu známek, vyplnění třídy, data, předmětu, a vybrání žáka ze seznamu (viz předchozí bod) uvidí učitel v pravém dolním sloupci seznam poznámek žáka. Tento seznam zobrazuje veškeré poznámky žáka. Červená pole označují negativní poznámky, zelená pozitivní, a bezbarvá neutrální.

Pro zápis poznámky stiskne učitel tlačítko *Nová*, čímž se mu zobrazí formulář pro zápis poznámek. Učitel zapíše text poznámky, a může navíc pomocí výběru karmy v horním pravém rohu formuláře určit zda se jedná o pozitivní či negativní poznamků. Nebude-li karma vybrána, jde o neutrální poznámku. Vyplněný formulář je potvrzen stisknutím tlačítka *Zapsat*, či zrušen tlačítkem *X*. Poznámka je povinná, karma nikoliv.

Pro úpravu poznámky je nutné poznámku ze seznamu vybrat, stisknout tlačítko *Upravit*, upravit poznámku v zobrazeném formuláři, a potvrdit či zrušit tlačitky *Zapsat* či *X*. Učitel může upravovat pouze poznámky, které sám zadal.

Detail třídy a omluva absencí

Je-li učitel třídním učitelem ve třídě, se kterou právě pracuje v modulu třídní knihy, má možnost vstoupit do detailního zobrazení dané třídy. Toto učiní kliknutím na tlačítko *Detail* v pravém horním rohu modulu třídní knihy. Po vstupu do detailu je se učiteli zobrazí odkaz na soubor seznam_zaku.pdf, který obsahuje automaticky generovaný seznam žáků, a pod ním seznam absencí celé třídy.

Seznam absencí obsahuje údaje o absencích jednotlivých žáků, kteří ve třídě někdy chyběli. Vedle jejich jména jsou pak zobrazeny údaje o jejich absencích daný den, absencích za posledních sedm dní, daný měsíc, a všech absencích. Jednotlivé sloupce obsahují hodnotu ve tvaru 120h (6/114), která znamená, že žák chyběl celkem 120 hodin, z čehož je jich 6 omluvených a 114 neomluvených. Po najetí kurzorem na danou hodnotu se zobrazí seznam dní, kdy žák ve vybraném období chyběl.

Chce-li učitel omluvit absenci některého ze žáků, vybere žáka ze seznamu, a potvrdí výběr tlačítkem *Vyberte* pod seznamem. Vpravo od seznamu absencí třídy se

tak zobrazí seznam dní za poslední týden, kdy vybraný žák chyběl. Rozsah tohoto seznamu lze však filtrovat na absence za poslední týden, tento měsíc, nebo všechny absence. Filtr je potvrzen stisknutïm tlačítka *Filtrovat*.

Pro samotné omluvení absence je potřeba vybrat datum absence ze seznamu absencí žáka a stisknout tlačítko *Omluvit absenci*, což zobrazí formulář pro omluvení absencí. V tomto formuláři jsou zobrazeny neomluvené absence pro vybraný den. Pokud chce některé z nich učitel omluvit, zaškrtne je ve formuláři a omluvení absencí potvrdí tlačítkem *Omluvit*. Omluvení lze zrušit tlačitkem *X*.

Přehled studia

Pro vstup do modulu přehledu studia klikne přihlášený student na tlačítko *Přehled studia* v pravém horním rohu aplikace.

Po vstupu do modulu mu budou zobrazeny tři sloupce dat. Největší, levý, sloupec je seznamem studentových známek za poslední měsíc, vpravo jsou pak, nahoře, seznam poznámek studenta a, dole, seznam absencí studenta. Toto zobrazení je čistě informativní, student zde tedy nemůže nic zadávat.

Seznam známek je možné filtrovat na seznam známek za poslední měsíc, nebo seznam známek jednotlivých předmětů. Výběr filtru je potvrzen tlačítkem *Filtrovat*. Seznam obsahuje datum zadání známek, předmět, ze kterého student známku dostal, poznámku, za co student známku dostal, známku samotnou, a kdo známku zadal.

Seznam poznámek obsahuje všechny studentovy poznámky. Poznámky jsou barevně rozděleny na negativní, červené, pozitivní, zelené, a neutrální, bezbarvé.

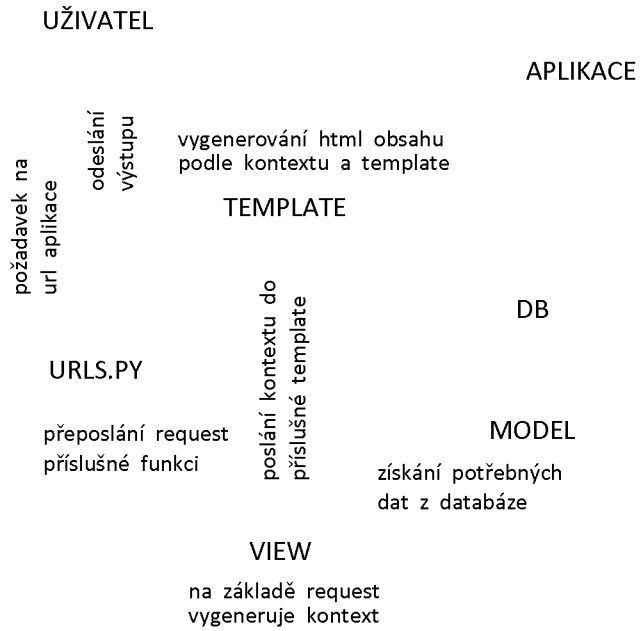
Seznam absencí obsahuje dvě pole. Vrchní pole zobrazuje počty absencí v daný den, za posledních sedm dní, daný měsíc, a všechny absence. Jednotlivé sloupce obsahují hodnotu ve tvaru 120h (6/114), která znamená, že žák chyběl celkem 120 hodin, z čehož je jich 6 omluvených a 114 neomluvených. Jednotlivé období lze vybrat a stisknutím tlačítka *Zobraz detail* tak zobrazit detailní seznam absencí pro dané období.

3 Použité technologie

Tato kapitola se věnuje nástrojům a technologiím použitým při vytváření popisované elektronické třídní knihy. Jedná se totiž o nástroje natolik zajímavé, že si alespoň kratší popis zaslouží. Pro kompletní popis a příklady použití však navštivte oficiální dokumentaci jednotlivých nástrojů.

3.1 Django

„Django je na Pythonu postavený web framework, který podporuje rychlý vývoj a čistý, pragmatický, design.“[7]. *„Django web framework následuje tzv. MVC (Model – View – Controller) architekturu, jelikož je však její interpretace v Djangu poněkud odlišná od podobných frameworků, jako např. Ruby on Rails, je Django nazýván spíše tzv. MTV (Model – Template – View, princip fungování lze vidět na obrázku č. 1) frameworkem.*“[8]



Obrázek č. 1 – Schéma MTV architektury

Mezi jeho hlavní výhody pak patří například možnost napsat celý databázový model v Pythonu, API pro dynamický přístup k databázi, přibalené administrátorské

prostředí odrážející navržený datový model, mocnou, rozšiřitelnou a hlavně uživatelsky příjemnou template language, hezké url adresy a mnoho, mnoho dalšího. Django se také snaží o maximální automatizaci při tvorbě aplikací a klade důraz na DRY (Don't Repeat Yourself) přístup. Django je od roku 2005 pod open source licencí.

Další zajímavostí Django frameworku je také jeho extrémně jednoduché naučení, a ohromné možnosti využití. Je tomu tak jednoduše proto, že *„Django je Python. Co jde v Pythonu, jde v Djangu, možnosti jsou tedy bezmezné.“*[9] Také to znamená, že naučit se Django je jen otázka *„naučení se programování v Pythonu a pochopení, jak fungují Django knihovny.“*[8]

Jelikož je Django základem aplikace, nemá příliš význam popisovat jeho konkrétní využití. Konkrétně bylo totiž Django v aplikaci využito k tvorbě naprosto všeho.

Krátce před napsáním této práce bylo vydáno Django verze 1.3 přinášející řadu novinek, které by šly v aplikaci elektronické třídní knihy dobře využít. V době práce na samotné aplikaci nebyla však tato verze ještě dostupná a byla tedy využita verze 1.2.5. Více o Django web frameworku viz oficiální web projektu [7].

3.2 Python

Jazyk Python patří v současné době již mezi uznávané a důkladně prověřené nástroje. Jedná se o *„pozoruhodně mocný“*[10] dynamický interpretovaný programovací jazyk vyznačující se především svojí čistou, lehce čitelnou syntaxí, plnou modularitou, a *„very high level“*[10] datovými typy. *„Význačnou vlastností jazyka Python je také produktivnost z hlediska rychlosti psaní programů. Týká se to jak nejjednodušších programů, tak aplikací velmi rozsáhlých. U jednoduchých programů se tato vlastnost projevuje především stručností zápisu. U velkých aplikací je produktivnost podpořena rysy, které se používají při programování ve velkém, jako jsou například přirozená podpora jmenných prostorů, používání výjimek, standardně dodávané prostředky pro psaní testů a dalšími. S vysokou produktivností souvisí dostupnost a snadná použitelnost široké škály knihovních modulů, umožňujících snadné řešení úloh z řady oblastí.“*[11] Kromě toho je Python také velmi jednoduchý na naučení, má být dokonce považován

za jeden z nejlepších jazyků pro výuku programování, a je pod open source licencí, díky níž je volně využitelný a šiřitelný, i pro komerční využití.

Stejně jako u Djanga, nemá přiliš význam popisovat využití Pythonu v aplikaci. Jelikož je Django postavené na Pythonu, byl využit prakticky všude. Konkrétně byl využit k napsání veškeré aplikační logiky.

V současné době existují dvě podporované řady tohoto jazyka, konkrétně 2.x a 3.x, podle oficiálních zpráv se však řada 2.x po vydání verze 2.7 nedočká již žádných velkých změn. Jelikož není Django web framework v současné době s Pythonem řady 3.x kompatibilní, byl při tvorbě elektronické třídnice použit Python 2.7.8. Více o jazyce Python viz oficiální web projektu [12].

3.3 ReportLab Toolkit

„*ReportLab Toolkit je open source, v Pythonu napsaná knihovna pro tvorbu dokumentů ve formátu PDF. Jde o robustní, flexibilní, a časem prověřené řešení, které uživateli dovoluje snadno a rychle vytvářet komplexní dokumenty.*“[13]

Tato knihovna slouží v aplikaci k tisku pdf souboru se seznamem třídy. Ač je to funkce značně triviální, dobře ukazuje možnosti dynamického generování pdf souborů.

V elektronické třídnici je použita knihovna verze 2.5. Více o ReportLab viz oficiální web projektu [13].

3.4 jQuery

„*jQuery je JavaScriptová knihovna určená ke zjednodušení HTML skriptování na straně klienta. Je podporována všemi webovými prohlížeči a v současné době je to nejpopulárnější JavaScriptová knihovna vůbec. Její syntax byla navržena tak, aby zjednodušila navigaci v HTML dokumentech, výběr DOM elementů, vytváření animací, zpracovávání událostí, a tvorbu AJAX aplikací. jQuery je licencovaná pod MIT licencí a GNU veřejnou licencí, jde tedy o open source nástroj.*“[14]

jQuery byla v aplikaci třídní knihy použita pro zlepšení uživatelského prostředí. V jQuery byla napsána veškerá AJAX volání, je pomocí ní ale i značně zpříjemněna práce s výběry ze seznamů. Konkrétně bylo díky jQuery dosaženo možnosti označovat pole tabulek v celé ploše pole, nikoli pouze samotným radio buttonem. jQuery také posloužila při zvýraznění vybraných polí tabulek.

jQuery je jediný nástroj, který je v elektronické třídnici zcela volitelný. Třídní kniha je stoprocentně funkční i bez využití JavaScriptu. Více o jQuery viz oficiální web projektu [15].

3.5 AJAX

„AJAX (Asynchronous JavaScript and XML) je obecné označení pro technologie vývoje interaktivních webových aplikací, které mění obsah svých stránek bez nutnosti jejich znovu načítání. Na rozdíl od klasických webových aplikací poskytují uživatelsky příjemnější prostředí, ale vyžadují použití moderních webových prohlížečů.“[16]

AJAX byl v aplikaci využit pouze pro zpříjemnění práce s aplikací. Konkrétně se stará o komunikaci mezi prohlížečem uživatele a serverem aplikace bez nutnosti znovu načítání webových stránek, se kterými uživatel pracuje. Ač není tato funkce nijak zásadní, práce s aplikací je s ní mnohem příjemnější.

4 Využití aplikace

Předchozí kapitoly ukázaly, jak by měla vypadat ideální aplikace pro komplexní správu výuky, jaké takovéto systémy jsou v současné době nejpoužívanější, a jak v kontrastu s nimi vypadá aplikace vytvořená v této bakalářské práci. Z tohoto kontrastu by se bohužel mohlo zdát, že je vzniklá aplikace něčím nedostačujícím, nebo dokonce nepotřebným. Následující dvě kapitoly se pokusí tento náhled na věc vyvrátit. Jejich cílem je totiž dokázat, že je aplikace i přes svoji jednoduchost již v její současné podobě velmi dobře prakticky využitelná, a zároveň obsahuje prostor pro rozšíření, která mohou její praktické využití ještě znásobit.

4.1 Využití stávající aplikace

Tato kapitola čtenáři přiblíží hlavní druhy využití stávající elektronické třídní knihy. Ač nebyla aplikace míněna jako komerční záležitost, může už ze své podstaty komerčním aplikacím alespoň částečné konkurovat. Důvodem je to, že je celá aplikace postavená na open source nástrojích, a sama je míněna jako open source projekt. Možné směry jejího využití jsou tedy jednak pochopitelně použití jako nástroj zastupující, resp. spíš doplňující, klasickou třídní knihu, hlavně však použití jako prototyp ukazující možnosti a obtížnost vytvoření takovéto aplikace jinými zájemci.

4.2 Doplnění klasické třídní knihy

Ač není pravděpodobné, že by byla klasická třídní kniha ze škol jakýmkoli softwarovým nástrojem ještě během několika příštích let zcela vytlačena, již v současné době je běžné, že je klasická třídnice softwarem doplněna. Současná situace potom vypadá většinou tak, že, rozhodne-li se vedení školy přistoupit k používání některého ze softwarů určeného ke správě výuky, musí kvůli jeho použití vynaložit nezanedbatelné úsilí pro jeho bezproblémové zavedení, a pochopitelně často finanční prostředky pro koupi samotné licence. V takovéto situaci by mohla navrhovaná elektronická třídnice vyniknout, jelikož se od výše zmíněného modelu liší ve dvou zásadních aspektech.

Jak bylo zmíněno v úvodu kapitoly, je celý projekt míněn jako open source aplikace přístupná všem, odpadá zde tedy pochopitelně nutnost platit jakékoli množství peněz za jeho využívání, či použití jeho částí v jiných projektech. Ač nedosahují ceny komerčních řešení vyloženě monstrózních rozměrů, jak je tomu bohužel zvykem v mnoha jiných odvětvích, je díky současné napjaté finanční situaci většiny školských zařízení nulová cena vítána a zcela jistě pozitivně přispěje k výběru právě navrhované aplikace. Obecně vzato může takováto cena také snadno sloužit jako reklama a podporovat tak nejenom další rozšíření aplikace, nýbrž i rozšiřovat myšlenku vytváření nových obdobných projektů, viz kapitola 4.1.2 Prototyp.

Druhým významným rozdílem oproti komerčním aplikacím je možnost libovolné úpravy aplikace. Jelikož mají různé školy různé požadavky, vzniká při použití jakéhokoli obecně navrhnutého softwaru jistý problém s jeho využitelností, pokud software daným požadavkům nevyhovuje. Většinou pak nezbývá nic jiného než přístup obrátit, a své požadavky přizpůsobit softwaru. Tento postup je podle nás pro uživatele minimálně nepříjemný, opět ovšem ukazuje, kde může vytvořená aplikace vyniknout. Nebude-li totiž aplikace svoji funkčností vyhovovat, a zároveň nebude mít uživatel zájem vytvořit zcela novou vlastní aplikaci, jak je popsáno v kapitole 4.1.2 Prototyp, není nic jednoduššího než stávající třídní knihu modifikovat. Toto je možné hlavně z toho důvodu, že je zdrojový kód celé aplikace veřejné dostupný, není tedy problém do aplikace přidat v podstatě jakékoli potřebné funkce. Fakt, že je aplikace ve své podstatě značně jednoduchá a dobře zdokumentovaná, možnostem její modifikace ještě napomáhá. Kdokoli se základní znalostí jazyka Python a Django web frameworku je tedy schopen aplikaci snadno a rychle přizpůsobit svým potřebám.

4.3 Prototyp pro obdobné projekty

Z předešlého textu je vidět, že aplikace by si i ve své stávající, ač poněkud omezené, podobě reálné využití najít mohla. Mnohem pravděpodobnější, a zároveň žádanější, využití je však jako prototyp pro jiné projekty s podobnou tématikou.

Důvodů, proč by měli uživatelé, tedy školy, alespoň zauvažovat nad možností tvorby jejich vlastní aplikace je několik. Za první skupinu lze považovat důvody již zmíněné, tedy finanční náročnost komerčních řešení, a nemožnost jejich modifikace pro konkrétní podmínky použití. Druhou skupinu pak tvoří fakta z předchozího vyplívající. Hlavně tedy skutečnost, že jsou uživatele mnohdy nuceni své požadavky měnit jednoduše proto, že je není schopen zakoupený produkt uspokojit. Byl-li by si však uživatel schopen vytvořit vlastní aplikaci, tyto věci by řešit nemusel, jelikož by takový produkt pochopitelně nekupoval, mohl by jej jakkoli modifikovat, a hlavně by díky aplikaci vytvořené tzv. na míru nemusel slevovat ze svých požadavků.

Právě možnost vytvoření aplikace, která opravdu odráží potřeby uživatele, je asi největší z důvodů proč se touto cestou vydat. Aplikace se totiž musí přizpůsobovat požadavkům, ne naopak. Přístup, kdy uživatelé sleví ze svých požadavků jen proto, že daná aplikace nenabízí přesně to, co by potřebovali, je podle nás naprosto špatný a uživatelé by se ho měli vyvarovat.

Přistoupí-li uživatel na myšlenku vytvoření vlastní aplikace, musí se však na vývoj připravit. Ačkoli není totiž takovýto projekt finančně náročný, a přináší ve finále velmi dobré výsledky, není zcela zadarmo. Je tomu tak především kvůli času, který je potřeba vývoji věnovat, a většinou bohužel také kvůli nutnosti přesvědčit vedení instituce o výhodách, a reálnosti dokončení, takového projektu.

S vyřešením většiny těchto komplikací pomůže právě využití prototypu. Dobu naprogramování aplikace lze totiž dobře zredukovat, nebude-li jejím vývoj začínat zcela od nuly. Jelikož byly v popisované aplikaci již všechny podstatné problémy vyřešeny, není jediný důvod, proč ji jako požadovaný prototyp nevyužít. Doba vývoje se tím značně zkrátí, samotný vývoj se zjednoduší, a výsledná aplikace bude podstatně kvalitnější, jelikož budou mít její návrháři díky existujícímu prototypu již od začátku dobrou představu o tom, kterým směrem zaměřit své úsilí.

Použití prototypu může však pomoci ještě před zahájením samotného vývoje aplikace. Není-li totiž vedení instituce nakloněno tomuto úsilí, nemá ani ten nejlepší projekt šanci na úspěch, resp. dokončení. Mezi zdroje takovéto nevole patří především

mylný, ovšem o to rozšířenější, názor, že musí být vývoj nové aplikace nutně náročný a spojen s nezanedbatelnými finančními prostředky, teoreticky vyššími než koupě licence komerčního řešení. Existující prototyp v takové situaci jednoduše ukáže opak. Prototyp postavený čistě na open source zdrojích nemůže být již ze své podstaty finančně náročný, a jelikož byl prototyp, míněno vytvořená aplikace, zároveň vytvořen jediným člověkem během pouhých pár týdnů, je zřejmé, že ani náročnost jeho vytvoření nemůže být velká. V takovém světle se již vytvoření aplikace obtížné zdát nemůže.

Vezme-li se ještě v potaz fakt, že má drtivá většina škol vlastní hardwarové zázemí a zkušené IT pracovníky, kterým může být takový projekt zadán, je myšlenka vytvoření vlastní aplikace více než lákavá.

4.4 Open source projekt

Aby mohla aplikace splnit výše zmíněné cíle, stát se tedy buď prototypem pro obdobné projekty, či posloužila již ve své současné podobě, bylo nutné zajistit, aby byla veřejné dostupná; aby byla open source projektem. Z tohoto důvodu byl kompletní zdrojový kód aplikace uvolněn pod GNU GPL v2 licencí na serveru SourceForge, zabývajícím se právě sdílením aplikací a kódu v open source komunitě. GNU GPL v2 licence byla zvolena z toho důvodu, že se autor snaží vytvořit volně šiřitelnou, tedy nekomerční, aplikaci a rád by docílil toho, že budou případné aplikace postavené na této postupovat stejně. Aplikaci lze v současné době na tomto serveru najít na URL http://sourceforge.net/projects/weirdradish.u/files/.

Do budoucna však plánuje autor zdrojové kódy aplikace umístit i na jiné servery sdílející open source aplikace, aby se vytvořené elektronické knize dostalo pokud možno co největší publicity, a zvýšila se tak šance na její praktické využití. Dle mínění autora není totiž důležité ani tak to, jak bude aplikace případně využita, např. bude-li aplikace využita jako součást větší aplikace, či bude-li využita pouze část jejího kódu, nýbrž to, že bude *nějak* využita.

Nevyužít vytvořený kód prakticky, či nedat ostatním možnost jej prakticky využít, by byla totiž bezesporu velká chyba.

5 Možnosti rozšíření aplikace

Jelikož není využití aplikace z dlouhodobého hlediska dáno pouze tím, co nabízí v době svého vypuštění, nýbrž i tím, jaký má potenciál pro další rozšíření a modifikace, bude se tato kapitola zabývat právě možnými rozšířeními popisované třídní knihy. Rozebrány zde budou jednak některé z funkcí, jež by mohly aplikaci vhodně doplnit a obohatit, dále pak ale také obecné možnosti zapojení modulu třídní knihy do většího systému, či samotná modularita aplikace, bez které by takovéto, v podstatě libovolné, rozšiřování nebylo dostatečně jednoduše proveditelné.

5.1 Oblasti možného rozšíření aplikace

I když bude na problém rozšiřování funkčnosti nahlíženo pouze z úrovně modulů, lze takových rozšíření najít celou řadu. Vzhledem k tomu, že je však stále většina potenciálních uživatelů ve stádiu, kdy musí vedle elektronické třídní knihy pracovat ještě s třídnicí klasickou, budou mezi nejpraktičtější úpravy patřit ty, které uživatelům s touto dvojitou prací pomohou, či jejich práci zautomatizují. Krásným příkladem funkcí, které pracují právě takovým způsobem, jsou již existující funkce v modulu třídní kniha, konkrétně počítání celkové absence žáků, či generátor pdf se seznamem žáků. Zcela stejný princip lze totiž aplikovat i na mnoho jiných procesů, se kterými by učitel za běžných okolností mohl trávit hodiny, jež ovšem počítač zvládne v řádu vteřin. Mezi dobré příklady takovýchto funkcí patří výčty odučených hodin v jednotlivých třídách či předmětech, různé seznamy žáků ve spojitosti s jejich výsledky, nebo například generování dat pro vysvědčení. Trochu jiným druhem úprav jsou pak úpravy funkcí stávajících, mezi které lze zařadit například doplnění vah známek, či možnost zapsat poznámku k absenci žáka. Všechny tyto funkce by modul elektronické třídní knihy nejenom vhodně doplnily, zároveň by také ale nebyl problém je vytvořit. U všech se totiž jedná o pár řádků kódu, a navíc všechny pracují s údaji, které již v aplikaci jsou.

Bude-li ovšem vycházeno z kapitoly 1.2 Popis ideálního systému, a bude na aplikaci nahlíženo jako na součást většího celku, zjistíme, že je oblastí, kde by mohla být

vylepšena, nejenom celá řada, také že ale možné změny začínají dosahovat nemalých rozměrů a efektivně dělají z aplikace docela jiný projekt. Důvodem k takovému stavu není samozřejmě jakákoli chybovost dané aplikace, zde je totiž nutné rozlišit mezi úpravami či vylepšeními, a opravami, nýbrž fakt, že nebyla popisovaná aplikace nikdy navrhována jako systém snažící se vyřešit vše. Aplikace byla vždy míněna právě pouze jako jedna součást, ačkoli samostatně funkční, systému většího.

Logický krok ke zlepšení je pak tedy pochopitelně tvorba oné vnější konstrukce, onoho komplexního systému, jehož bude aplikace součástí. Tento krok sám o sobě modul třídní knihy příliš neovlivní, vytvoří však základy pro jeho zakomponování v rámci jiných modulů, čímž se celý projekt posune ze stádia „*jednoduchého nástroje*“ blíže „*komplexnímu systému*“.

Podaří-li se vytvořit takovouto nosnou konstrukci, je již projekt modifikovatelný v podstatě do jakékoli míry a jeho komplexnost, tedy možnosti, jež systém nabízí, je pak limitována pouze vynalézavostí vývojářů při vytváření jednotlivých modulů. Mezi tyto moduly může patřit prakticky cokoli od modulů zajišťujících tvorbu všelijakých reportů, či třeba generování vysvědčení, až po moduly starající se o rozvrhy a obecné plánování výuky.

Jak je z předešlého textu zřejmé, možných úprav stávající aplikace je přehršel. Některé mění stávající funkce, jiné nové funkce přidávají, další pak aplikaci připravují na přerod v mnohem obsáhlejší systém. Obecně lze však říci, že záleží jen a jen na požadavcích a možnostech uživatele.

5.2 Modulární přístup

Předchozí text nastínil možnosti dalších rozšíření stávající aplikace. Také ale ukázal, že, je-li zde ambice rozšířit tuto aplikaci z její současné podoby do podoby komplexnějšího systému, je nutné při návrhu budovaného systému počítat s využitím modulárního přístupu. Ač je totiž možné vytvářet takto komplexní aplikaci formou jediného monobloku obsahujícího veškerou funkcionalitu, není to zdaleka řešení nejjednodušší, a už vůbec ne nejelegantnější. Vývoj takové aplikace lze jen těžko rozdělit

do dílčích částí, které by bylo možné psát a testovat samostatně. Z aplikace se tak stává nekontrolovatelný moloch, který již svou formou podporuje zanášení chyb do kódu, a jehož jakékoli modifikace jsou bez velmi důkladné dokumentace mnohonásobně komplikovanější a zabírají zbytečně mnoho času.

Modulární přístup na druhé straně dovoluje nejenom kvalitnější a rychlejší vývoj a testování, jak ale dobře ukazuje sama aplikace, jelikož již v současné podobě není oním jediným monoblokem, nemusí být takový návrh nijak složitý a navíc nabízí značné výhody při pozdějším rozšiřování či úpravách. To co je zde pojmenováno „*elektronická třídní kniha*“ je totiž kombinace čtyř různých modulů, konkrétně modulu administrátorského zobrazení, modulu zápisu známek, modulu přehledu studia a nakonec právě modulu třídní knihy. Celá aplikace je tedy rozložitelná na samostatné, a přesto zcela funkční, dílčí části, které lze použít v tomto, nebo třeba v docela jiném projektu. Ačkoli lze při bližším zkoumání kódu aplikace zjistit, že jsou jednotlivé moduly přece jenom minimálně propojeny skrze soubor models.py definující databázový model, je myšlenka modulárního přístupu zřejmá.

Vytvoření komplexnějšího systému postaveného na spolupráci několika různých modelů pak tedy neznamená o mnoho víc než vytvoření kvalitní společné konstrukce, tedy minimální společný databázový model, minimální společnou logiku, urls.py pro správné přesměrování požadavků a template pro minimální XHTML výstup celé aplikace, vytvoření logiky jednotlivých modulů, vytvoření jejich models.py vyžadují-li speciální objekty v databázi, jejich import do aplikace, a nakonec vytvoření template pro jejich správné zobrazení. Všechny takto vytvořené součásti jsou na sobě stoprocentně nezávislé, jsou samostatně funkční, a díky jejich oddělitelnosti se velmi dobře testují a upravují.

Modulární návrh aplikací je navíc podporován přímo použitým Django web frameworkem, nevyužít jej by byl tedy bezesporu krok špatným směrem.

6 Model

Jak bylo řečeno v kapitole 3.1 Django, je Django web framework tzv. MTV framework. *„M zde znamená Model, vrstvu pro přístup k datům. Tato vrstva obsahuje veškeré informace o datech – jak k nim přistupovat, jak je validovat, jak se chovají, a jaké mají mezi sebou vazby.“*[8] Následující dvě podkapitoly se budou věnovat právě datovému modelu elektronické třídní knihy. Nejprve budou rozebrány autorovy požadavky na datový model, poté samotné zpracování datového modelu, a na závěr možnosti jeho přizpůsobení nastane-li potřeba požadavky změnit.

Definici modelů lze nalézt na přiloženém nosiči v souboru /bp/trst/models.py.

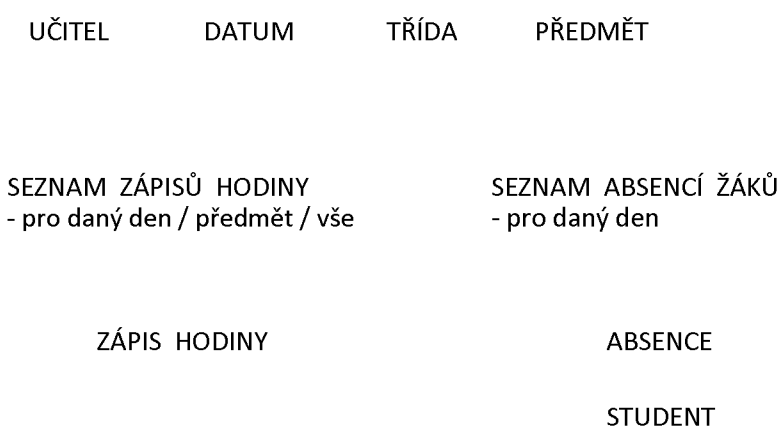
6.1 Požadavky na databázový model

Ještě předtím, než mohla začít jakákoli reálná práce na vytváření aplikace elektronické třídnice, musely být definovány požadavky aplikace na databázový model. Tyto požadavky přímo vyplývaly z funkcí, jež měla aplikace poskytovat.

Postup získání potřebných požadavků vypadal tak, že bylo vytvořeno funkční schéma jednotlivých částí aplikace, tzn., bylo vytvořeno schéma pro třídní knihu, pro zápis známek a pro přehled studia, díky čemuž vyšly na povrch požadované součásti jednotlivých modulů. Tyto požadavky byly následně spojeny a byla z nich vytvořena skupina požadovaných databázových modelu.

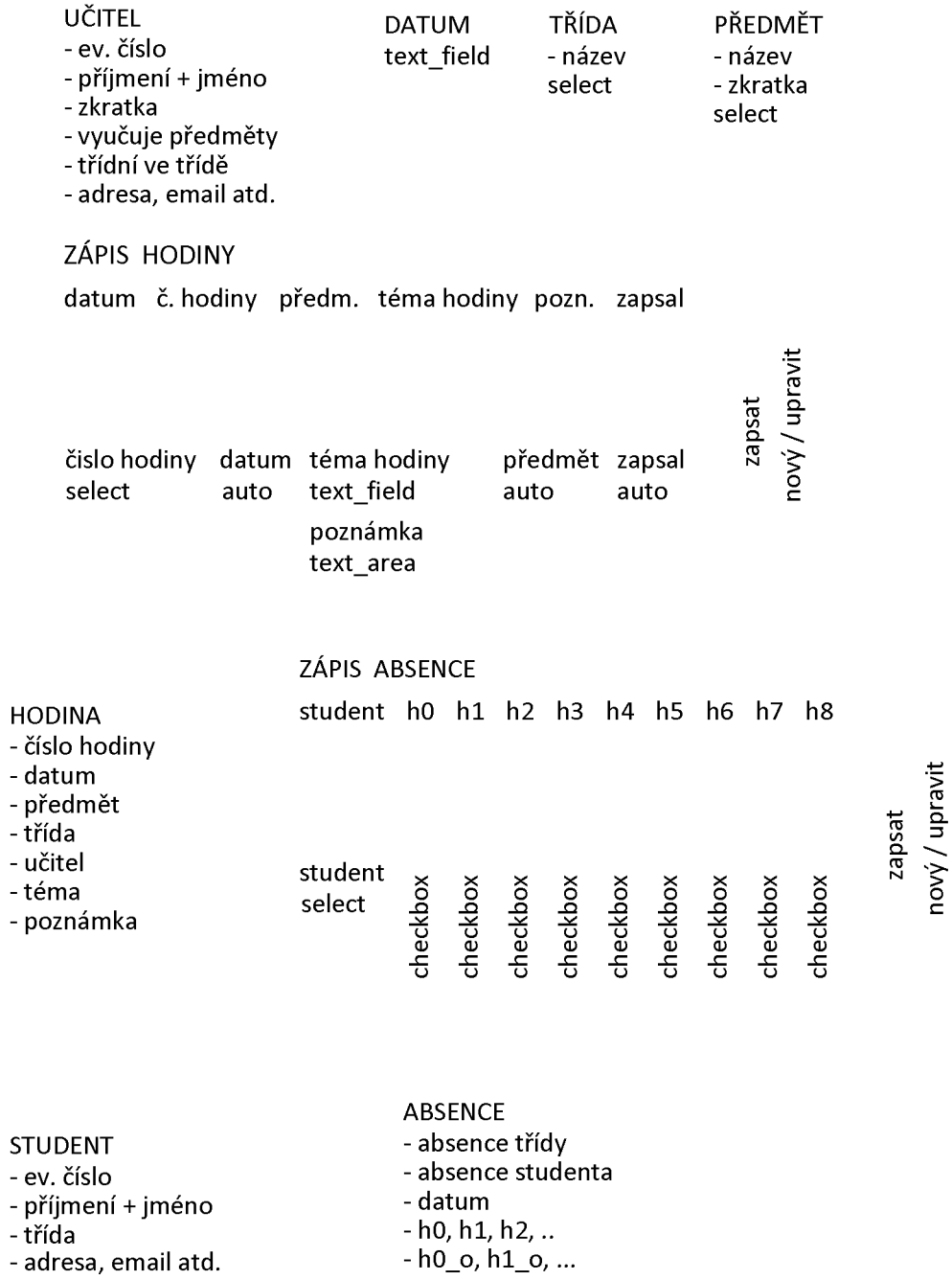
Tento postup lze předvést např. na modulu třídní knihy. Funkčnost této části aplikace lze zjednodušené popsat tak, že měla uživateli umožnit zapisovat, či upravovat, informace o jednotlivých hodinách a zapisovat či omlouvat absence žáků na daných hodinách. Učitel měl mít zároveň možnost prohlížet předchozí zápisy hodin, či předchozí zápisy pro předmět, který zrovna vyučuje. Absence se však pro přehlednost zobrazovaly vždy jen pro daný den. Z tohoto popisu lze vyčíst několik důležitých věcí. Učitel potřebuje nějakým způsobem definovat seznam zápisů hodiny, se kterým bude pracovat. Jedná se tedy buď o seznam zápisů pro daný den, nebo pro daný předmět, nebo o všechny zápisy v třídnici. Obdobně je potřeba definovat seznam absencí.

Z těchto dvou požadavků po jistém zamyšlení vyjde, že mohou být oba dva seznamy, tedy jak seznam zápisů, tak seznam absencí, definovány pomocí tří údajů – data, kdy učitel do třídnice zapisuje, třídy, ve které učitel právě učí, a předmětu, který učitel právě vyučuje. Tyto tři údaje jsou tedy klíčové pro vyfiltrování vhodného seznamu. Jelikož jde však o seznamy, musí obsahovat také jednotlivé položky. Pro seznam zápisů hodiny je potřeba definovat samotný zápis hodiny, a pro seznam absencí jednotlivé absence. Absence pak musí být ještě nějak vztažena k žákům, a žáci k nějaké třídě. Touto jednoduchou metodou z funkčního schématu modulu třídní knihy vyplynulo, že je pro jeho fungování potřeba definovat model předmětu, třídy, studenta, zápisu hodiny, absence, a samozřejmě model učitele. Výsledné schéma spolu s vyplývajícím mini databázovým modelem by pak mohlo vypadat například jako schéma na obrázku č. 2.



Obrázek č. 2 – Jednoduché schéma návrhu DB modelu

Pro korektnost je nutné zmínit, že je výše uvedený postup poněkud zjednodušený. Není samozřejmě nijak chybný, pouze neobsahuje veškeré kroky, které byly v praxi při návrhu databázového modelu učiněny. Ve skutečnosti vypadalo výsledné schéma totiž spíše jako obrázek č. 3. Jak je vidět, je jeho výstupem stejná skupina modelů, je však obohacen o množství dalších informací. Při reálné analýze byla totiž často definována velká část, občas dokonce všechny, vlastností jednotlivých modelů, jejich vztahy, a občas dokonce i vzhled.



Obrázek č. 3 – Reálné schéma návrhu DB modelu

Ze schématu pak tedy vyčteme jednak např. to, že má model Učitel vlastnosti jako jméno, přijmení, atd., také ale to, že je tento model propojen s modely Třídy a Předmět, nebo např. to, že má být při vytvoření nového zápisu v třídnici do formu-

láře automaticky doplněno id přihlášeného učitele. Obecně vzato lze říci, že vzniklo schéma na obrázku č. 3 stejným postupem jako schéma zjednodušené, autor si pouze musel klást více otázek. Nesnažil se při jeho vytváření tedy zjistit pouze jaké funkce má daný model poskytovat, a jak se tyto potřeby odrážejí v datovém modelu, nýbrž i otázky, jako jaké konkrétní vlastnosti budou modely mít, jak konkrétně budou muset být jednotlivé modely navzájem propojeny, aby potřebné funkce poskytly, či jakou formou se budou data zapisovat, tedy jaké vlastnosti budou mít konkrétní formuláře v daném modulu. Všechny tyto aspekty byly následně promítnuty do vytvořených Django modelů.

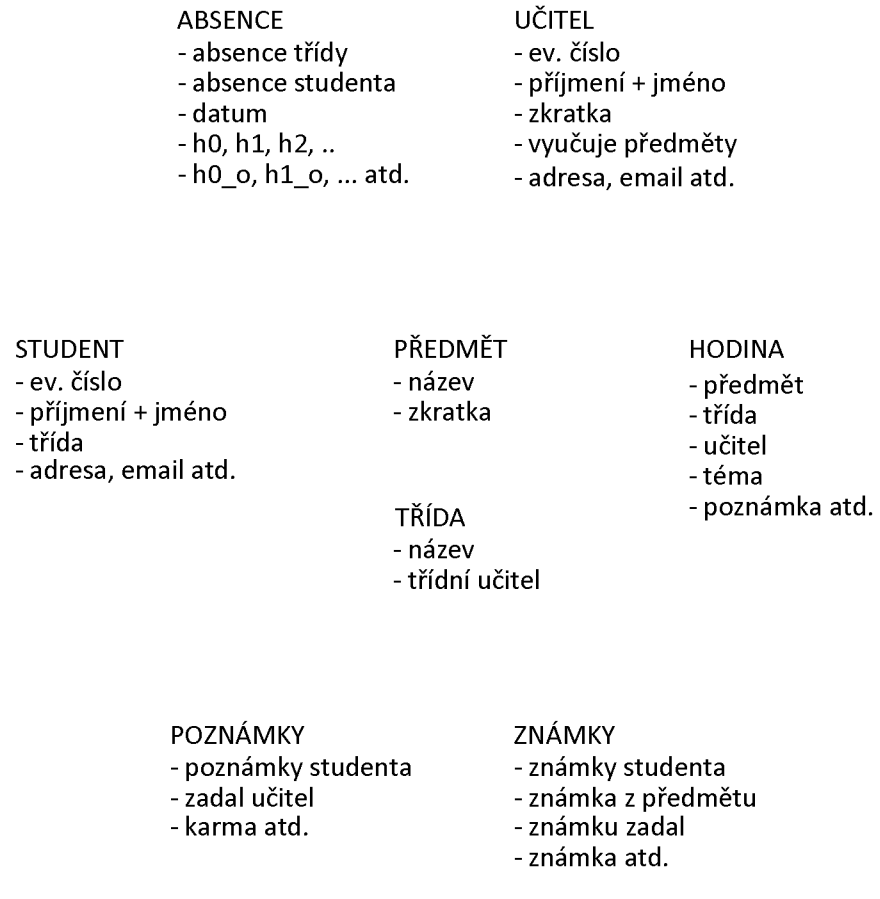
6.2 Vytvořené databázové modely

Spojením všech výstupů výše popsaného postupu byl vytvořen seznam Django modelů potřebných pro běh aplikace. Jedná se o pouhých osm modelů, konkrétné je to šest modelů již zmíněných, tedy předmět, třída, student, hodina, a absence, a dva modely potřebné pro zápis známek, jmenovitě známka a poznámka. Modul přehledu studia sám o sobě žádný speciální model nevyžaduje, pouze zobrazuje data z již existujících modelů.

Definice těchto modelů je zaznamenána v souboru models.py. Zajímavostí zde je, že, jelikož se Django snaží o maximální usnadnění práce, a je samozřejmě postavené na Pythonu, jsou jednotlivé modely zapsány jako pythonovské třídy. Každý model, tedy každá třída odvozená od django.db.models.Model, zastupuje relaci v databázi. Každý atribut takové třídy pak zastupuje sloupec v relaci. Tento přístup jednak opravdu zrychluje vytvoření definice celého databázového modelu, je také ale extrémně dobře čitelný. Vytvoření databáze na základě takovéto definice, nebo pouhý výpis potřebných SQL příkazů, je pak otázkou jediného příkazu v Shellu.

Rozebírat jednotlivé vytvořené modely by pravděpodobně nemělo příliš velký význam. Jednak jsou jejich kompletní definice k nahlédnutí ve zdrojových souborech, jsou ale také natolik jednoduché, že by byl jejich důkladnější popis opravdu nadbytečný. Pro zobrazení vztahů mezi jednotlivými modely je zde obrázek č. 4.

Za zmínku však stojí jeden z atributů modelu Znamka, konkrétně atribut *znamka* a jeho choices, resp. proměnná ZNAMKY_CHOICES. Zajímavostí zde je, že ačkoli povoluje Django vytvářet takovéto tuple možných voleb pro daný atribut, nedovoluje vytvořit tuple skládající se z dalších tuple, resp. pak takové tuple považuje za pojmenovanou skupinu. V případě této aplikace by byla taková vlastnost velice vítána, jelikož by bylo logické vytvořit pro atribut znamka možnosti mající tři, a ne pouze dvě, hodnoty. Místo současných možností stylu (7, '4-'), mající význam „*hodnota 7*", „*zobrazená známka 4-*", by bylo značně přijatelnější mít možnosti stylu (7, '4-', 4.5), mající význam „*hodnota 7*", „*zobrazená známka 4-*", „*float hodnota 4.5*". Tento přístup by poněkud zjednodušil počítání průměrů v modulu přehledu studia, kde je v současné době nutné známky následně ještě převádět z hodnoty (7, '4-') právě na (7, 4.5), aby bylo možné takovou hodnotu vůbec použít.



Obrázek č. 4 – Vztahy mezi vytvořenými modely

Kromě samotných modelů byla v models.py vytvořena ještě čtveřice tzv. receiver funkcí sloužících k jisté automatizaci při vytváření nových uživatelů. Uživatelé samy o sobě nejsou totiž součástí modelu aplikace. O uživatele, stejně jako o celý proces autentizace uživatelů, se stará Django modul django.contrib.auth. Vytvoření nového učitele nebo studenta tak tedy pochopitelně není to samé co vytvoření nového uživatele, jsou to dva oddělené modely. Právě z tohoto důvodu, tedy kvůli zjednodušení vytváření nových uživatelů, byly vytvořeny zmíněné receiver funkce. Jejich účel spočívá jednoduše v tom, že jsou připojeny k signálům označujícím vytvoření či smazání učitelů či studentů, a v případě, že takový signál přijde, vytvoří uživatele s potřebnými vlastnostmi. V takovém případě bude tedy při vytvoření studenta vytvořen aktivní uživatel patřící do skupiny „*je_student*“, zatímco při vytvoření učitele bude uživatel patřit do skupiny „*je_ucitel*“ (obě tyto skupiny je třeba před tvorbou jakýchkoli nových uživatelů potřeba vytvořit v administrátorském rozhraní). V obou případech bude pak jako uživatelské jméno použito jeho evidenční číslo, pro heslo uživatele bude použita kombinace „*X+rodné číslo*“, jméno a příjmení uživatelů bude pochopitelně převzato z příslušných modelů učitele a studenta. Při smazání učitele či studenta dojde k obdobnému procesu, akorát s opačným efektem.

6.3 Možnosti přizpůsobení db modelu

Stejné jako cokoli jiného v této aplikaci, resp. cokoli vytvořené v Djangu, je i databázový model velmi lehce upravitelný.

Jedná-li se o přidání zcela nových modelů do databáze, např. při přidání nových modulů do aplikace, spočívá celý proces úpravy databáze ve spuštění příkazu „python manage.py syncdb“, který synchronizuje databázi s databázovým modelem. O nic víc se nemusí uživatel starat.

Takováto synchronizace funguje bohužel pouze tehdy, přidávají-li se záznamy nové. Syncdb totiž nikdy nedá databázi příkaz ALTER TABLE. Oficiální dokumentace zdůvodňuje tento přístup snahou o neohrožení stávajících dat v databázi. Mění-li se totiž třídy v modelu, nebo schéma databáze, vzniká často jistá forma dvojsmyslnosti

ve významu takové změny. Django by v té chvíli muselo hádat, jak databázi správně upravit, což by mohlo mít fatální následky právě pro stávající data. Django však nenechá uživatele ani v této situaci zcela bez pomoci. Pomocí příkazu *sql* je možné vygenerovat novou strukturu SQL, což nám značně pomůže při ručních změnách v databázi.

6.4 Použitá databáze

V elektronické třídnici byla využita, k Pythonu přibalená, odlehčená SQLite3 databáze. „*SQLite je v C napsaná knihovna, jenž poskytuje odlehčenou, na disku umístěnou, databázi, která nevyžaduje vlastní server a umožňuje přístup k databázi pomocí nestandardní varianty SQL jazyka. SQLite je ideální pro použití v prototypu aplikace předtím, než bude její kód portován do větší databáze jako je PostgreSQL či Oracle.*“[17] Více o SQLite3 viz oficiální web projektu [18].

SQLite byla v projektu elektronické třídnice využita hlavně kvůli její snadné přenositelnosti, která značně usnadňovala práci při testování a předvádění aplikace na různých strojích. Jelikož není ovšem Django na druhu použité databáze v podstatě závislé, podporuje použití PostgreSQL, MySQL, SQLite3, a Oracle, nebude při praktickém využití aplikace nejmenší problém na plnohodnotnou databázi přejít. Takovýto přechod je dokonce doporučován.

7 View

„V znamená v MTV architektuře tzv. View, logickou vrstvu. Tato vrstva obsahuje logiku, která přistupuje k modelu, nad získanými daty může provádět jisté operace, a výsledky přeposílá do příslušné template. Dá se považovat za jistý most mezi modelem a template.“[8] Následující tři kapitoly se budou věnovat jednak krátce struktuře View v aplikaci, poté Django request.session, bez které by byla aplikace po nasazení na produkční server, např. Apache, prakticky nepoužitelná, a na závěr bude věnován prostor obecnému popisu View aplikace, zajímavostem, které pří jeho vytváření, resp. vytváření jeho dílčích částí, vyvstaly, a obecně principu jeho fungování.

7.1 Struktura View

Jelikož byla elektronická třídní kniha navržena modulárně, je pochopitelné, že není její View jediným blokem kódu. Každý modul aplikace má vlastní, zcela oddělitelný, soubor obsahující logiku pro jeho správné fungování. View jednotlivých modulů lze nalézt na přiloženém nosiči v souboru /bp/hlstr/views.py a ve složce /bp/trst/views/.

Takovéto modularity je dosaženo jednak vytvořením views jednotlivých modulů, tedy separovaných souborů s pythonovským kódem obsluhující jednotlivé části aplikace, ale také vytvořením tzv. pythonovského balíčku, package, jehož budou views moduly součástí. Takovýto balíček je v podstatě pouze složka obsahující inicializační soubor __init__.py a dané soubory s kódem. Pro zjednodušení práce s takovýmto balíčkem jsou pak jednotlivé views importovány do jmenného prostoru __init__.py, což z dané složky efektivně dělá modul views.py.

Možných přístupů k této problematice je samozřejmě několik. Obecně vzato však neexistuje jediný správný a nejlepší přístup; existují pouze dobrá či špatná řešení. Námi zvolené řešení lze však bezpochyby za jedno z těch dobrých považovat. Je díky němu nejenom dosaženo požadovaných výsledků, je ale i velmi lehce proveditelné, a navíc relativně elegantní.

7.2 Django sessions

Z návrhu aplikace vyplývá, že musí být pro její správné fungování během jednotlivých uživatelských relací nějakou formou zaznamenáno určité množství údajů o takové relaci. Jedná se především o údaje typu: byl-li daný uživatel vůbec autorizován pro přístup do dané aplikace, s jakými daty uživatel během své relace pracuje, nebo třeba jaké části webu se mu mají v závislosti k jeho aktivitám zobrazit.

Jako nejjednodušší řešení tohoto problému by se mohlo jevit použití klasických proměnných, resp. globálních proměnných. Jelikož se views.py chová jako jediný kus kódu, mohou všechny funkce k takto uloženým datům bezproblémově přistupovat či je měnit. Dokud bude v chodu interpret, není žádný očividný důvod proč daný problém řešit jinak.

Bude-li taková aplikace použita na vývojovém Django serveru, nenastane žádný problém. Údaje z jednotlivých relací se budou zapisovat do globálních proměnných, odkud k nim bude přistupováno jinými částmi kódu; taková proměnná, a údaje v ní obsažené, bude žít, dokud poběží vývojový server, tedy dokud poběží interpret Pythonu. Fráze *„dokud poběží interpret“* je zde však bohužel klíčová a zcela zásadně určuje použitelnost daného kódu na jiném nežli vývojovém serveru. Bude-li taková aplikace nasazena na serveru produkčním, lze tedy očekávat problémy. Například Apache server totiž spouští interpret Pythonu pouze když jej vyloženě potřebuje; přijde-li tedy na server požadavek, který má být skrze urls.py přesměrován příslušné funkci ve views.py, server kvůli úspěšnému obsloužení požadavku spustí interpret Pythonu, načte do něj views.py, které požadavek zpracují, odešlou výsledek dané operace, a Apache Python interpret opět ukončí. Veškeré proměnné, a data v nich obsažená, jsou tak ztraceny, a pochopitelně nepoužitelné. Právě tato zdánlivá *„zapomnětlivost“* aplikace běžící na produkčním serveru byla důvodem pro využití Django sessions.

„Django poskytuje plnou podporu anonymních relací. Session framework umožňuje ukládat a zpětně načítat jakákoli data pro každého jednotlivého návštěvníka aplikace. Data ukládá na straně serveru a zajišťuje odesílání a přijímání cookies. Cookies obsahují ID relace, nikoli data samotná.“[19] Django sessions jsou implementovány

skrze SessionMiddleware a defaultně fungují tak, že je v databázi po syncdb vytvořena relace django_session právě pro session data. Django sessions ale umožňují ukládat data i na disku či přímo v paměti. Každý HttpRequest má pak atribut session, který se chová jako dictionary-like objekt, do kterého lze data zapisovat, nebo je z něho naopak číst. Na rozdíl od klasického dictionary nemá session metodu update().

Díky využití sessions nejsou údaje o jednotlivých uživatelských relacích nijak vázány na interpret Pythonu, aplikace „*nezapomíná*“, a chová se tak dle očekávání i na produkčním serveru.

7.3 Popis views a jejich fungování

Při pohledu na kterýkoli z dílčích modulů views je zřejmé, že se nejedná no nic víc než sérii pythonovských funkcí. Oficiální dokumentace o Views říká následující. „*View funkce, nebo pro zkrácení pouze view, je jednoduchá pythonovská funkce, která přijímá web request a vrací web response. Tato odezva, response, může být HTML obsah webové stránky, přesměrování, 404 error, XML dokument, obrázek... nebo v podstatě cokoli jiného. Samotný view obsahuje logiku potřebnou k vytvoření požadované odezvy, a daný kód může být umístěn naprosto kdekoli, dokud je cesta k němu obsažena v Python path. Žádné další požadavky, žádná další takříkajíc magie, zde není.*“[20]

V případě elektronické třídní knihy pak existuje jedna funkce pro každou webovou stránku, kterou lze v aplikaci navštívit. Důvodem k tomu je jednoduše to, že neobsahuje aplikace žádný nadbytečný obsah; každá stránka dostupná z aplikace něco „*produkuje*“. Jedinou výjimkou proti tomuto pravidlu je pak funkce pdf v modulu tridni_kniha.py, která neobsluhuje žádnou stránku, nýbrž slouží k vygenerování pdf se seznamem žáků.

Funkce obsluhující jednotlivé stránky jsou pak složeny ze série podmínek, které jednak rozlišují jednotlivé uživatele, administrátora tedy přesměrují do admin rozhraní, studenty do přehledu studia, a učitele do třídní knihy, dále pak ale hlavně na základě obsahu POST dat z web requestu vytváří HTML obsah, který je poté odeslán zpět uživateli.

Tento princip vypadá v praxi tak, že učitel například v aplikaci zmáčkne tlačítko pro zobrazení seznamu zápisů v třídnici pro daný den, třídu a datum, tento požadavek bude serverem přeposlán urls.py, ty, jelikož se jedná o požadavek na url /trst/tridnice/, jej přepošlou funkci tridni_kniha, kde je rozebrán obsah POST dat daného požadavku. Pokud obsahují POST data řetězec „*vyber_tridy_button*“, což v tomto případě obsahují, vyhoví požadavku první podmínka. V ní se pak, pokud jsou data poslaná uživatelem validní, zkontroluje, zdali je přihlášený učitel třídním učitelem v dané třídě, a vytvoří se odpovídající boolean proměnnou ucitel_je_tridni, na základě vybrané třídy a data se z databáze získá seznam záznamů zápisů hodiny, pokud v databázi potřebná data nejsou, vytvoří se seznam prázdný, obdobně se vytvoří seznam absencí, a všechny tyto údaje se zapíší do request.session['context']. Jelikož ostatní podmínky v dané funkci nevyhoví, postoupí proces do fáze vytvoření kompletního kontextu, který bude posléze poslán do template /trst/tridni_kniha.html, která bude zobrazena uživateli. request.session['context'] slouží v této fázi pro update kompletního kontextu.

Prakticky zcela stejným způsobem, jako výše popsané volání, funguje i jakékoli jiné volání v celé aplikaci, nemá tedy velký význam tento proces dále popisovat. Ačkoli je však zdrojový kód jednoduchý, většinou doslova sebevysvětlující, najde se v něm i pár zajímavých řádků. Právě těm bude věnován prostor ve zbytku této kapitoly.

První zajímavost lze najít například na řádku 430 v souboru zapis_znamek.py, vyskytuje se však v každém z views. Jde o samotný důvod pro použití slovníku kompletni_context, místo toho, aby byla pro poslání kontextu do template jednoduše použita request.session['context']. Důvodem k tomuto, zdánlivě zbytečnému, kroku je request.session samotná, jelikož, ač je to dictionary-like objekt, není slovník v pravém slova smyslu. Od klasického dictionary, např. právě kompletni_context, se totiž kromě metody update() liší také tím, že nemůže obsahovat callable objekty. V aplikaci tak vzniká problém, je-li do request.session potřeba zapsat například form(context), tedy formulář s daty, kontextem, který by měl být následně poslán do template. Toto by bylo použitím samotné request.session neproveditelné, a musí být tedy zvolen postup, kdy se do request.session sice zapisuje drtivá většina potřebných dat, sérií podmínek se pak ale vybere „zbylý kontext“ pro danou template, který je daty z request.session up-

datován, čímž vznikne kontext kompletní, který již do template být zaslán může. Možnost poslat do template libovolná data je totiž klíčová, bez ní bychom nebyli schopni vytvořit ani fungující formuláře.

Další zajímavost je Django login_required dekorátor, tedy @login_required jenž se nachází skoro před každou z view funkcí. Tento dekorátor se za nás u každé view funkce stará o autentizaci uživatele. Je-li funkce označena tímto dekorátorem, provede se její obsah pouze v případě, že je uživatel přihlášen; v opačném případě je přesměrován na login.

Následující zajímavé řádky jsou v souborech tridni_kniha.py a zapis_znamek.py, v zapis_znamek.py konkrétně na řáku 442. Zde jde o snahu neduplikovat příliš kód. V kódu bylo potřeba vytvořit pro kompletni_context sérii objektů formuláře s požadovanými daty, jelikož jde však o jednu a tu samou operaci pro každý z formulářů, bylo by velmi neelegantní psát stejný kód několikrát za sebou. Situace byla vyřešena použitím for cyklu, který pro každou dvojici ze seznamu tuple vytvoří odpovídající zápis do kompletni_context. Malá komplikace při tomto postupu nastává však ve chvíli, kdy je nutné vytvořit právě dvojici form(context). Název požadovaného formuláře je totiž znám, jeho objekt však nikoli. Proto je třeba daný objekt hledat v globals(), tedy v globálním jmenném prostoru daného modulu. Tímto způsobem lze snadno a rychle vytvořit potřebnou sérii formulářů bez zbytečné duplikace kódu.

Jak již bylo jednou zmíněno, je při počítání průměrů známek studentů nutno převést stringovou hodnotu známek na float. Tento převod je zaručen řádkem 39 v souboru prehled_studia.py. V podstatě se jedná o pouhý for cyklus, který pro každou známku studenta, tedy první hodnotu z tuple (číslo známky, zobrazená známka), vybere odpovídající floatovou hodnotu ze seznamu float_znamky, a vytvoří tak seznam známek ve floatové podobě, se kterým se pak pracuje při samotném výpočtu průměru. Ač se jedná o triviální proces, je pro daný výpočet nezbytný.

Poslední zajímavost, která bude zmíněna, je funkce pdf nacházející se na řádku 574 v souboru tridni_kniha.py, určená pro generování pdf. Sama o sobě není ničím výjimečným, v podstatě pouze vykreslí šablonu a do každého pole zapíše jméno studenta.

Co však už za zmínku stojí je fakt, že ReportLab, tedy nástroj starající se o samotné generování pdf, defaultně nepodporuje české znaky. Kvůli tomuto nedostatku je nutné pro správné fungování dané funkce importovat font, který je podporovat bude. Zde však vzniká problém při nasazení aplikace na jiné distribuci OS Linux než Fedora 14, ve které byla vyvíjena. Jelikož mají různé distribuce fonty často umístěné v lehce rozličných složkách, nastane při startu serveru chyba způsobená právě nemožností načíst požadovaný font, v případě této aplikace jde o DejaVuSerif. Má-li být tedy aplikace provozována jinde než na distribuci Fedora 14, je nutné tuto část kódu zkontrolovat, a případné upravit.

8 Template

Posledním pilířem MTV architektury je *„T, tedy Template, prezentační vrstva. Tato vrstva obsahuje vše, co se týká prezentace, tzn. jak se má něco zobrazit na webové stránce nebo v jiném typu dokumentu.“*[8] V aplikaci jsou templates pochopitelně použity právě k definování dynamického obsahu, a částečně vzhledu, jednotlivých webových stránek. Následující dvě kapitoly se budou věnovat zaprvé Django template language a výhodám jejího použití, zadruhé pak příkladům jejího konkrétního využití v aplikaci. Jelikož jsou templates, stejně jako zbytek aplikace, velmi jednoduchou záležitostí, a jejich kód je v podstatě sebevysvětlující, budou v popisu jejich využití zmíněny pouze ty zajímavější části kódu.

8.1 Django template language a její výhody

Django template language je jazyk sloužící k jednoduchému a rychlému vytváření dynamického obsahu stránek. Mezi jeho hlavní přínosy patří možnost použití variables, tedy proměnných, tagů, tedy značek, mezi jejichž funkce patří např. podpora základních logických operací, filtry, a hlavně také dědičnost.

Ačkoli podporuje template language jisté logické operace, je nutné brát v potaz filosofii návrhu tohoto jazyka, a to že se jedná o nástroj vytvářející prezentační část aplikace, nikoli programovou logiku. Nachází se zde tedy pouze opravdu základní operace, jako jsou podmínky či for cykly (kompletní seznam tagů viz [21]). I ty však velmi ulehčí práci při vytváření obsahu, navíc je díky jejich využití možné vytvářet obsah opravdu dynamicky; v elektronické třídní knize jsou hojně využívány. Veškeré složitější operace však musí být vyřešeny ve Views.

Proměnné jsou to, co dává v praxi šablonám jejich obsah. V kódu je proměnná označena {{ proměnná }}, a k jejím atributům se přistupuje pomocí tečkové notace. Proměnné jsou do template poslány jako kontext z View, a mohou mít tedy jakoukoli podobu. Fakt, že lze pomocí kontextu do template poslat jakékoli proměnné je jedním z důvodů proč nepotřebuje template language provádět složitější logické operace. Vše

jde jednoduše vyřešit logikou ve views a poté požadovaný výsledek zaslat template jako proměnnou; logika a vzhled jsou tak krásné odděleny.

Filtry v podstatě pouze upravují zobrazení jednotlivých proměnných, upravují jejich formát. Bude-li např. potřeba proměnnou s datem ve formátu „*16. březen 2010*“ zobrazit ve formátu „*16.03.2010*“, stačí použit filtr {{ proměnnáldate:“d.m.Y“ }}.

Jednou z nejvýznamnějších vlastností template language je však již zmíněná dědičnost. Dědičnost totiž umožňuje vytvořit jednu základní šablony, tzv. base template, obsahující přepisovatelné bloky kódu, které mohou být upraveny jejími potomky. Základní šablona pak může tedy obsahovat pouze vzhled společný pro všechny stránky webu, tedy například hlavičku, patu, navigaci a podobně, zatímco samotný obsah je právě oním měnitelným blokem, který je rozdílný pro každou z částí webu. Tyto jednotlivé části, tedy samostatné templates které pomocí tagu {% extends %} rozšiřují base template, se pak mohou zase měnit v závislosti na aktivitách uživatele, hloubka dědičnosti totiž není omezena. Tento přístup značně zjednodušuje orientaci ve struktuře webu, jelikož nejde o jediný masivní kus kódu, nýbrž o sérii hierarchicky rozdělených templates. Takovýto přístup navíc podporuje princip DRY, jelikož již ze své podstaty zamezuje zbytečné duplikaci kódu.

8.2 Využití template language

Z pohledu na kteroukoli z templates vytvořených pro aplikaci třídní knihy, umístěných v adresáři /bp/templates/ a /bp/templates/trst/ na přiloženém nosiči, je jasné, že byla template language využívána velmi často. Samotné zaměření aplikace totiž není již ze své podstaty slučitelné s myšlenkou statických webových stránek; pro její smysluplnou funkčnost je bezesporu vyžadován obsah dynamický. To je přesně pole, kde Django template language exceluje, a kvůli čemu byla template language v aplikaci tak hojně využita.

Jedním z typických příkladů, kde byla template language využita, je generování obsahu různých tabulek. Jelikož není v drtivé většině případů předem známo, kolik bude mít daná tabulka položek, nelze ji proto vytvořit klasickým způsobem, je při

vytváření tabulek v aplikaci využito for cyklů. Z kontextu zaslaného do template je načtena proměnná, v takovémto případě se jedná o seznam hodnot vypočtený ve views, a pro každou jeho položku je vytvořen požadovaný řádek tabulky. Princip je to naprosto triviální, efektivně lze však díky němu vytvářet prakticky jakékoli standardizované výstupy. Tento příklad také dobře ukazuje myšlenku oddělení logiky od prezentace. Ačkoli je totiž využita základní logická operace, jedná se stále pouze o prezentaci obsahu, nikoli o jeho vytváření.

Dalším typickým využitím template language jsou všemožné podmínky, ty jsou v templates doslova na každém kroku. Obecně vzato jde však o dva druhy podmínek. První se nachází většinou v souborech označených příponou _base, tedy například /bp/templates/trst/tridni_kniha_base.py. Tyto soubory neobsahují obsah stránky jako takový, slouží nýbrž k jeho členění. Prakticky to znamená, že je v takové template obsah rozdělen do menších bloků, většinou se jedná o jednotlivé formuláře, kterým jsou následně určeny podmínky, za kterých se mají tyto části zobrazit. Není-li například v kontextu proměnná vybrana_trida, uživatel tedy nevyplnil se kterou třídou pracuje, nezobrazí se žádný z formulářů na této proměnné závislý, tzn. budou skryty všechny. Obdobně, není-li v kontextu proměnná ucitel_je_tridni, nezobrazí se přihlášenému učiteli tlačítko pro vstup do detailu třídy. Druhým typem podmínek jsou pak ty, které ovlivňují samotný obsah. Zobrazení absence je vhodný příklad takovýchto podmínek. Absence se totiž zobrazuje dvěma způsoby, podle toho, zdali byla omluvena, či nikoliv. V template je toto dvojí zobrazení podchyceno právě podmínkou. Obsahuje-li daná absence hodnotu absence.hx_omluveno = True, má daná buňka tabulky class="abs_o" a v prohlížeči se zobrazí jako velké O v zeleném poli. Je-li absence.hx_omluveno False, ale hodnota absence.hx je stále True, potom má buňka class="abs" a v prohlížeči se zobrazí jako X v červeném poli. Jsou-li obě hodnoty False, je dane pole tabulky prázdné. Je zřejmé, že i zde slouží podmínky pouze k upravování prezentace obsahu, nikoli k jeho vytváření či úpravám.

Ač se použití template language obecně často zúží do užití for cyklů a podmínek, je v kódu alespoň jedna opravdová zajímavost. Tou je autorem vytvořený filtr sloužící k odečítání dvou proměnných, umístěný v souboru /bp/trst/templatetags/odecti.py. Vy-

užit je na řádku 50 v template zapis_znmek.html. Důvod jeho vytvoření je ten, že bylo při vytváření seznamu známek třídy potřeba vytvořit vzhlednou tabulku. Vytvoření seznamu známek a z něho pak pomocí for cyklu vypsat požadovaný seznam nebyl pochopitelně problém. Problém nastal když měla být taková tabulka, tedy tabulka jejíž každý řádek má různý počet sloupců, zobrazena v různých prohlížečích. Jelikož přistupují různé prohlížeče k zobrazení tabulek rozdílně, mnohdy poměrně drasticky, byly výsledky takového výstupu minimálně nedostačující. Správného zobrazení tabulky bylo dosaženo pouze tehdy, měl-li každý její řádek stejný počet sloupců. Takový přístup je však v situaci, kdy dopředu nevíte jaký je největší počet sloupců v tabulce, poněkud nepříjemný. Ve views nebylo z důvodu zcela zbytečného opakovaného výpočtu polí tabulky (první výpočet je pro vytvoření seznamu známek, největšího počtu známek studenta, a největšího počtu známek ve třídě; největší počet známek ve třídě zjistíme až po vyhodnocení všech studentů; druhý pro dopočítání rozdílu mezi počtem známek studenta a největším počtem známek ve třídě pro doplnění tabulky), a tedy zbytečného, a nemalého, zdržení při zobrazení takové tabulky, možné dané hodnoty dopočítat. Byl tedy zvolen přístup, kdy jsou hodnoty porovnávány až v template, kde ovšem pro takovou operaci, tedy odečtení dvou integerových hodnot, neexistuje žádná funkce. Bylo tedy nutné vytvořit nový filtr, který by danou operaci provedl, a na základě jejího výsledku by byl dopočítán colspan jednotlivých řádků. Oním filtrem je pak právě odecti.py. Filtr je jediným řádkem kódu, nemá nejmenší význa jej proto popisovat. I přesto však dobře ukazuje možnosti, které nám Django template language poskytuje, tedy možnost vytvořit vlasní template tagy pro prakticky jakoukoli operaci.

Na závěr je vhodné zmínit, že našla Django template language v aplikaci uplatnění ještě i v trošku jiném směru. Konkrétně díky ní aplikace, resp. její prezentační část, stále dodržuje myšlenku modularity. Jelikož byla aplikace v rámci lepší orientaci ve struktuře, a snaze o zamezení duplikace kódu, roztříštěna do množství templates odpovídající jednotlivým modulům, stala se tak i tato část aplikace plně modulární. Bude-li tedy v budoucnu do aplikace přidán další modul, je jeho zobrazení pouze otázkou vytvoření nové template a podmínky, jenž ji v požadovanou chvíli zobrazí; zbytek kódu zůstane nedotčen. Struktura templates aplikace je znázorněna na obrázku č. 5.

definuje základ HTML dokumentu

base.html

{% block head %}{% endblock %}
{% block navigace %}{% endblock %}
{% block body %}{% endblock %}

podmiňuje zobrazení jednotlivých bloků

tridni_kniha_base.html

{% extends "base.html" %}
{% block head %} ... {% endblock %}
{% block navigace %} ... {% endblock %}
{% block body %}
  {% block vyber_tridy_data_predmetu %}
    {% endblock %}
  {% block seznam_zapisu_hodiny %}
    {% endblock %}
  {% block zapis_hodiny %}
    {% endblock %}
  {% block seznam_absenci %}
    {% endblock %}
  {% block zapis_absenci %}
    {% endblock %}
{% endblock %}

tridni_kniha_detail_base.html

zapis_znamek_base.html

prehled_studia.html

osobni.html

hlavni_strana.html

definují HTML obsah bloků

definují HTML obsah bloků

tridni_kniha.html

{% extends "tridni_kniha_base.html" %}
{% block vyber_tridy_data_predmetu %}
  ... {% endblock %}
{% block seznam_zapisu_hodiny %}
  ... {% endblock %}
{% block zapis_hodiny %}
  ... {% endblock %}
{% block seznam_absenci %}
  ... {% endblock %}
{% block zapis_absenci %}
  ... {% endblock %}

tridni_kniha_detail.html

zapis_znamek.html

Obrázek č. 5 – Struktura templates aplikace

9 Mobilní aplikace

Jedním z požadavků definovaných v zadání této práce bylo, že musí vytvořená aplikace podporovat více platforem, nejenom tedy PC, ale i mobilní zařízení. Následující dvě kapitoly se proto zabývají jednak požadavky mobilních aplikací, tedy co musí aplikace splňovat, aby ji bylo možné považovat za mobile-friendly, také ale konkrétním změnám v aplikaci, jenž musely být provedeny, aby požadovaných vlastností aplikace dosáhla. K zobrazení aplikace na mobilních zařízeních je nutné ještě dodat, že nebylo možné takovéto zobrazení testovat na reálných mobilních zařízeních, při vytváření práce nebyla žádná taková zařízení k dispozici. Aplikace byla proto testována pomocí W3 mobileOK Checker [22] a Firefox iPhone emulátoru [23]. Výsledné mobilní zobrazení tak nemusí být v praxi zcela korektní, bez možnosti testovat aplikaci na reálných zařízeních je toto však nejlepší možný výsledek.

9.1 Požadavky na mobilní aplikaci

Požadavky na mobilní aplikaci, které určovaly úpravy aplikace pro její mobilní použití, byly v podstatě definovány W3 mobileOK Checkerem, jelikož tento nástroj ukazuje validitu a použitelnost daného webu na mobilních zařízeních. V praxi se pak jedná především o použití doctype vhodného pro mobilní zařízení, naprosté oddělení obsahu webu a jeho vzhledu, tedy důsledné používání CSS, dosažení stoprocentní funkčnosti aplikace bez javascriptu, a zmenšení velikosti výsledného webu tak, aby zbytečně nezatěžoval připojení mobilního zařízení.

9.2 Změny v aplikaci pro její mobilní použití

Pro splnění požadavků webu pro mobilní zařízení, a pro úspěšné zobrazení webu na takových zařízeních, muselo být v kódu aplikace provedeno několik změn.

První skupinou změn byly ty, které se týkaly požadavků stanovených W3 mobileOK Checkerem. Pro jejich splnění byl web třídní knihy převeden na XHTML

Basic 1.1 doctype, který je pro mobilní zařízení přímo určen, veškeré vzhled ovlivňující prvky kódu byly přesunuty do CSS šablony, a kód celého webu byl minimalizován, resp. očištěn o bílé znaky, kvůli kterým byl zbytečně objemný. Jelikož byla aplikace již od svého počátku vytvářena tak, aby fungovala bez javascriptu, nemuselo se v tomto směru nic měnit. Po aplikování těchto změn byla aplikace označena jako mobile-friendly.

Druhou skupinou změn byly poté změny, díky kterým se aplikace správně zobrazovala v iPhone emulátoru. Zde se jednalo především o upravení base template aplikace tak, aby byl při požadavku vyslaného z mobilního zařízení místo klasického CSS, tedy /bp/media/css/tridnice.css, načten soubor obsahující CSS pro mobilní zařízení, tedy /bp/media/css/tridnice_mobile.css. Toho bylo docíleno jednoduchou podmínkou, která kontrolovala, je-li proměnná „*mobile*“ True či False. Aby bylo možné mobilní volání však rozpoznat, musela být v souboru /bp/ulstr/views.py a v souboru /bp/trst/views/hlavni_strana.py vytvořena funkce „*mobileBrowser*“ určující user agent, ze kterého volání přišlo. Funkce kontroluje hodnotu „*HTTP_USER_AGENT*“, získanou z META dat příchozího požadavku, vůči seznamu mobilních user agentů. Výstupem této funkce je pak právě požadovaná proměnná „*mobile*“, říkající template zdali se má zobrazit klasicky, či nikoliv. Tato funkce byla převzata z článku Build a Mobile and Desktop-Friendly Application in Django in 15 Minutes, viz [24]. Poslední významnou úpravou bylo poté vytvoření samotné CSS šablony tridnice_mobile.css, která mobilní vzhled definovala. V šabloně byl pochopitelně zachován styl aplikace, musely být však provedeny jisté změny typu zmenšení fontu, zrušení minimálních šířek tabulek, a podobně.

Po těchto úpravách splňovala elektronická třídní kniha požadavky W3 mobileOK Checkeru vždy minimálně na 90 %, aplikace byla tedy uznána jako velmi mobile-friendly, a navíc se správně zobrazovala v iPhone emulátoru. 100% hodnocení ve W3 mobileOK Checkeru se nepodařilo dosáhnout pouze kvůli množství přenášených dat.

10 Zabezpečení aplikace

Aplikace elektronické třídní knihy má sice sloužit hlavně jako prototyp pro jiné obdobné projekty, protože je ale možné ji využít již v její současné podobě, musela být nějakou formou, alespoň do jisté míry, vyřešena i otázka jejího zabezpečení. Jelikož byla aplikace vytvořena pomocí Django web frameworku, je práce spojená s takovým zabezpečením aplikace však značně zredukována, resp. prakticky nulová. Vzhledem k tomu, že je Django profesionálním, a hlavně v praxi hojně využívaným, nástrojem pro tvorbu webových aplikací, přináší sám o sobě řešení této problematiky. Následující kapitola tedy krátce shrne zabezpečení aplikace proti SQL injection, XSS a CSRF. Toto téma je v oficiální dokumentaci dobře popsané, následující text je proto značně zběžný.

10.1 SQL injection

„SQL injection je běžný exploit během něhož útočník změní parametry webové stránky (jako např. GET/POST data či URL), aby do ní vložil vlastní SQL příkazy, jež jsou webovou aplikací spuštěny v databázi. Jedná se pravděpodobně o nejnebezpečnější, zcela jistě také ale bohužel o jeden z nejběžnějších, druhů chyb zabezpečení.“[8]

Ačkoli nejsou v aplikaci třídní knihy žádné ručně psané SQL příkazy, tedy v ní nedochází k situaci, kdy se tato chyba zabezpečení projevuje nejčastěji, je dobré vědět, že jsou data aplikace v bezpečí. Jelikož je podstatou této chyby důvěřování datům přicházejícím do aplikace z prohlížeče, spočívá ochrana jednoduše v kontrolování oněch dat, a zajištění jejich neškodnosti. Přesně to za nás dělá při každém zápisu do databáze Django, jelikož automaticky escapuje veškeré speciální znaky, jenž jsou součástí řetězců, se kterými má databáze pracovat. Podrobněji viz [25].

10.2 XSS (Cross-site Scripting)

„Pravděpodobné nejběžnější druh chyby zabezpečení. Cross-site scripting, nebo také XSS, lze najít ve webových aplikacích, které selžou při escapování obsahu zasla-

ného uživatelem předtím, než jej vyrenderují do HTML. Tato chyba dovoluje útočníkovi do webu vkládat libovolný, většinou však značně zákeřný, HTML kód; většinou formou <script> tagů."[8]

Stejně jako předchozí chyba zabezpečení, i tato staví na slepé důvěře k datům zaslaným do aplikace uživatelem. Díky využití Django template systému se však není třeba čehokoli bát, jelikož Django templates automaticky escapují obsah všech svých proměnných. Situace, kdy se do proměnné dostane cizí kód a bude aplikací vyrenderován, se tak nemůže stát, jelikož bude díky escapování znaků, jako <, >, ‘, “, & a podobně, proměnná vyrenderována jako neškodný textový řetězec. Dále viz [26].

10.3 CSRF (Cross-site Request Forgery)

„*CSRF middleware a template tagy poskytují jednoduchou ochranu proti Cross-site Request Forgery. Tento druh útoku se vyskytne, když obsahuje zákeřná webová stránka link, tlačítko formuláře, nebo javascript, který je určen k provedení nějaké operace na Vašem webu za použití údajů již přihlášeného uživatele, který takový zákeřný web navštíví ve svém prohlížeči. Útok spojený s tímto, ‘login CSRF’, kde útočící stránka donutí uživatelův prohlížeč přihlásit se do webu za pomocí přihlašovacích údajů někoho jiného, je také pokryta.*"[27]

U elektronické třídní knihy je pochopitelně nutné zajistit, že data do aplikace odesílá pouze povolaný uživatel, učitel, a že přichází pouze z dané aplikace, ne z jiných zdrojů. Ochrana proti CSRF v ní má tedy reálné opodstatnění. Jak ale ukazuje předchozí odstavec, Django framework se snaží ošetřit i tuto stránku ochrany aplikace a poskytuje tedy jednak CSRF middleware, který vytváří a kontroluje tajnou hodnotu csrf tokenu a csrf cookie, a csrf token samotný, který do odchozího POST formuláře vloží skryté pole s csrf hodnotou. Pro každý příchozí POST požadavek musí být poté přítomna jak CSRF cookie, tak odpovídající hodnota „*csrfmiddlewaretoken*”, tedy hodnota csrf tokenu. Tímto způsobem je docíleno, že přichází veškeré požadavky do aplikace opravdu z formulářů aplikace a opravdu z prohlížeče přihlášeného uživatele, ne z cizího zdroje. Podrobněji o CSRF viz [28][29] a o Django CSRF ochraně viz [27].

11 Aplikace v produkčním prostředí

Předchozí kapitoly popisovaly teorii kolem vytvořené aplikace, či její kód; žádná z nich však nepopsala konkrétní postup jejího spuštění, popis jejího nasazení do produkčního prostředí. Následující kapitoly proto popíší dva způsoby, jak ji do takového prostředí převést. Konkrétně jde o zprovoznění na Fedoře 14 a na CentOS 5.5. Tedy prostředí, ve kterém byla vytvořena, a prostředí, ve kterém běží její zkušební verze na serveru TUL. Ač zde budou zmíněna pouze tyto dvě prostředí, existuje prostředí, ve kterých lze aplikaci úspěšně nasadit, bezpočet; popis všech je však pochopitelně mimo rámec a možnosti této práce.

11.1 Provoz ve vývojovém prostředí

Má-li být aplikace schopná fungovat v produkčním prostředí, musí pochopitelně fungovat i v prostředí vývojovém. Tato kapitola ukáže jak dostat aplikaci do stavu, kdy je připravena pro nasazení na Apache serveru, tedy kdy je provozuschopná právě ve vývojovém prostředí. Kroky spojené s vytvořením a provozováním databáze na jednotlivých databázových serverech zde nebudou probírány, není to předmětem této kapitoly; pro jejich popis navštivte oficiální dokumentaci.

Pro zprovoznění vytvořené aplikace na Django vývojovém serveru je zapotřebí nainstalovat následující nástroje a knihovny – Django 1.2.5, minimálně Python 2.4 (doporučen je však alespoň Python 2.5, jelikož má k sobě již přibalenou SQLite3; pokud bude tedy použita starší verze, je nutné ještě nainstalovat balíček python-setuptools, který umožní instalaci balíčku python-sqlite2; bez python-sqlite2 není u Pythonu 2.4 možné spustit aplikaci s SQLite3 databází), sadu DejaVu fontů, a python-reportlab.

Jelikož byla aplikace třídní knihy popisovaná v této práci vytvořena v distribuci Fedora 14, je pochopitelné, že právě v ní nejsou žádné problémy při jejím zprovoznění, a tedy připravení pro následné nasazení do produkčního prostředí. Celý proces by se dal shrnout do následujících tří kroků:

1. `yum update`

2. `yum install Django`
3. `yum install python-reportlab`

Nic víc není na Fedoře 14 potřeba. Python 2.7 a DejaVu fonty jsou defaultně součástí instalace této distribuce, jediné co je tedy potřeba nainstalovat je Django a python-reportlab. Fakt, že je vhodné pracovat s naupdatovaným systémem, nemá snad význam ani zmiňovat. Aplikace v tuto chvíli běží na Django vývojovém serveru s SQLite3 databází.

Bude-li však potřeba aplikaci zprovoznit na CentOS 5.5, distribuci používanou v době psaní této práce na TUL, vyskytne se problémů hned několik. Část těchto potíží je způsobena tím, že CentOS 5.5 používá starší verzi Pythonu, konkrétně python 2.4.3, zbytek je způsoben tím, že nelze Django 1.2.5 defaultně na CentOS ani nainstalovat. Fakt, že má CentOS lehce odlišnou cestu k fontům je taktéž problém. Postup zprovoznění aplikace na této distribuci vypadá tedy následovně:

1. `rpm -Uvh http://download.fedora.redhat.com/pub/epel/5/i386/epel-release-5-4.noarch.rpm`
2. `yum update`
3. `yum install python-setuptools python-reportlab`
4. `yum install python-sqlite2`
5. `wget http://pypi.python.org/packages/source/D/Django/Django-1.2.5.tar.gz`

   `tar -xf Django-1.2.5.tar.gz`

   `python setup.py install`
6. zkontrolovat, je-li v souboru /bp/trst/views/tridni_kniha.py na řádku 17 uvedena správná cesta k fontu DejaVuSerif, pokud ne, je nutné ji opravit

Jak je vidět, není tento postup zdaleka tak jednoduchý jako u Fedory 14; Aplikace s SQLite3 databází však i zde ve finále ve vývojovém prostředí funguje. Podobné potíže jako u CentOS 5.5 se dají při zprovozňování aplikace bohužel očekávat i u jiných distribucí OS Linux. Zde je však nutné zdůraznit, že nejsou tyto potíže způsobeny aplikací! Aplikace byla vytvořena v prostředí, kde byly aktuální verze potřebných balíčků, a je tedy pochopitelné, že je bude pro svůj bezproblémový běh vyžadovat. Pří-

klad nasazení na CentOS 5.5 měl tedy hlavně ukázat, že je možné aplikaci zprovoznit i v nevyhovujícím prostředí; je nutné se však připravit na, mnohdy značné, komplikace.

11.2 Nasazení na Apache serveru

Podaří-li se aplikaci zprovoznit ve vývojovém prostředí, zbývá už jen krůček od zprovoznění v prostředí produkčním. Jako produkční server byl v tomto případě vybrán Apache server s mod_wsgi jelikož se jedná o nejběžněji používaný web server, a protože mají Django projekty s mod_wsgi dobré výsledky. Django projekty je možné na Apache serveru zprovoznit ještě za pomoci mod_python, vzhledem k odezvě online komunity na toto řešení je to však považováno za horší ze dvou zmíněných, a nebude zde tedy probíráno.

Samotné zprovoznění aplikace na Apache serveru není naštěstí v podání Fedory 14 a CentOS 5.5 tak rozdílné, je resp. vlastně zcela stejné. Jediné, co je potřeba pro nasazení na Apache udělat, je:

1. `yum install httpd mod_wsgi`
2. vytvoření kopie složky aplikace, která bude patřit uživateli a skupině apache, a ke které bude mít tento uživatel maximální práva a hlavně přístup
3. úprava cest v settings.py tak, aby se správně odkazovaly právě na tuto kopii
4. vytvoření souboru django.wsgi umístěného nejlépe ve složce apache v oné kopii projektu, v aplikaci se jedná o /bp/apache/django.wsgi, obsahující následující řádky:

```
import os, sys
sys.path.append('/cesta/k/projektu/')
os.environ['DJANGO_SETTINGS_MODULE'] ='nazev.projektu.settings'
import django.core.handlers.wsgi
application = django.core.handlers.wsgi.WSGIHandler()
```

5. vytvoření bp.conf ve složce /etc/httpd/conf.d s následujícím obsahem:

```
alias /media/ /cesta/ke/kopii/projektu/media/
<Directory /cesta/ke/kopii/projektu/media/>
```

```
Order deny,allow
Allow from all
</Directory>
WSGIScriptAlias / /cesta/ke/kopii/projektu/apache/django.wsgi
<Directory /cesta/ke/kopii/projektu/apache/>
Order deny,allow
Allow from all
</Directory>
```

6. konfigurace zbytku Apache serveru dle potřeb uživatele
7. odkomentování řádku LoadModule v souboru /etc/httpd/conf.d/wsgi.conf

To je vše. Nejedná se tedy o nic jiného než o instalaci mod_wsgi, vytvoření kopie projektu patřící Apache serveru, vytvoření wsgi skriptu, který se bude starat o správné spuštění Djanga v Apachi, následovné definování, kde se daný skript nachází a jak s ním zacházet, kde se nachází složka s médii Django aplikace, a nakonec odkomentování načtení samotného wsgi modulu do Apache.

V tuto chvíli by měla být aplikace plně funkční i na produkčním serveru. Vyskytnou-li se nějaké potíže, jde především o odepření přístupu do aplikace. To může být způsobeno tím, že nemá Apache dostatečná práva pro přístup do složky nadřazené složce s projektem, nebo proto, že je kopie projektu patřící Apachi nevhodně umístěna a SELinux k ní tak Apachi přístup odepře.

Závěr

Hlavním cílem této bakalářské práce bylo vytvoření plně funkční webové aplikace zastupující funkci klasické třídní knihy. Tento cíl byl, jak dokazuje nejenom tento text, nýbrž i zdrojový kód a zkušební verze této aplikace, splněn. Kromě samotného vytvoření aplikace byl pak také učiněn první krok pro její reálné využití coby prototypu pro jiné obdobné projekty – zdrojový kód aplikace byl zveřejněn na serveru SourceForge, kde k němu budou mít případní zájemci neomezený přístup.

Ač je v textu dobře popsáno, co by mělo být přínosem vytvořené aplikace, a k čemu by mohla být v budoucnu využita, vidím po dokončení této práce jako její největší přínos to, že mi práce na ní dovolila, v podstatě jako první za celou dobu mého studia na TUL, osvojit si mnoho z nástrojů a přístupů využívaných při reálné tvorbě aplikací. Nejde tedy pouze o osvojení Django web frameworku, a s ním nerozlučně spojeného jazyka Python, nýbrž i zásad pro tvorbu webových aplikací obecně, či třeba nutnost vývoj takové aplikace řádně plánovat. Fakt, že bylo v aplikaci využito v průběhu jejího vývoje několik různých databázových serverů, různých webových nástrojů a přístupů, že byla ve finále tato práce vysázena pomocí nástroje LaTeX, a že bylo nutné aplikaci testovat na různých Linuxových distribucích, také značně rozšířil mé znalosti.

Jelikož se vytvořená aplikace v současné podobě skládá v podstatě pouze ze tří modulů, je logickým krokem snažit se v případné práci navazující na tuto o rozšíření aplikace do fáze „*systému*“, jak bylo popsáno v kapitole 4.2.1 Oblasti možného rozšíření aplikace, nebo o zakomponování stávajících modulů do systému jiného.

Reference

[1] NEUMAJER, Ondřej. *Školní informační systémy* [online].
URL: <http://clanky.rvp.cz/clanek/c/Z/8019/skolni-informacni-systemy.html/>
[cit. 2011-4-12].

[2] *Gaudeamus – O programu* [online].
URL: <http://www.isgaudeamus.cz/o-programu.html> [cit. 2011-4-12].

[3] *Bakaláři – Třídní kniha* [online].
URL: <http://www.bakalari.cz/trkniha.aspx> [cit. 2011-4-12].

[4] *Etřídnice – Informační systém pro školy - Etřídnice* [online].
URL: <http://www.etridnice.cz/o-etridnici/> [cit. 2011-4-12].

[5] *iškola.cz – Doprovodné texty* [online].
URL: <http://www.iskola.cz/texty/coumi.php> [cit. 2011-4-12].

[6] *MP-Soft a.s. – SAS* [online].
URL: <http://www.mp-soft.cz/index.php?id=sas%2Fcast0&mf=5002048>
[cit. 2011-4-12].

[7] *Django | The Web framework for perfectionists with deadlines* [online].
URL: <http://www.djangoproject.com/> [cit. 2011-4-12].

[8] HOLOVATY, Adrian. – KAPLAN-MOSS, Jacob.
The Definitive Guide to Django: Web Development Done Right, Second Edition.
USA: Apress, červenec 2009. 536 s. ISBN-13: 978-1-4302-1936-1

[9] ALCHIN, Marty. *Pro Django.*
USA: Apress, prosinec 2008. 320 s. ISBN-13: 978-1-4302-1047-4

[10] *Python – About* [online].
URL: <http://www.python.org/about/> [cit. 2011-4-12].

[11] *Python – Wikipedie* [online].
URL: <http://cs.wikipedia.org/wiki/Python> [cit. 2011-4-12].

[12] *Python Programming Language – Official Website* [online].
URL: <http://www.python.org/> [cit. 2011-4-12].

[13] *ReportLab Toolkit - ReportLab.com* [online].
URL: <http://www.reportlab.com/software/opensource/rl-toolkit/>
[cit. 2011-4-12].

[14] *jQuery – Wikipedia* [online].
URL: <http://en.wikipedia.org/wiki/JQuery> [cit. 2011-4-12].

[15] *jQuery: The Write Less, Do More, JavaScript Library* [online].
URL: <http://jquery.com/> [cit. 2011-4-12].

[16] *AJAX – Wikipedie* [online].
URL: <http://cs.wikipedia.org/wiki/AJAX> [cit. 2011-4-12].

[17] *Python v2.7.1 documentation – 11.13. sqlite3* [online].
URL: <http://docs.python.org/library/sqlite3.html> [cit. 2011-4-12].

[18] *SQLite Home Page* [online].
URL: <http://www.sqlite.org/> [cit. 2011-4-12].

[19] *Django | How to use sessions* [online].
URL: <http://docs.djangoproject.com/en/1.2/topics/http/sessions/>
[cit. 2011-4-12].

[20] *Django | Writing views* [online].
URL: <http://docs.djangoproject.com/en/1.2/topics/http/views/>
[cit. 2011-4-12].

[21] *Django | Built-in template tags and filters* [online].
URL: <http://docs.djangoproject.com/en/1.2/ref/templates/builtins/>
[cit. 2011-4-12].

[22] *W3C mobileOK Checker* [online].
URL: <http://validator.w3.org/mobile/> [cit. 2011-4-12].

[23] *sandip chitale's blog* [online].
URL: <http://blogs.sun.com/scblog/entry/iphone_emulator_for_firefox>
[cit. 2011-4-12].

[24] *Build a Mobile and Desktop-Friendly Application in Django in 15 Minutes*
URL: <http://mobiforge.mobi/developing/story/build-a-mobile-and-desktop-friendly-application-django-15-minutes?dm_switcher=%27true>
[cit. 2011-4-12].

[25] *Django | Performing raw SQL queries* [online].
URL: <http://docs.djangoproject.com/en/1.2/topics/db/sql/> [cit. 2011-4-12].

[26] *Django | The Django template language* [online].
URL: <http://docs.djangoproject.com/en/1.2/topics/templates/> [cit. 2011-4-12].

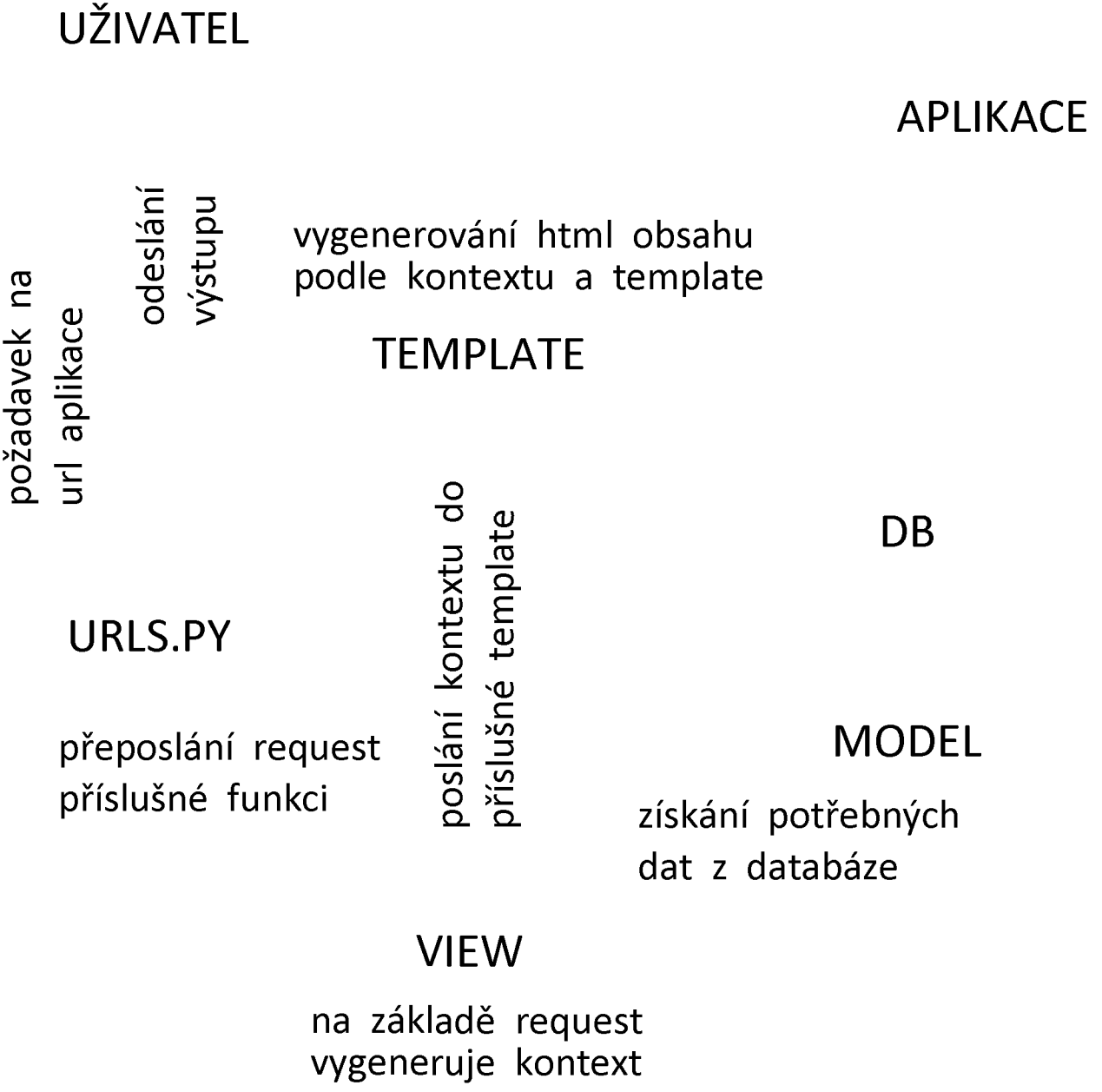
[27] *Django | Cross Site Request Forgery protection* [online].
URL: <http://docs.djangoproject.com/en/1.2/ref/contrib/csrf/> [cit. 2011-4-12].

[28] *Cross-site request forgery – Wikipedia* [online].
URL: <http://en.wikipedia.org/wiki/Cross-site_request_forgery>
[cit. 2011-4-12].

[29] *Security tips for web developers* [online].
URL: <http://www.squarefree.com/securitytips/web-developers.html#CSRF>
[cit. 2011-4-12].

[30] PILGRIM, Mark. *Dive into Python.*
USA: Apress, květen 2004. 413 s. ISBN-13: 978-1590593561

[31] HETLAND, Magnus Lie. *Beginning Python: From Novice to Professional.*
USA: Apress, říjen 2005. 640 s. ISBN-13: 978-1590595190


UŽIVATEL
APLIKACE
požadavek na url aplikace
odeslání výstupu
vygenerování html obsahu podle kontextu a template
TEMPLATE
poslání kontextu do příslušné template
DB
URLS.PY
přeposlání request příslušné funkci
MODEL
získání potřebných dat z databáze
VIEW
na základě request vygeneruje kontext

UČITEL DATUM TŘÍDA PŘEDMĚT

SEZNAM ZÁPISŮ HODINY
- pro daný den / předmět / vše

SEZNAM ABSENCÍ ŽÁKŮ
- pro daný den

ZÁPIS HODINY

ABSENCE

STUDENT

UČITEL
- ev. číslo
- příjmení + jméno
- zkratka
- vyučuje předměty
- třídní ve třídě
- adresa, email atd.

DATUM
text_field

TŘÍDA
- název
select

PŘEDMĚT
- název
- zkratka
select

ZÁPIS HODINY

datum | č. hodiny | předm. | téma hodiny | pozn. | zapsal

čislo hodiny	datum	téma hodiny	předmět	zapsal	zapsat	nový / upravit
select	auto	text_field poznámka text_area	auto	auto		

ZÁPIS ABSENCE

student	h0	h1	h2	h3	h4	h5	h6	h7	h8	zapsat	nový / upravit
student select	checkbox	checkbox	checkbox	checkbox	checkbox	checkbox	checkbox	checkbox	checkbox		

HODINA
- číslo hodiny
- datum
- předmět
- třída
- učitel
- téma
- poznámka

STUDENT
- ev. číslo
- příjmení + jméno
- třída
- adresa, email atd.

ABSENCE
- absence třídy
- absence studenta
- datum
- h0, h1, h2, ..
- h0_o, h1_o, ...

ABSENCE
- absence třídy
- absence studenta
- datum
- h0, h1, h2, ..
- h0_o, h1_o, ... atd.

UČITEL
- ev. číslo
- příjmení + jméno
- zkratka
- vyučuje předměty
- adresa, email atd.

STUDENT
- ev. číslo
- příjmení + jméno
- třída
- adresa, email atd.

PŘEDMĚT
- název
- zkratka

HODINA
- předmět
- třída
- učitel
- téma
- poznámka atd.

TŘÍDA
- název
- třídní učitel

POZNÁMKY
- poznámky studenta
- zadal učitel
- karma atd.

ZNÁMKY
- známky studenta
- známka z předmětu
- známku zadal
- známka atd.

definuje základ HTML dokumentu

base.html

```
{% block head %}{% endblock %}
{% block navigace %}{% endblock %}
{% block body %}{% endblock %}
```

podmiňuje zobrazení jednotlivých bloků

tridni_kniha_base.html

```
{% extends "base.html" %}
{% block head %} ... {% endblock %}
{% block navigace %} ... {% endblock %}
{% block body %}
    {% block vyber_tridy_data_predmetu %}
        {% endblock %}
    {% block seznam_zapisu_hodiny %}
        {% endblock %}
    {% block zapis_hodiny %}
        {% endblock %}
    {% block seznam_absenci %}
        {% endblock %}
    {% block zapis_absenci %}
        {% endblock %}
{% endblock %}
```

tridni_kniha_detail_base.html

zapis_znamek_base.html

prehled_studia.html

osobni.html

hlavni_strana.html

definují HTML obsah bloků

definují HTML obsah bloků

tridni_kniha.html

```
{% extends "tridni_kniha_base.html" %}
{% block vyber_tridy_data_predmetu %}
    ... {% endblock %}
{% block seznam_zapisu_hodiny %}
    ... {% endblock %}
{% block zapis_hodiny %}
    ... {% endblock %}
{% block seznam_absenci %}
    ... {% endblock %}
{% block zapis_absenci %}
    ... {% endblock %}
```

tridni_kniha_detail.html

zapis_znamek.html